

Black Box Software Testing

(Academic Course - Fall 2001)

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

Florida Institute of Technology

Section: 21 :

Scenario Testing

Contact Information:

kaner@kaner.com

www.kaner.com (testing practitioners)

www.badsoftware.com (software law)

www.testingeducation.org (education research)

Copyright (c) Cem Kaner 2001.

I grant permission to make digital or hard copies of this work for personal or classroom use, 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. Abstracting with credit is permitted. The proper citation for this work is Cem Kaner, *A Course in Black Box Software Testing (Academic Version)*, Fall 2001, 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.

Copyright (c) Cem Kaner 2001

Scenario Testing

- **Tag lines**
 - “Do something useful and interesting”
 - “Do one thing after another.”
- **Fundamental question or goal**
 - Challenging cases that reflect real use.
- **Paradigmatic case(s)**
 - Appraise product against business rules, customer data, competitors’ output
 - Life history testing (Hans Buwalda’s “soap opera testing.”)
 - Use cases are a simpler form, often derived from product capabilities and user model rather than from naturalistic observation of systems of this kind.

Scenario Testing

- **The ideal scenario has several characteristics:**
 - It is realistic (e.g. it comes from actual customer or competitor situations).
 - There is no ambiguity about whether a test passed or failed.
 - The test is complex, that is, it uses several features and functions.
 - There is a stakeholder who has influence and will protest if the program doesn't pass this scenario.
- **Strengths**
 - Complex, realistic events. Can handle (help with) situations that are too complex to model.
 - Exposes failures that occur (develop) over time
- **Blind spots**
 - Single function failures can make this test inefficient.
 - Must think carefully to achieve good coverage.

Scenarios

- **Some ways to trigger thinking about scenarios:**
 - **Benefits-driven:** People want to achieve X. How will they do it, for the following X's?
 - **Sequence-driven:** People (or the system) typically does task X in an order. What are the most common orders (sequences) of subtasks in achieving X?
 - **Transaction-driven:** We are trying to complete a specific transaction, such as opening a bank account or sending a message. What are all the steps, data items, outputs and displays, etc.?
 - **Get use ideas from competing product:** Their docs, advertisements, help, etc., all suggest best or most interesting uses of their products. How would our product do these things?

Scenarios

- Some ways to trigger thinking about scenarios:
 - **Competitor's output driven:** Hey, look at these cool documents they can make. Look (think of Netscape's superb handling of often screwy HTML code) at how well they display things. How do we do with these?
 - **Customer's forms driven:** Here are the forms the customer produces in her business. How can we work with (read, fill out, display, verify, whatever) them?

Soap Operas

- Build a scenario based on real-life experience. This means client/customer experience.
- Exaggerate each aspect of it:
 - example, for each variable, substitute a more extreme value
 - example, if a scenario can include a repeating element, repeat it lots of times
 - make the environment less hospitable to the case (increase or decrease memory, printer resolution, video resolution, etc.)
- Create a real-life story that combines all of the elements into a test case narrative.
- **(Thanks to Hans Buwalda for developing this approach and patiently explaining it to me.)**

Soap Operas

- **(As these have evolved, Hans distinguishes between *normal soap operas*, which combine many issues based on user requirements—typically derived from meetings with the user community and probably don't exaggerate beyond normal use—and *killer soap operas*, which combine *and exaggerate to produce extreme cases.*)**