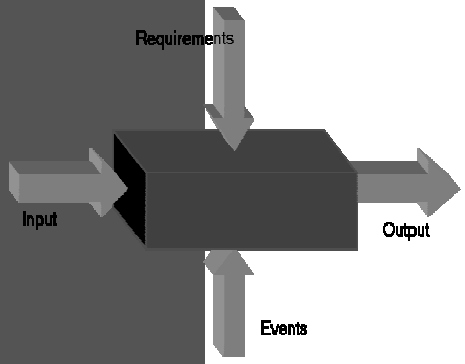
Black-Box Testing

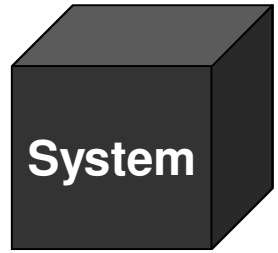


Eshcar Hillel

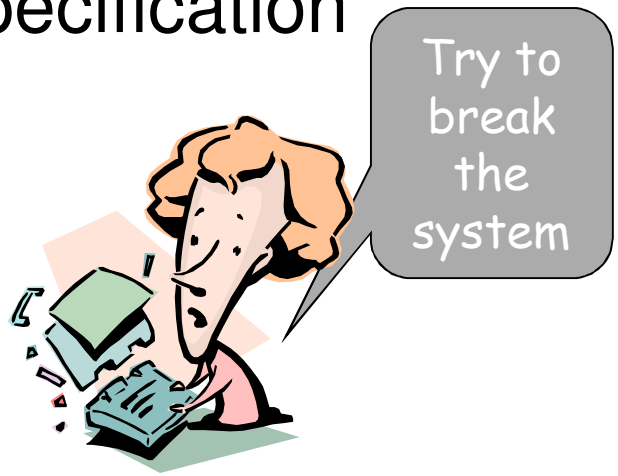
Tutorial Outline

- What is black box testing?
- Testing Paradigms
  - Requirement/use case testing
  - Function testing
  - Domain testing
- Test plan

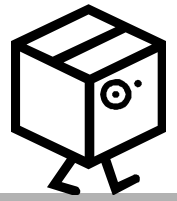
Black-Box means Testing by Specification



- Execution-based testing that treats the system/module as a black box
- Test cases are based upon specification
  - Functional requirements
  - Use cases
  - Class specification
- Test valid and invalid inputs
  - Exhaustive testing is not an option



Testing Paradigms



- The task is to find a subset of inputs that represent all inputs
  - Requirement/use case testing
  - Function testing
  - Domain (boundary) testing
  - Many other techniques
- **They are not mutually exclusive**
  - You may combine overlapping techniques

Requirement and Use Case Testing

- Verify the system's conformance with
  - Requirement document
  - Use case model
  - User manual
  - Customer stories (in extreme programming)
- Reflects the use of the system

The requirements need to be testable

On Site Reading Test Cases

- Trivial case
  - Init catalog with one title with a free copy
  - Select the single copy
  - Expected result: success
- Simple case
  - Init catalog with one title and several free copies
  - Select all the copies
  - Expected result: success
- Complex case
  - Init catalog with several titles and several free copies
  - Select all titles and all copies
  - Expected result: success

On Site Reading Test Cases

- Alternatives
  - No free copies but some are available
    - Held by another reader
    - Being copied by another reader
- Exceptions
  - No free or available copies
  - The reader doesn't take the copy from the robot after a minutes wait

Requirement and Use Case Testing

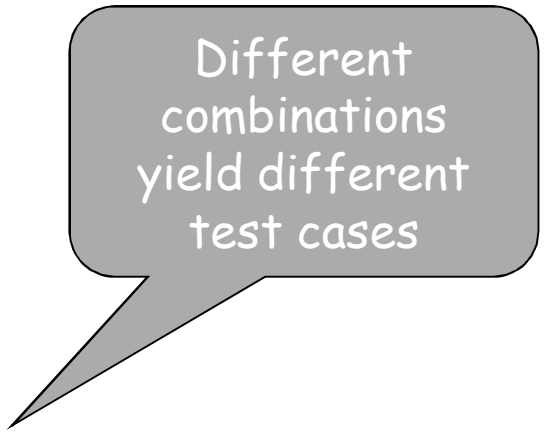
- Pros:
  - Test the system as a whole
  - Test complex and realistic scenarios
- Cons:
  - Single function failures makes the test inefficient
  - Hard to achieve good coverage

Function Testing

- Black box unit testing
- Test each function thoroughly, one at a time

Reading Post Test Cases

login(r)	Success
login(r)	Fail
getMyReader()	r
logout()	Success
getMyReader()	null
login(r)	Success



Function Testing

- Pros:
  - Thorough analysis of each item tested
- Cons:
  - Misses units interaction
  - Require additional integration testing

Domain Testing

- Equivalence partitioning subdivides the world into classes
- A group of test cases form an equivalence class if:
  - One reveals a fault iff the other ones will too (probably)
- Discover best representatives of the classes

