CISC/CMPE 327 Software Quality Assurance

Queen's University, 2019-fall

Part II-2

Blackbox Testing

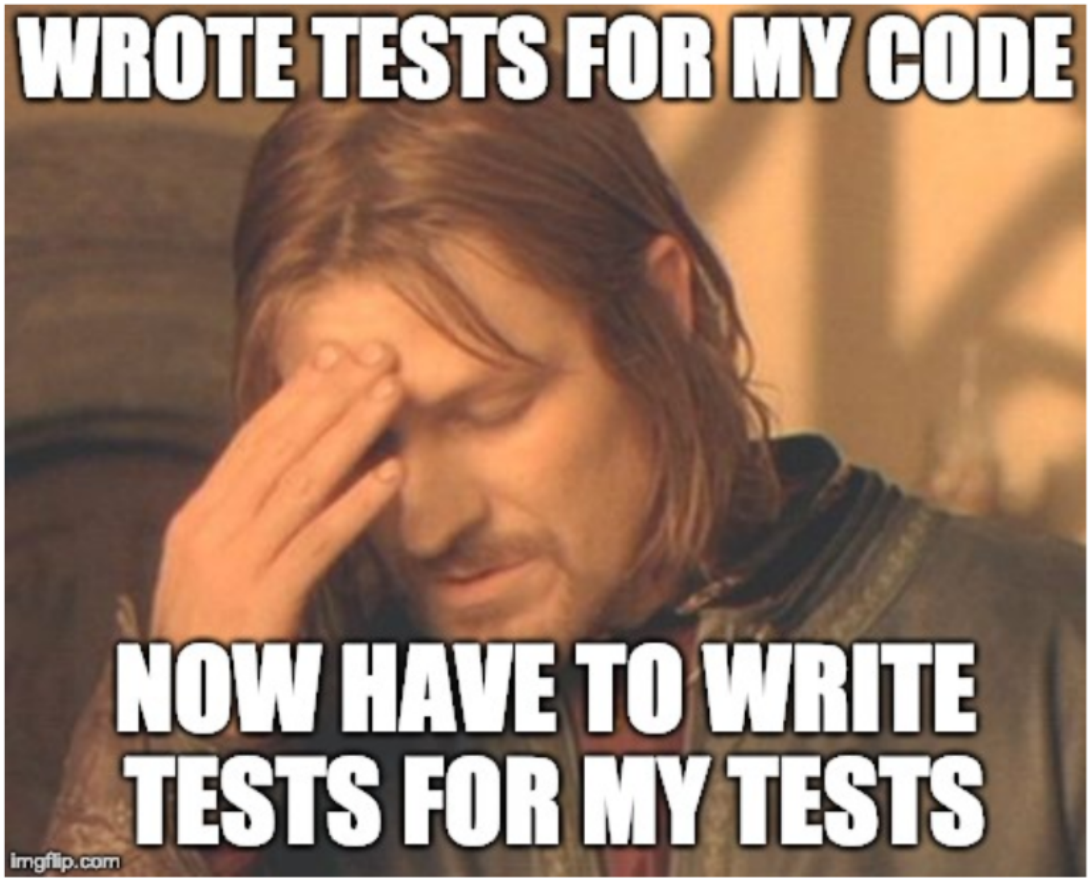
Black Box Testing

- Outline
  - Introduction to testing methods: black box and white box
  - Kinds of black box methods
  - Black box method 1: Systematic functionality testing
  - Black box method 2: Input Coverage

Systematic Testing Methods

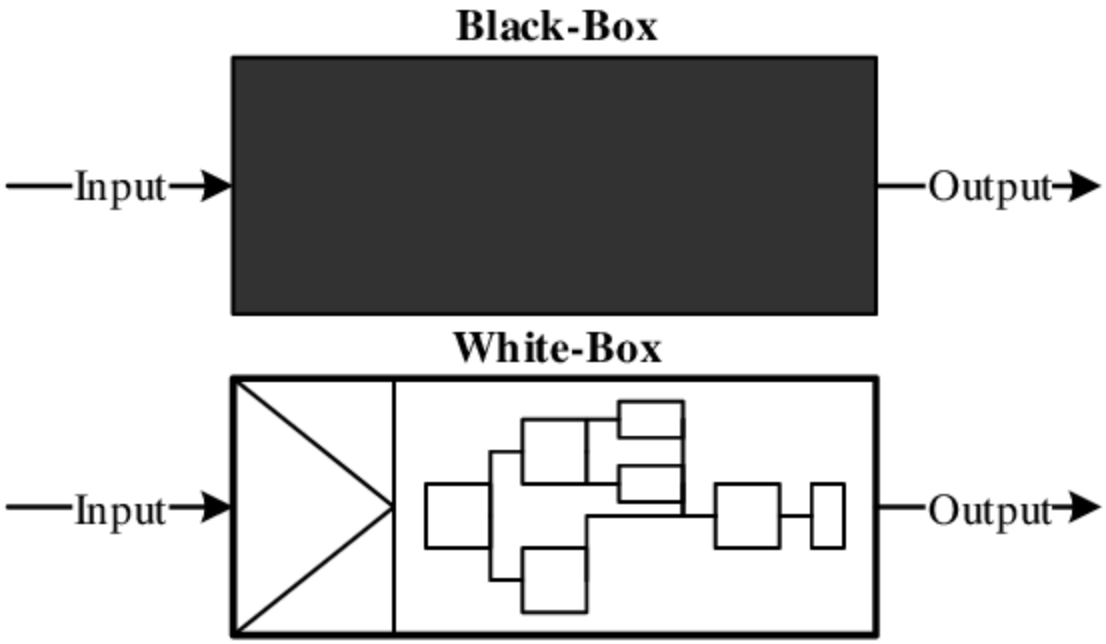
- systematic, it must have:
  - a system (rule) for creating tests
  - a measure of completeness
- What to test
- How to test
- When we're done

