# Testing
# a Routine, Module or Program

SFWR ENG 2B03

2003

Robert L. Baber

# Test What?

Object of the test:

- a routine, module, or program
- against its specification

The *Design Report* is the main basis and object of the test,
but other documentation may also be used.

Errors discovered in the inspection and review should be corrected before testing, if possible.

# Goal of Testing Software

- identify errors (in the software, in interfaces, in assumptions about the environment or in the specification)

Testing can show the presence of errors, but not the absence of errors. [Dijkstra] Testing can sometimes show the absence of errors, but only in special and uncommon circumstances.

When testing, assume that the program contains errors. Select test cases to demonstrate them.

# Testing Procedure

1. Select test cases
- types of testing, criteria for selecting test cases
2. Run test cases
- test bed for controlling inputs, recording outputs
3. Check individual results
- manually, mechanized, compare with expected results, evaluate against specification
4. Analyze collection of results for error patterns

# Testing Procedure

After changing (e.g. correcting) a program, *all* previous test cases must be run again. New test cases may be added to the collection of test cases.

Therefore, save old test cases and mechanize (automate) the testing process.

Running the previous collection of test cases again is called “regression testing”.

# Types of Testing

- Black box testing
- Gray box testing
- White (clear) box testing
- Statistical testing (“statistical random”)
- Random testing (“wild random”)

# Black Box Testing

Test cases chosen by looking at

- specification only (MIS)
- *not* at data structures
- *not* at code

# Gray Box Testing

Test cases chosen by looking at

- specification (MIS)
- data structures (e.g. concrete state variables, MID)
- but *not* at source code

# White (Clear) Box Testing

Test cases chosen by looking at

- specification (MIS)
- data structures (e.g. concrete state variables, MID)

and

- source code
- any available relevant document

# Statistical Testing

Test cases chosen at random

- selected from “operational profile”
- i.e. from same probability distribution as input data in actual operation

“Operational profile” problematic

Example: “beta testing”

# Random Testing (“Wild Random”)

Test cases chosen at random

- selected from any convenient probability distribution, e.g. uniform distribution
- advantage: includes extreme cases overlooked by other methods, easy to generate, avoids subconscious, preconceived patterns
- disadvantage: can generate invalid or operationally impossible cases leading to difficult or unproductive analysis

# Test Strategy

- Use all feasible types of testing (black box, gray box, white box, statistical, random).
- Define clearly, explicitly and systematically your criteria for selecting test cases.
- Identify test cases others would overlook (be imaginative).
- Select typical cases for test.
- Select extreme, atypical, degenerate test cases. (They arise in practice, too.)

# Test Strategy

- Select erroneous cases.
- Try very large numbers.
- Try very small numbers.
- Try numbers that are very close to each other.
- Think of cases that *nobody* thinks of.
- Include ridiculous cases, ridiculous combinations of values.
- Include "impossible", "very rare" cases.

# Test Strategy

With white box testing, select test cases to

- execute every statement in code
- take every if-branch, every loop condition branch
- execute every path through code (often very large number of paths, especially for program with loops, often infeasible)

