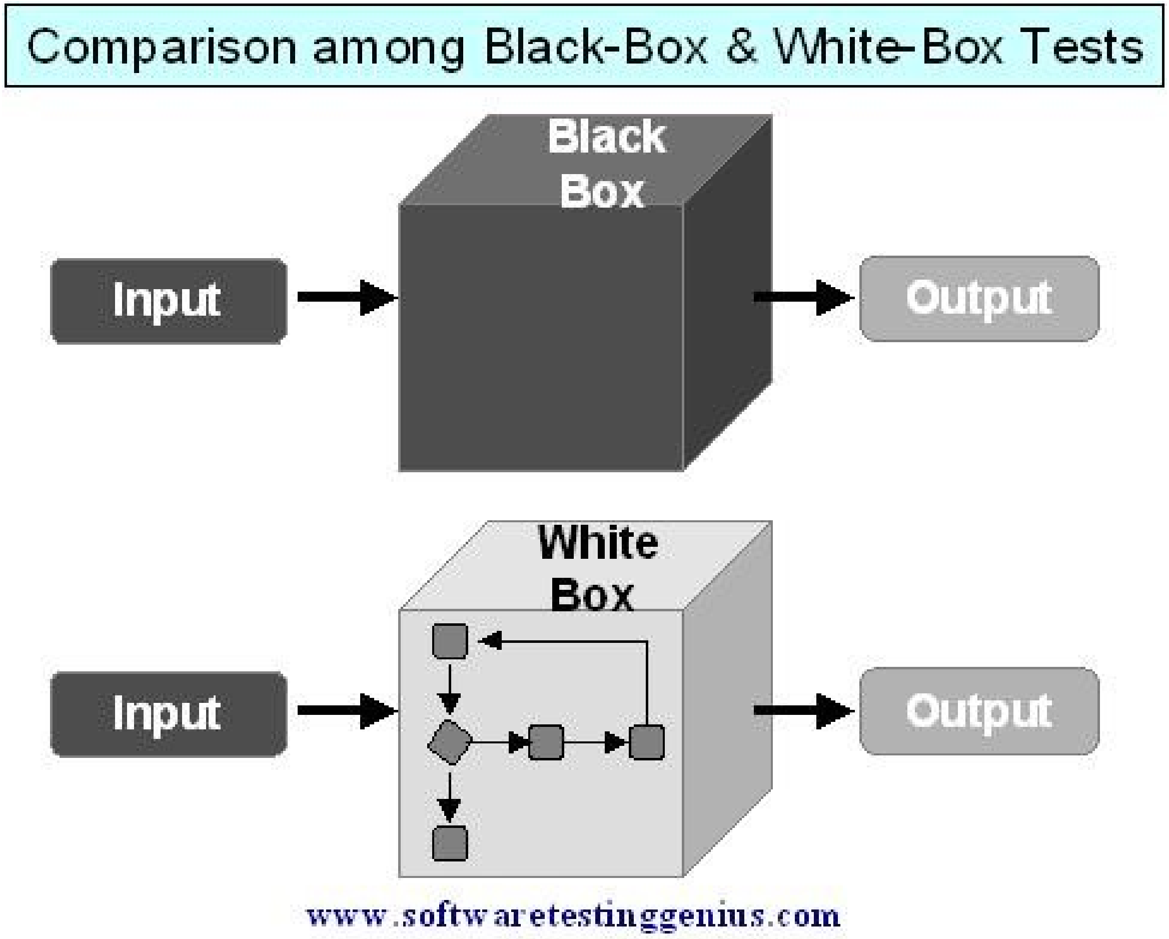
Sommerville Chapter 8
(we will come back here later)

Software Testing: Requirements Based (Black box)

Topics for Today

- Black-Box Tests
- Selecting Black-Box Test Cases

Black and White Box


Comparison among Black-Box & White-Box Tests
Black Box
Input
Output
White Box
Input
Output
www.softwaretestinggenius.com