- Test Adequacy



Testing in the Life Cycle

- **Black box** testing methods are based on the software's **specifications/functionality**
- **White box** (or glass box) testing methods are based on the software's **code**



Black Box Methods

- Black Box Methods
  - Chosen based on requirements, specification, or (sometimes) design documents
  - Advantage: independently of the software
    - Parallel development of test cases.
  - Based on the functional specification (requirements) for the software system

Black Box Methods

- Functional Specifications
  - Formal (mathematical),
  - Informal (in a natural language)
  - at least three kinds of information:
    - the intended inputs
    - the corresponding intended actions
    - the corresponding intended outputs

Black Box Methods

- Input coverage tests
  - An analysis of the intended inputs
- Output coverage tests
  - An analysis of the intended outputs
- Functionality coverage tests
  - An analysis of the intended actions

Systematic Functionality Testing

- An example
  - partition the functional specification
    - into a set of small, separate requirements
  - Example: Suppose that the informal requirements for a program we are to write are as follows:

    - "Given as input two integers x and y, output all the numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

Requirements Partitioning

- "Given as input two integers x and y"
  - R1. Accept two integers as input.
- "output ... the numbers"
  - R2. Output zero or more (integer) numbers.
- "smaller than or equal to x"
  - R3. All numbers output must be less than or equal to the first input number.
- "evenly divisible by y"
  - R4. All numbers output must be evenly divisible by the second number.
- "all the numbers"
  - R5. Output must contain **all** numbers that meet both R3 and R4.
- "If either x or y is zero, then output zero."
  - R6. Output must be zero (only) in the case where either first or second input integer is zero.

Test Case Selection

- Test Cases for **Each** Requirement
  - Each requirement: independent
  - Example: For the partitioned requirement:
    - "If either x or y is zero, then output zero."
      R6. Output must be zero (only) in the case where either first or second input integer is zero.

A Systematic Method

- Black Box Functionality Coverage
  - a system for creating functionality test cases
    - It tells us when we are done –covered all partitions
  - not the same as acceptance testing
    - a **separated** view of functional requirement
    - Cannot replace acceptance testing
    - It is a systematic method
    - it is only a partial test, like other systematic methods

Catching Errors in Requirements

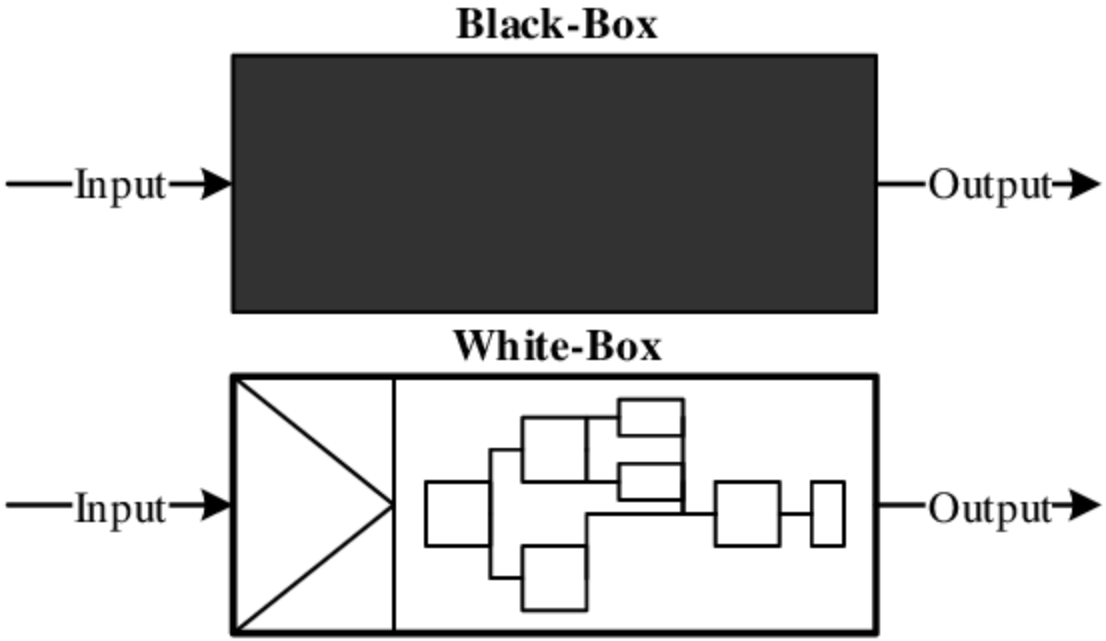
- Systematically partitioning
  - Find/fix missing requirements