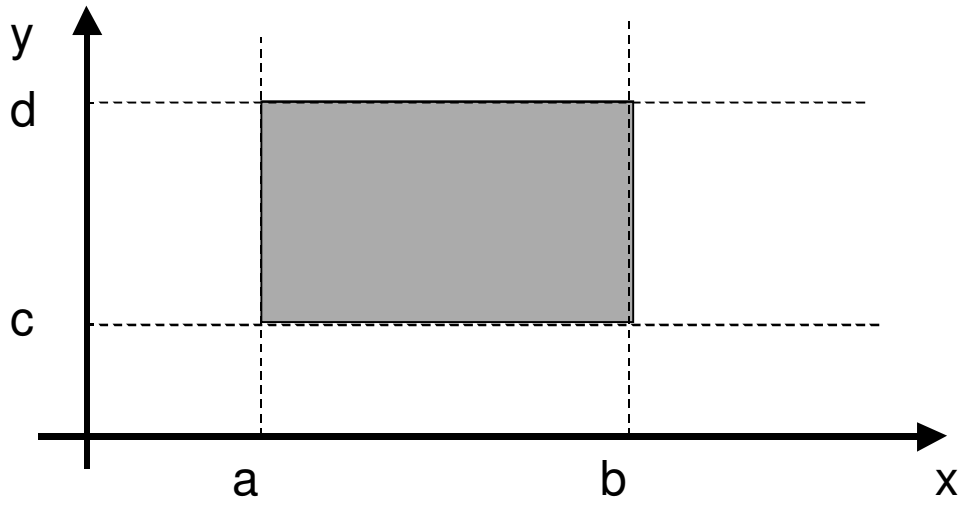
Domain Testing: Boundary Analysis

- Boundaries mark the point of transition from one equivalence class to another
- The program is more likely to fail at a boundary, so these are good representatives of the classes
  - Choose one (or more) arbitrary value(s) in each equivalence class
  - Choose valid values on lower and upper boundaries
  - Choose invalid values immediately below and above each boundary (if applicable)
- Choose inputs that invoke output boundary values

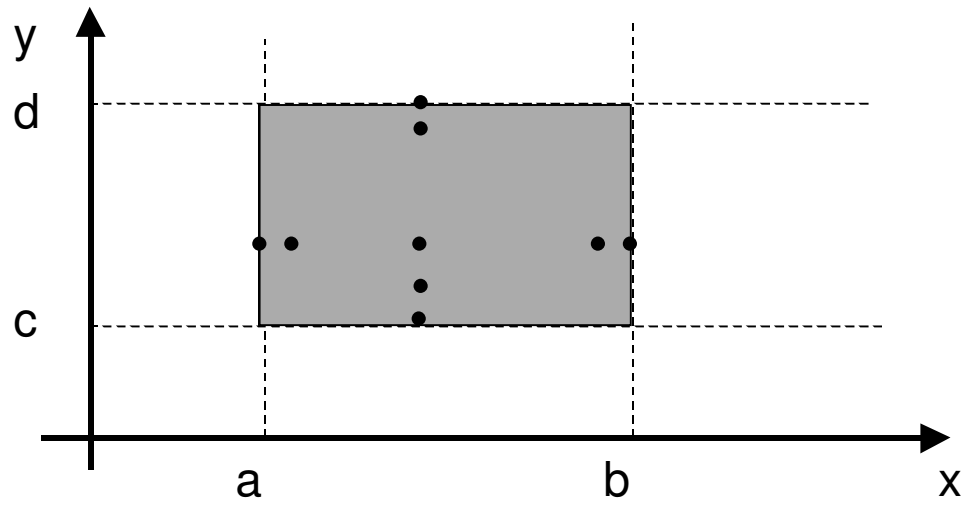
Domain Testing: Example

- Consider the following function:

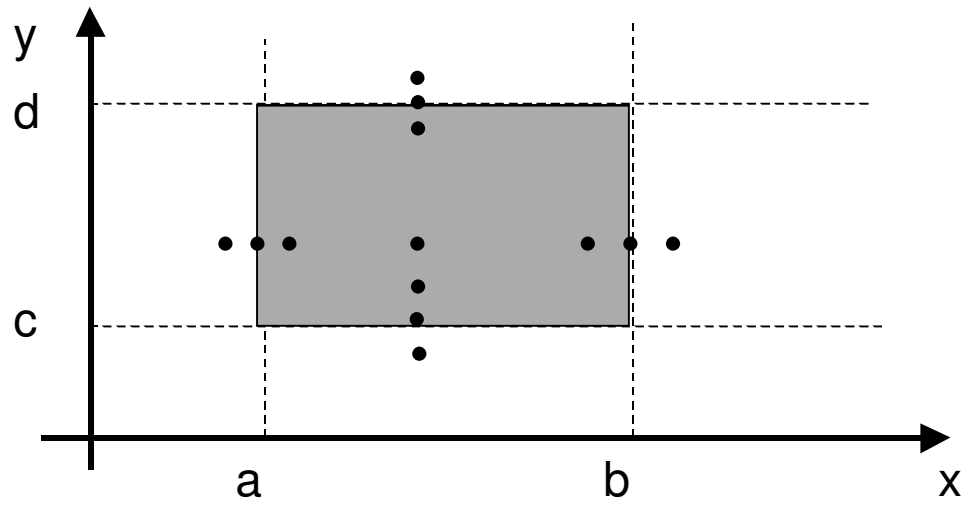
  f(x,y), where a≤x≤b, c≤y≤d



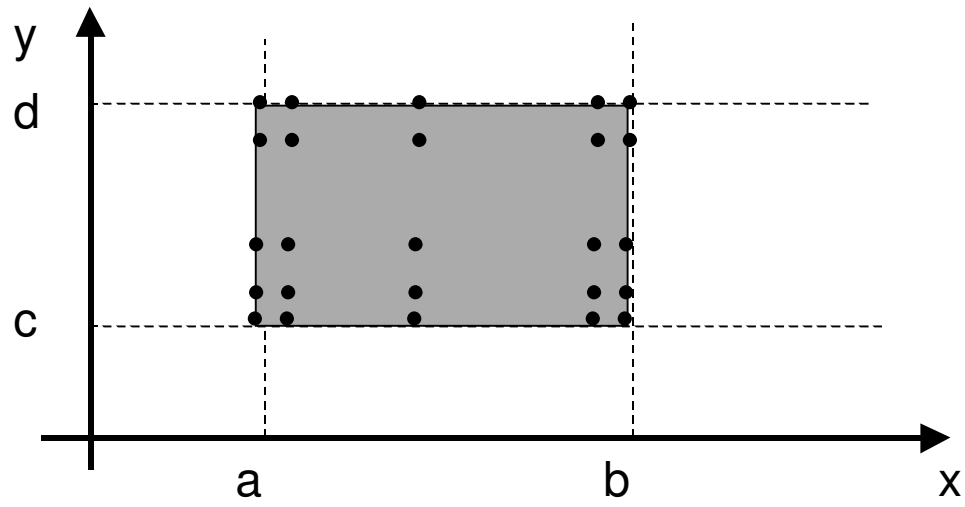
Boundary Value Analysis



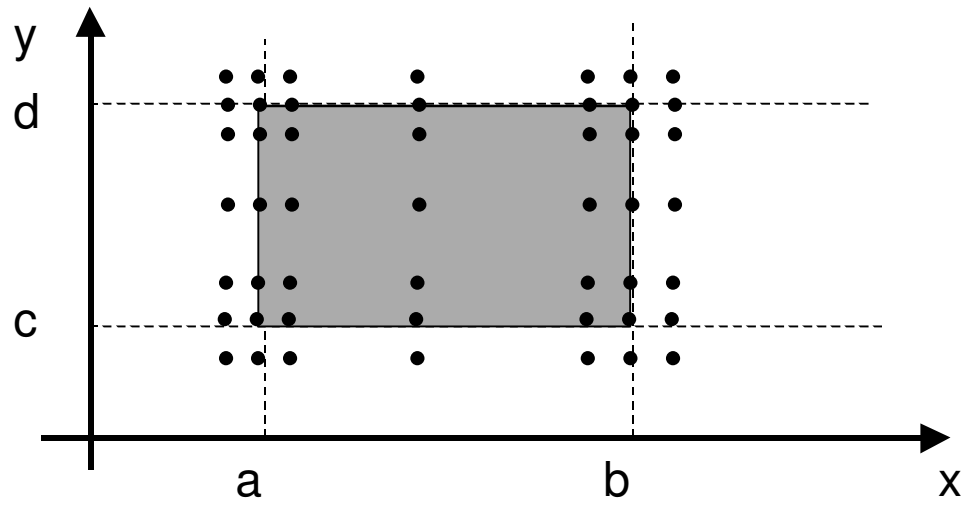
Robustness Test Cases



Worst Case Testing



Robust Worst Case Testing



Domain Testing

- Pros:
  - Intuitively clear approach for numeric features
  - Find highest probability errors with a relatively small set of tests
- Cons:
  - The actual domains are often unknowable
  - Trying to combine more than one feature complicates things

Domain Testing: A Broad Concept

The notion of equivalence class is much broader than numeric ranges

- Membership in a common group
  - employees vs. non-employees
- Equivalent hardware
  - groups of printers
- Equivalent event times
  - before-timeout and after
- Equivalent operating environments
  - French & English versions of Windows

Test plan

- Describe the strategy for testing
  - Type of the tests
  - Schedule, distribution
  - Measures for completing the tests
- Describe the test environment
  - Specific constructed for the purpose of testing
- Test procedure: enlisting relevant test cases
  - Derived from the use cases
  - Derived from the requirements

Test Procedure

Req	Verify that	Test description	Expected result
9	The administrator can remove rooms	1. Initiate the system with a room list 2. The administrator selects a room 3. The administrator removes the room 4. The administrator selects the room	The room does not appear in the selection
9	Only the administrator can remove rooms	…	Error message indicating the user cannot remove a room

Extended Test Procedure Table

Test procedure				Test Result report	
Req	Verify that	Test description	Expected result	Pass/Fail	Failure analysis
9	…	…	…	P	
9	…	…	…	F	The button was not disabled