

# *Black Box Software Testing*

## *(Professional Seminar)*

**Cem Kaner, J.D., Ph.D.**

Professor of Computer Sciences  
Florida Institute of Technology

**Section:8**

### **Introduction to Black Box Test Case design**

Summer, 2002

Contact Information:

kaner@kaner.com

[www.kaner.com](http://www.kaner.com) (testing website)

[www.badsoftware.com](http://www.badsoftware.com) (legal website)

I grant permission to make digital or hard copies of this work for personal or classroom use, with or without fee, provided that (a) copies are not made or distributed for profit or commercial advantage, (b) copies bear this notice and full citation on the first page, and if you distribute the work in portions, the notice and citation must appear on the first page of each portion, (c) each page bear the notice "Copyright (c) Cem Kaner" or if you changed the page, "Adapted from Notes Provided by Cem Kaner". Abstracting with credit is permitted. The proper citation for this work is Cem Kaner, *A Course in Black Box Software Testing (Professional Version)*, Summer-2002, [www.testing-education.org](http://www.testing-education.org). To copy otherwise, to republish or post on servers, or to distribute to lists requires prior specific permission and a fee. Request permission to republish from kaner@kaner.com.

# *Test Case Design*

---

**We'll soon see a wide variety of strategies for designing test cases. Each one offers its own definition of excellence in test case design.**

**This section summarizes some traditional wisdom about good test case design.**

# *Test Case Design*

**If the purpose of testing is to gain information about the product, then a test case's function is to elicit information quickly and efficiently.**

**In information theory, we define “information” in terms of reduction of uncertainty. If there is little uncertainty, there is little information to be gained.**

**A test case that promises no information is poorly designed. A good test case will provide information of value whether the program passes the test or fails it.**

# *Models*

## **A Model is...**

- A map of a territory
- A simplified perspective
- A relationship of ideas
- An incomplete representation of reality
- A diagram, list, outline, matrix...

**No good test design has ever been done without models.**

**The trick is to become aware of how you model the product, and learn different ways of modeling.**

# *Good Test Case Design*

**An excellent test case satisfies the following criteria:**

- Reasonable probability of catching an error
- Exercises an area of interest
- Does interesting things
- Doesn't do unnecessary things
- Neither too simple nor too complex
- Not redundant with other tests
- Makes failures obvious
- Allows isolation and identification of errors

# *Good Test Case Design: Neither Too Simple Nor Too Complex*

- What makes test cases simple or complex? (*A simple test manipulates one variable at a time.*)
- Advantages of simplicity?
- Advantages of complexity?
- Transition from simple cases to complex cases (*You should increase the power and complexity of tests over time.*)
- Automation tools can bias your development toward overly simple or complex tests

Refer to Testing Computer Software, pages 125, 241, 289, 433

# *Good Test Case Design: Make Program Failures Obvious*

***Important failures have been missed because they weren't noticed after they were found.***

Some common strategies:

- **Show expected results.**
- **Only print failures.**
- **Log failures to a separate file.**
- **Keep the output simple and well formatted.**
- **Automate comparison against known good output.**

Refer to Testing Computer Software, pages 125, 160, 161-164

# *Make Test Output Obvious: Examples from Printer Testing*

**Normal.** *Italics.* **Bold.** *Bold Italics.* **Normal.** *Italics.* **Bold.** *Bold Italics.*

**Normal.** *Italics.* **Bold.** *Bold Italics.* **Normal.** *Italics.* **Bold.** *Bold Italics.*

**Normal.** *Italics.* **Bold.** *Bold Italics.* **Normal.** *Italics.* **Bold.** *Bold Italics.*

**Normal.** *Italics.* **Bold.** *Bold Italics.* **Normal.** *Italics.* **Bold.** *Bold Italics.*

*Frame graphical output*

**Put a one-pixel wide frame around the graphic and/or around the edge of the printable page. Off-by-one-pixel errors will often erase one of these borders.**

*For color output, create color sample charts.*

**Print a color wheel or some other standard progression (light to dark or rainbow progression) that is visually obvious. Name the colors. Have a comparison page handy.**

*Draw regular line-art that can show distortions*

**Draw circles and perfect squares. Look for bowing, stretching, dropout. Use some dashed lines, to see if dashes are drawn unequally long.**

# *Making a Good Test*

- Start with a known state
- Design variation into the tests
  - » configuration variables
  - » specifiable (e.g. table-loadable) data values
- Check for errors
- Put your analysis into the test itself
- Capture information when the error is found (not later)
  - » test results
  - » environment results
- Don't encourage error cascades