Black-Box Testing

If Switch is pressed or Clap is detected, then Light will turn on

1. Switch = Pressed, Light = On

Input **Expected Output**

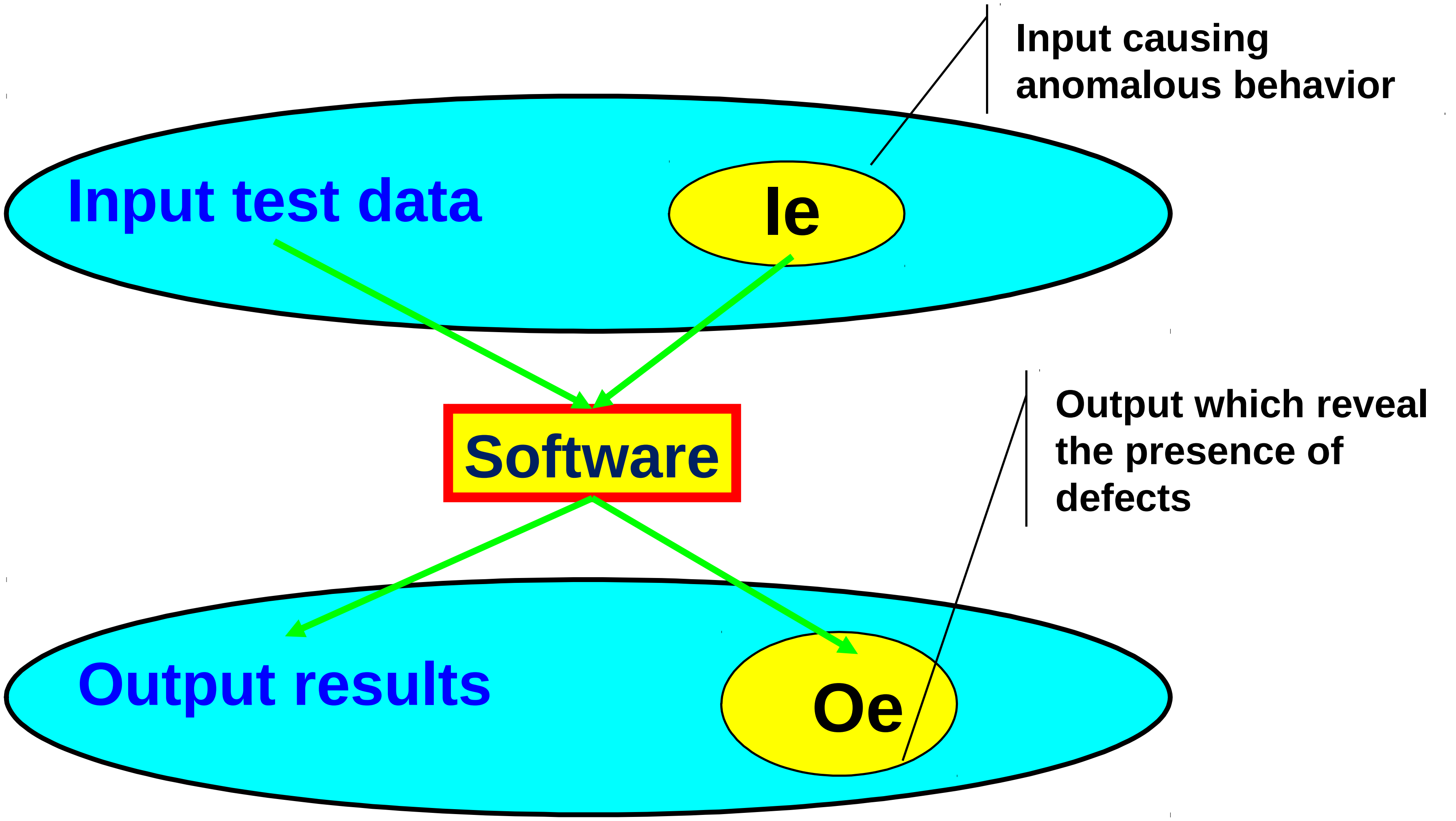
Does passing this test case indicate that the system has correctly implemented the requirement?