- Systematic creation of tests from different points of view
  - Expose problems in the software
  - Expose problems in specification
  - A way to 'test' requirement before running actual text

Black Box Testing

- Black Box Testing
  - Functionality Coverage
  - Input Coverage
  - Output Coverage

Black Box Testing



What is 'input'

- All the possible inputs allowed by the functional specifications (requirements)

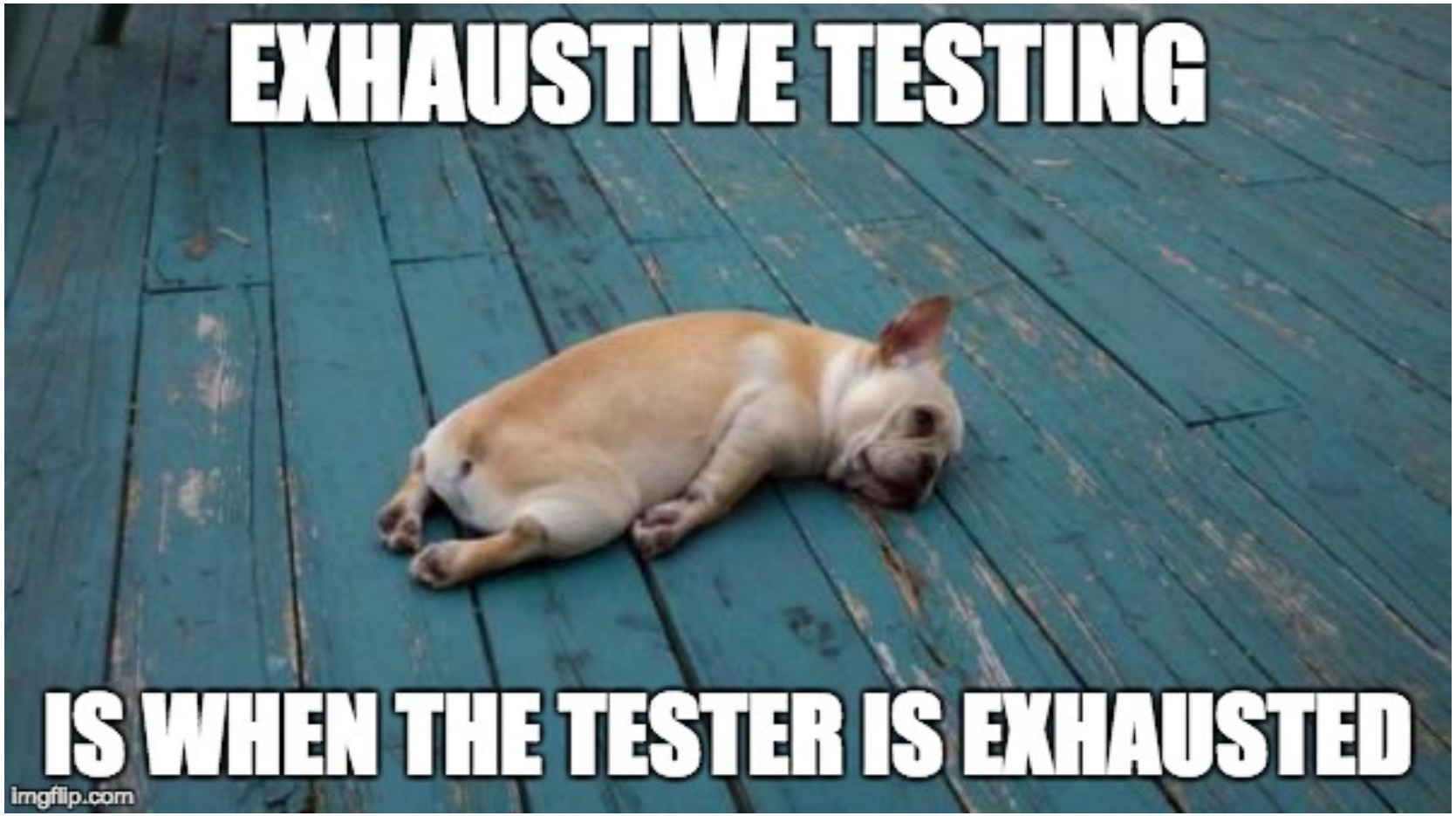
Input Coverage Testing

- Input Coverage
  1. Analyze all the possible inputs
  2. Create test case based on the analysis

  Exhaustive, input partitioning, shotgun, (robustness) boundary

  – Objective: Show software correctly handles all allowed inputs

Exhaustive Testing??



