

Black Box Software Testing

(Professional Seminar)

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

Professor of Computer Sciences
Florida Institute of Technology

Section:22

Introduction to Test Documentation

Summer, 2002

Contact Information:

kaner@kaner.com

www.kaner.com (testing website)

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. 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.

Black Box Software Testing

Introduction to Test Documentation

Co-author: James Bach

Acknowledgement

Some of this material is from the Third Los Altos Workshop on Software Testing (LAWST), February 7 and 8, 1998. I founded the LAWST and was co-organizer of LAWST 3.

At LAWST, we discussed test documentation (test planning strategies and materials). The agenda item was:

- » How do we know what test cases we have? How do we know which areas of the program are well covered and which are not?
- » How do we develop this documentation EFFICIENTLY? As many of you know, I despise thick test plans and I begrudge every millisecond that I spend on test case documentation. Unfortunately, some work is necessary. My question is, how little can we get away with, while still preserving the value of our asset?

The following people attended LAWST 3: Chris Agruss, James Bach, Karla Fisher, David Gelperin, Kenneth Groder, Elisabeth Hendrickson, Doug Hoffman, III, Bob Johnson, Cem Kaner, Brian Lawrence, Brian Marick, Thanga Meenakshi, Noel Nyman, Jeffery E. Payne, Bret Pettichord, Johanna Rothman, Jane Stepak, Melora Svoboda, Jeremy White, and Rodney Wilson.

Definitions

The test planning documents include:

- **The *Testing Project Plan*, which identifies classes of tasks and broadly allocates people and resources to them;**
- **High-level designs for *test cases* (individual tests) and *test suites* (collections of related tests);**
- **Detailed lists or descriptions of test cases;**
- **Anything else that you would put in a hard copy or virtual binder that documents the collection of test cases that you will develop and run.**

This collection of materials is sometimes called the *test plan* and sometimes called the *test documentation*.

Levels of Analysis

1-Variable

- Basic user interface verification
- Filters
- *Many testers stop here, at the most superficial level of testing*

2-Variable

- Shared or related filters
- Logical relationships among variables

Multi-Variable

- The core problem is the vastness of the space of test cases. We need a sampling strategy.

Basic Test Documentation Components

Lists:

- Such as lists of fields, error messages, DLLs

Outlines: An outline organizes information into a hierarchy of lists and sublists

- Such as the testing objectives list later in the course notes

Tables: A table organizes information in two dimensions showing relationships between variables.

- Such as boundary tables, decision tables, combination test tables

Matrices: A matrix is a special type of table used for data collection.

- Such as the numeric input field matrix, configuration matrices

Refer to Testing Computer Software, pages 217-241. For more examples, see page Testing Computer Software, page 218.

Configuration Planning Table

	Var 1	Var 2	Var 3	Var 4	Var 5
Config 1	V1-1	V2-1	V3-1	V4-1	V5-1
Config 2	V1-2	V2-2	V3-2	V4-2	V5-2
Config 3	V1-3	V2-3	V3-3	V4-3	V5-3
Config 4	V1-4	V2-4	V3-4	V4-4	V5-4
Config 5	V1-5	V2-5	V3-5	V4-5	V5-5
Config 6	V1-6	V2-6	V3-6	V4-6	V5-6

This table defines 6 standard configurations for testing. In later tests, the lab will set up a Config-1 system, a Config-2 system, etc., and will do its compatibility testing on these systems. The variables might be software or hardware choices. For example, Var 1 could be the operating system (V1-1 is Win 2000, V1-2 is Win ME, etc.) Var 2 could be how much RAM on the computer under test (V2-1 is 128 meg, V2-2 is 32 meg., etc.). Var 3 could be the type of printer, Var 4 the machine's language setting (French, German, etc.). Config planning tables are often filled in using the All Pairs algorithm.

Configuration Test Matrix

	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6
Test 1	Pass	Pass	Pass	Pass	Pass	
Test 2		Fail	Pass	Pass	Pass	
Test 3	Pass	Pass	Pass	Pass	Pass	
Test 4	Pass	Fail	Fail		Pass	
Test 5	Fail	Pass	Fail	Pass	Pass	

This matrix records the results of a series of tests against the 6 standard configurations that we defined in the Configuration Planning Table.

In this table, Config 1 has passed 3 tests, failed 1, and hasn't yet been tested with Test 2. Config 6 is still untested.