2. Clap_Detected = True, Light = On

3. Switch = Pressed, Clap_Detected = True, Light = On

4.. Switch = Not Pressed, Clap_Detected = False

Black-Box Testing


Input causing anomalous behavior
Input test data
Ie
Software
Output which reveal the presence of defects
Output results
Oe

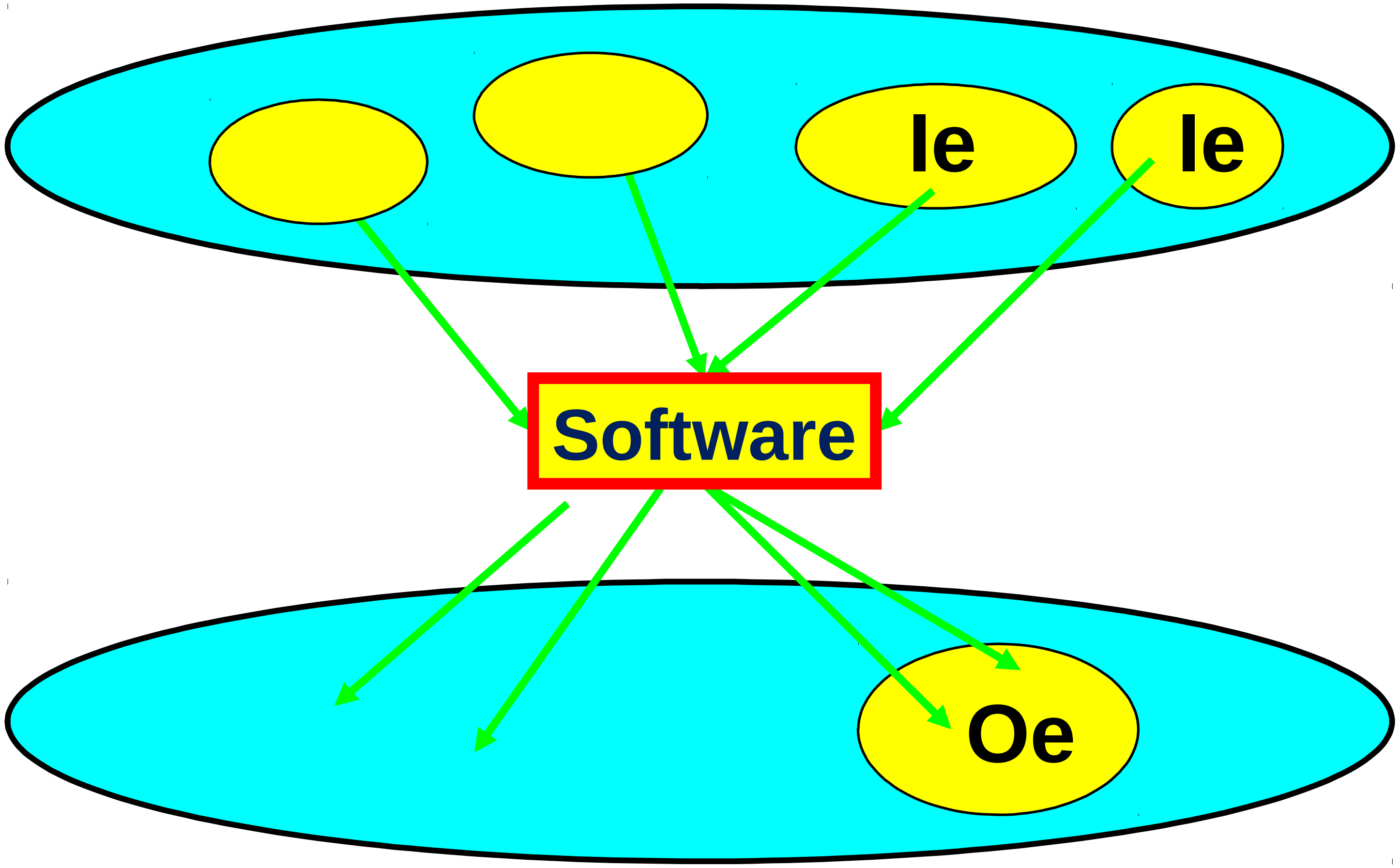
Independently Testable Feature

- A well defined function that can be tested in (somewhat) isolation
  - Identified to “divide and conquer” the complexity of functionality