Exhaustive Testing

- What does "all" mean?
  - Cover every possible input to the program
  - Yields a strong result
  - Easy system for test cases, obvious when done

Exhaustive Testing

- Requirement: "return the logical AND result of two boolean inputs"

Exhaustive Testing

- Requirement: "return the logical AND result of two boolean inputs"

C1.	false	false
C2.	true	false
C3.	false	true
C4.	true	true

Exhaustive Testing

"Given as input two integers x and y, output all the positive numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

Exhaustive Testing

"Given as input two integers x and y, output all the positive numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

- Assume that each integer has 32 bits, there are still more than
- 16,000,000,000,000,000,000 pairs to test

Input Partition Testing

- Exhaustive Testing - Practical? **extremely** rare
- Partition all the possible inputs into equivalence classes
  - Share with something in common

Input Partition Testing

"Given as input two integers x and y, output all the positive numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

Input Partition Testing

"Given as input two integers x and y, output all the positive numbers smaller than or equal to x that are evenly divisible by y. If either x or y is zero, then output zero."

Partition	x input	y input
P1	0	nonzero
P2	nonzero	0
P3	0	0
P4	less than zero	less than zero
P5	less than zero	greater than zero
P6	greater than zero	less than zero
P7	greater than zero	greater than zero

Input Partition Testing

- Covering Partitions
  - simplest input values
  - vary them as little as possible

Input Partition Testing

- Covering Partitions

Partition	x partition	y partition	x input	y input
P1	0	nonzero	**0**	**1**
P2	nonzero	0	**1**	**0**
P3	0	0	**0**	**0**
P4	less than zero	less than zero	**-1**	**-1**
P5	less than zero	greater than zero	**-1**	**1**
P6	greater than zero	less than zero	**1**	**-1**
P7	greater than zero	greater than zero	**1**	**1**

Input Partition Testing

- Covering Partitions

Partition	x partition	y partition	x input	y input
P1	0	nonzero	**0**	**1**
P2	nonzero	0	**1**	**0**
P3	0	0	**0**	**0**
P4	less than zero	less than zero	**-1**	**-1**
P5	less than zero	greater than zero	**-1**	**1**
P6	greater than zero	less than zero	**1**	**-1**
P7	greater than zero	greater than zero	**1**	**1**

Do not take into account the intention or actions of the program, only that it handles all its input classes

Advantages of Input Partition Testing

- intuitively for testing
  - Different response to each kind of input
- straightforward to identify a set of partitions

- We know when we are done:

Advantages of Input Partition Testing

- intuitively for testing
  - Different response to each kind of input
- straightforward to identify a set of partitions

- We know when we are done:
  - Partition coverage

- Program can at least handle one example of each different kind of input correctly

Black Box Shotgun Testing

- Black box shotgun testing consists of
  - choosing random input values Repeat for a large number of times
  - Verify outputs and observe crashes
- Practically, legal set and illegal set as separate sets of shotgun tests

Test	x input	y input
T1	682	27631
T2	-89	5244
T3	7368279	-82763

and so on..

Shotgun Testing: Systematic?

- Partition? Coverage?
- Completion?
- Require a very large number of test cases
- Automated verification?

Input Partition + Shotgun Testing

- A Hybrid Method
  - shotgun method to choose random input within each partition
  - additional confidence:
    - simple input values 👍
    - random value 👍
  - Difficulties:
    - automated output verification to be practical
    - simple input first -> then shotgun

Input Partition + Shotgun Testing

- A Hybrid Method

Input Robustness Testing

- Robustness is the property that a program doesn't crash or halt unexpectedly, no matter what the input

- Robustness testing tests for this property
  1. Shotgun robustness testing
  2. Boundary value robustness testing

Input Boundary Testing

- Boundary Values
  - typical failures come at the boundaries of the legal or expected range of values
  - Example (exercise): reverse a linked list!

Input Boundary Testing

- Boundary Values
  - black box testers often create boundary value (edge cases)
  - Boundary value testing is a systematic test method
    - An easy way to choose test cases,
    - an easy way to know when we are done
      - when all boundary tested?