# Run Test Cases

Create test bed

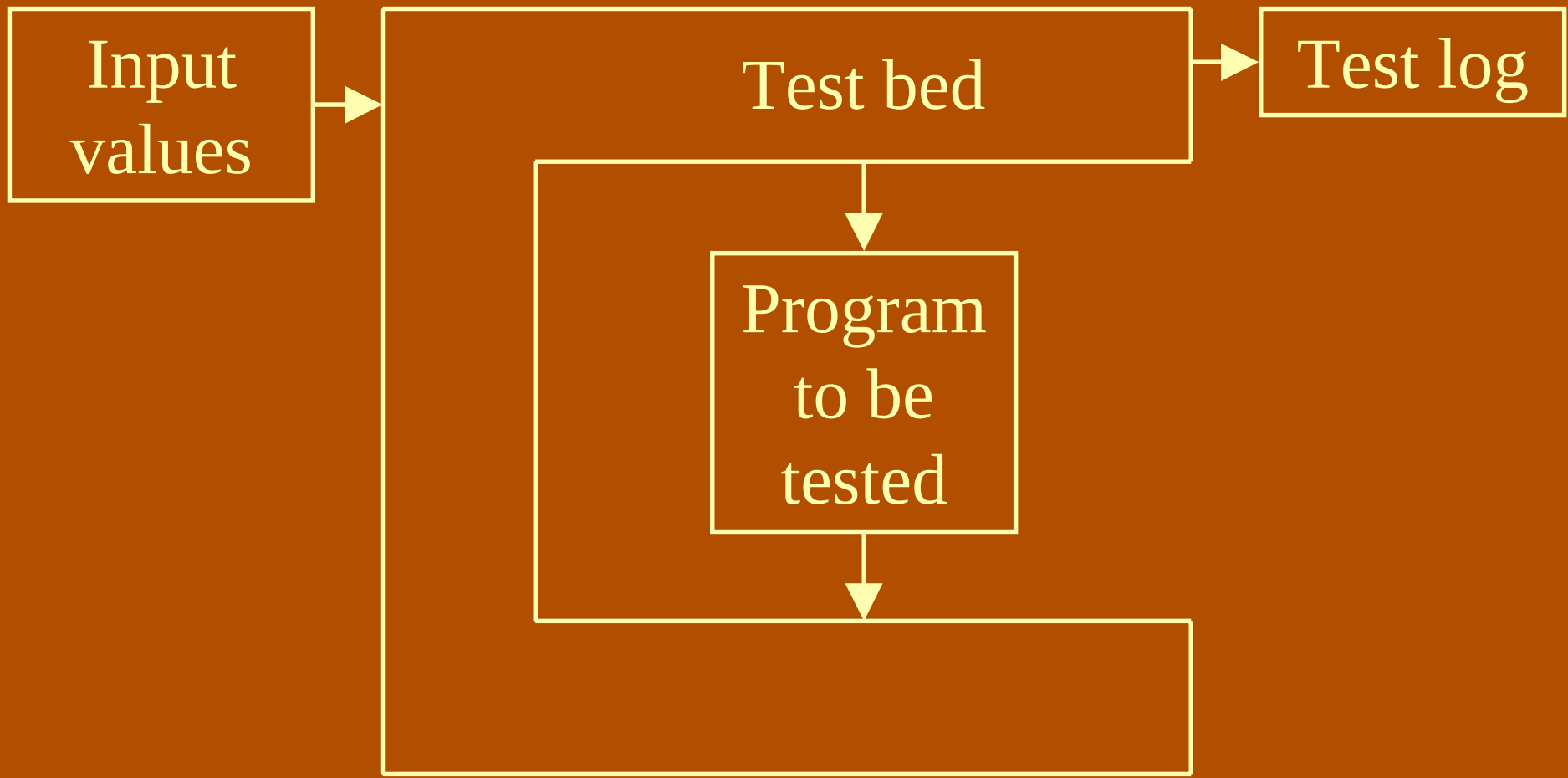
- control values of input parameters and state variables before running each test case
- control values of results of executing subsidiary procedures not yet available
- record values of output parameters and state variables after running each test case

# Run Test Cases

Test bed

- test case input values: manual input or input from a file
- include expected values of results in input file if appropriate
- test log file: record each test case run and the results

# Test Bed



# Check Individual Results

- manually and visually: time consuming
- compare with expected results:
  - calculate expected results before test run
  - use verified results from previous runs
  - but: valid results not always unique
- evaluate result against specification
  - specification is a Boolean function, evaluate it; true: result correct, else: result wrong

# Analyze Collection of Results

Patterns of errors

- common lines, segments, paths in program?
- common values of input variables?
- common values of state variables?

Malfunction rate (fraction of wrong results)

- meaningful only for statistical testing

Your own criteria for analyzing results

# Test Report

Your Test Report must contain at least

- how you generated your test cases
- your rationale and criteria for generating or selecting test cases of each type (black box, gray box, white box, statistical, random)
- how many test cases you ran
- results of running your test cases (errors identified, analysis)

# Test Report

Your Test Report must show, for each test case:

- basis (white, gray or black box)
- criteria for selection (typical or extreme values, combination of values, path segment X, decision branch Y, etc.)
- path/path segments traversed
- type of data state tested (e.g. cannot qualify for reward, must accept punishment, etc.)

# Test Report

In your Test Report include appendices:

- your test cases with actual results and whether or not correct
- correctly handled test cases
- incorrectly handled test cases

# Summary

- goal of testing: demonstrate the presence of errors
- as many as possible
- many types of testing
- many strategies for testing
- mechanize your testing

Assume that the program contains errors and do your best to identify them all. Be *very suspicious*.

# Summary

- If your test finds **no errors** in the program, your test has

failed.

- A successful test finds errors. The more the errors, the more successful the test.

# References

See

- 2A04 literature
- the vast literature on testing in books, scientific research journals and the software development trade literature