- Described by all the inputs that form their execution environment

Examples

- Class Registration
  - What are some independently testable features?

Equivalence Partitioning



Equivalence Class?

- A group of tests form an equivalence class if
  - They all test the same thing
  - If one test reveals a fault, the other ones (probably) will too
  - If a test does not reveal a fault, the other ones (probably) will not either

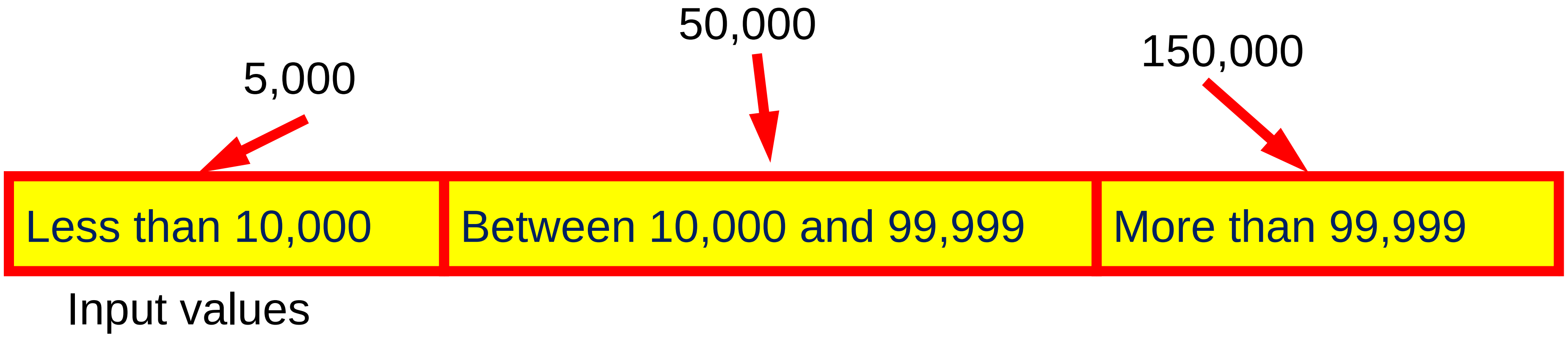
What Goes in an Equivalence?

