

Things YOU Might Not Know About

“Powering-Up” Your Existing Tests

by Dawn Haynes

Testers often come to my class looking for ways to make their testing more effective and valuable without having to make wholesale changes to their existing testing strategy. Over the years, I’ve collected ten heuristics for “powering up” tests by intelligently varying the conditions, sequence, data, or perspective of the tests during execution.

1

DÉJÀ VU. Starting from a known clean state or initial startup state, execute a test or scenario. Then, immediately repeat the same test, using the same data if possible, without cleaning up or restarting between test executions. This can expose issues with default values or states by causing the application to exhibit a different behavior or output due to subtle variables.

2

LATHER, RINSE, REPEAT . . . AND REPEAT, AND REPEAT. Repetition often mimics the real usage model of the application or software under test. Most users do several cycles of a small number of operations many times a day. Not testing realistic usage patterns can lead to unwanted surprises in production (like memory leaks).

3

THE ROAD LESS TRAVELED. Don’t always follow the instructions, procedure, steps, or obvious path—many of your users won’t. Some of the most costly defects in production that I have seen were found but dismissed as “training issues” or left unresolved with the declaration “a user would never do that.” Even if your software is only being accessed by other systems, it’s risky to assume that the other systems will always interact with yours using the same pattern or by following precisely the paths of the specifications or design.

4

I’M A GENIUS. Take the expert or super-user approach. Take shortcuts, jump around, skip steps, use command line interfaces, enter through back doors, stretch customizations, use undocumented features, or operate the interface quickly. Often systems and user interfaces are designed for the average user. Look for ways to “crank up” your usage.

5

“DOH.” When walking through obvious, common, or critical scenarios, stop, smack yourself on the forehead (gently if you are prone to headaches), then try to go back and fix something, do something you forgot to do, edit an entry, etc. Basically, throw an interrupt and then change something you did by trying to Go back, Redo, Overwrite, Undo, or change Perspective or mode of usage midstream (GROUP). If your application has a UI, especially if it is browser-based, you can count on users doing some of these things when it’s least desired.

6

THINK BIG. Specifically, think about big data. During the life of your software, it is likely to encounter large databases, large data sets, large files, large input field values, large transactions, and large volumes of transactions. Finding limits and constraints that are likely to be encountered in production, near term, or in the future can be extremely valuable in terms of planning and executing successful software implementations.

7

PUZZLE PIECES. Systems are often conceived and built in pieces that are later assembled in a manner intended to provide value to businesses, users, or customers. How do the pieces fit when the puzzle is completed? Think about stringing pieces together in ways that may or may not have been intended through design or workflow.

8

BLENDER: MIX, PULSE, FRAPPÉ, CRUSH. Blend variables, features, transactions, and usage scenarios to various degrees (mix gently vs. chaos).

9

VARY SPACE AND TIME CONTINUUMS. Operate the system very slowly (novice user, one-finger typist, Curious George, Sunday driver) or very fast (Speedy Gonzalez). Each of these taxes the system in a different way. Shift between fast and slow modes. Interrupt modes with things like the “coffee break test”—abandon an operation mid-stream without saving or exiting. When traveling through wormholes to get back and forth between the slow and fast universes, interesting things can happen.

10

NATURAL DISASTERS. Consider a few scenarios related to usage as opposed to features, functionality, and “proper” operation. Try the “shoe on the keyboard test”: Put a shoe (or hand or coffee mug) on a keyboard (preferably on the ENTER key) and observe what happens. Of course, the “spilling coffee on the keyboard” and “ripping out the power cord” tests might be a bit more costly than small unexpected events. Select these wisely.