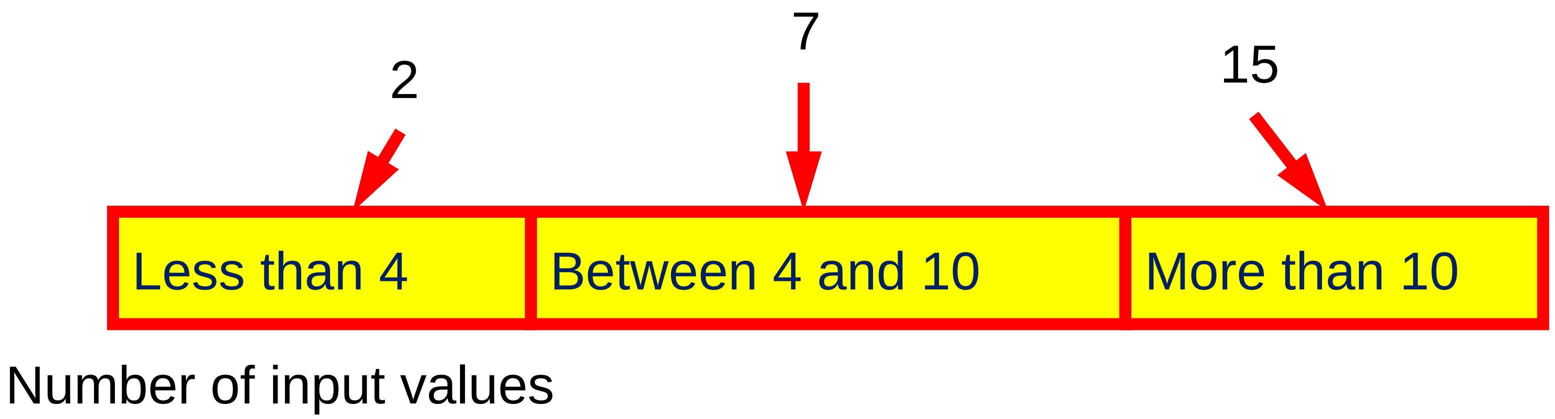
- There must be a good reason to group tests in one equivalence class
  - They involve the same input variables
  - The result in similar (identical?) operations in the program
  - They affect the same output variable
  - None force the program to do error handling, or all of them do

Equivalence Partitioning—1

- Partition system inputs and outputs into “equivalence sets”
  - If input is a 5-digit integer between 10,000 and 99,999, equivalence partitions are <10,000, 10,000-99, 999 and > 100, 000
- Choose test cases in these partitions
  - 5,000, 50,000, 150,000

Equivalence Partitions





Equivalence Classes for Max

```
FUNCTION Max(a IS INTEGER, b IS INTEGER): INTEGER
```

Equivalence Classes for Max

```
FUNCTION Max(a IS INTEGER, b IS INTEGER): INTEGER
```

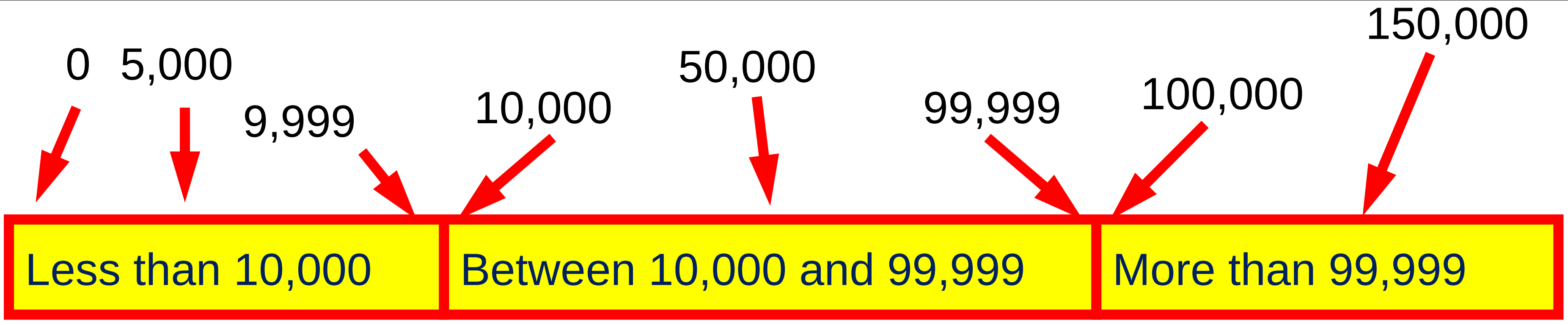
```
FUNCTION Max(a IS INTEGER, b IS INTEGER): INTEGER
        EQUALS a  IF a > b
        EQUALS b  IF a < b
END FUNCTION
```

```
FUNCTION Max(a IS INTEGER, b IS INTEGER): INTEGER
        EQUALS a  IF a > b and a != 4
        EQUALS b  IF a <= b and a != 4
        EQUALS 0 IF a = 4
END FUNCTION
```

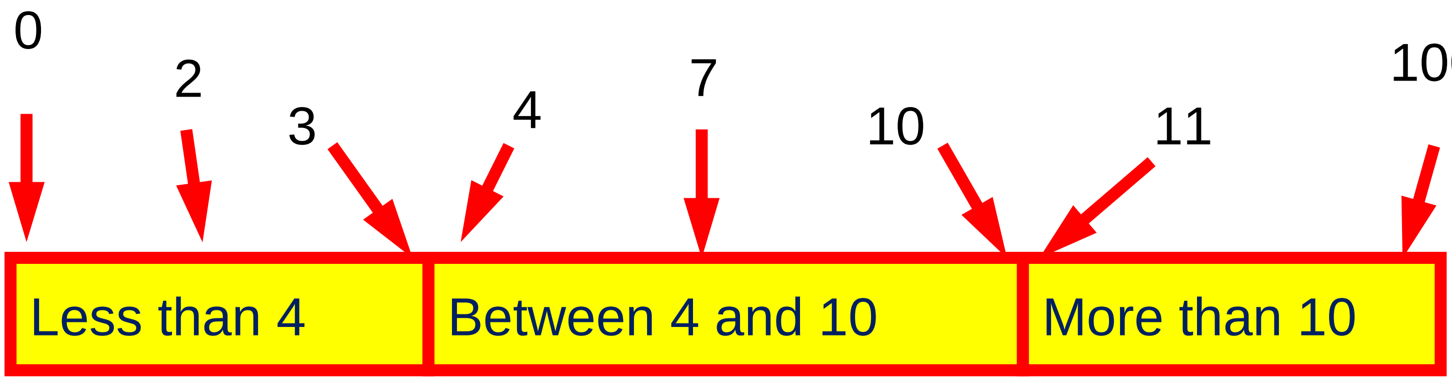
Equivalence Partitioning—2

- Partition system inputs and outputs into “equivalence sets”
  - If input is a 5-digit integer between 10,000 and 99,999, equivalence partitions are <10,000, 10,000-99, 999 and > 100, 000
- Choose test cases at the boundary of these sets also
  - 00000, 5000, 9999, 10000, 10001, 50000, 99999, 100000, 100001, 100001, 150000

Equivalence Partitions Revisited



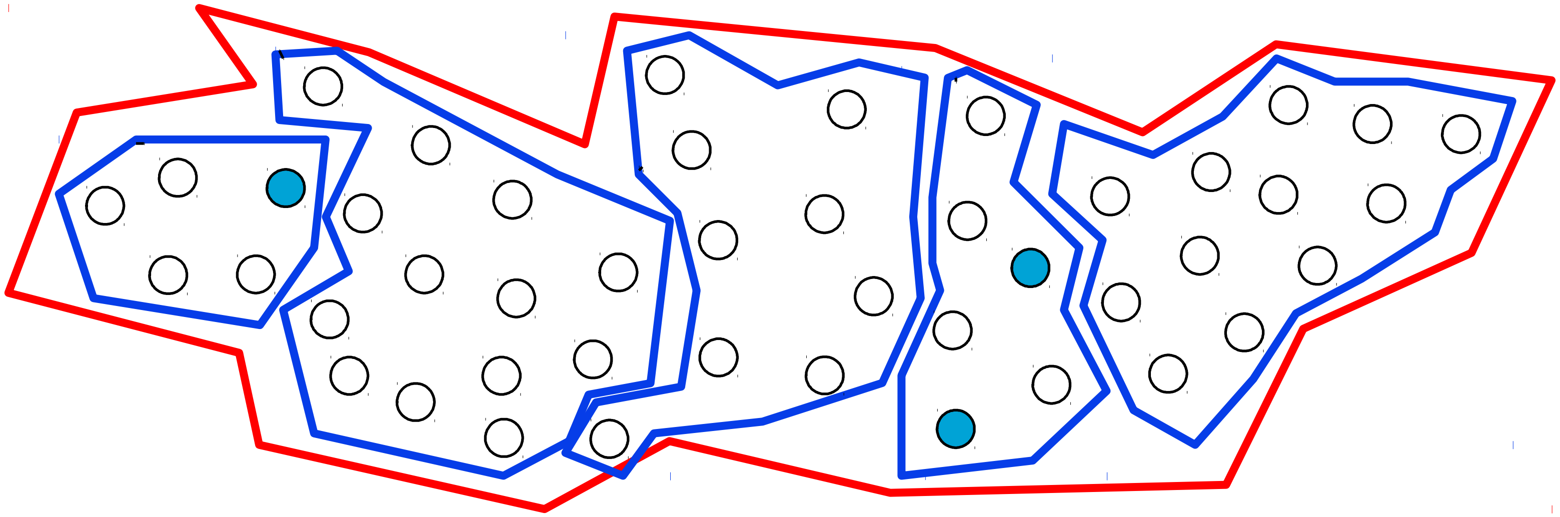
Input values



Number of input values

Partition Testing

- Basic idea: Divide program input space into (quasi-) equivalence classes
  - Underlying idea of specification-based, structural, and fault-based testing



Finding Equivalence Classes

- Look for ranges of numbers
- Look for membership in a group
- Look for time dependent equivalence classes (pump accident)
- Analyze responses to lists and menus
- Look for equivalent output events
- Look for equivalent operating environments
- Do not forget equivalence classes for invalid inputs
- Organize your classification

Look for Ranges of Numbers

- If input is a 5-digit integer between 10,000 and 99,999, equivalence partitions are <10,000, 10,000-99, 999 and > 10, 000
- May want to consider non-numbers as a special equivalence class

Look for Membership in a Group

- Consider the following inputs to a program
  - A valid C++ identifier
  - A letter
  - A country name

- All make up equivalence classes
- All can then be subdivided further
- How?

Time Dependent Classes

- Push the “Esc” key before, during, and after the program is writing to (or reading from) disk
- The timing and duration of an input may be as important as the value of the input
- Very hard and also very critical

Look for Equivalent Outputs

- It may be easier to find good tests by looking at the outputs (work backwards)
- A graphics routine that draws lines on a canvas
  - No line
  - Thin short line
  - Thin long line
  - Thick short line
  - Etc.

Responses to Lists and Menus

- Menu choices naturally partitions the input domain
  - Do you want to print? (Y, Yes, yes, y and N, No, no, n)

- In graphical interfaces you have many combinations of things
  - Topic for a different day

Equivalent Operating Environments

- The environment may effect the behavior of a program

- Memory may effect the program
  - Try with different sized machines
- Processor speed (race conditions)
- Client server environments
  - No clients, some clients, many clients
  - PeopleSoft problems

Do Not Forget Invalid Inputs!

- Most likely to cause problems
  - Exception handling is a well know problem area
  - People tend to think about what the program shall do, not what it shall protect itself against
- Take this into account with all selection criteria we have discussed this far

Organize the Classification

Input/Output	Valid Classes	Invalid Classes
Enter a number (n)	1<= n <= 99	n=0 n>99 n<0 n not a number
Enter first letter of a name	character is capital letter character is lower case letter	character is not a letter
Etc	Etc	Etc

We Have Learned

- Test definitions and language
- Black box testing is concerned with the functional specification of the software
- Equivalence partitions are sets of test cases where the program should behave in an equivalent way
- Use guidelines to help you find good partitions

- Next time
  - Sommerville Chapter 5
  - Web reading
  - Will It Work?