Black Box Software Testing (Professional Seminar)

Cem Kaner, J.D., Ph.D.
Professor of Computer Sciences
Florida Institute of Technology

Section:4
The Impossibility of Complete Testing

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.

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing

If you test completely, then at the end of testing, there cannot be any undiscovered errors. This is impossible because:

1. The domain of possible inputs is too large.
2. There are too many combinations of data to test.
3. There are too many possible paths through the program to test.
4. And then there are the user interface errors, the configuration/compatibility failures, and dozens of other dimensions of analysis.

Read Testing Computer Software, p. 17 - 22

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing 1. Too Many Inputs

The domain of possible inputs is too large

- Valid inputs
  - » In *some* cases, complete testing makes sense.
    - *Doug Hoffman worked for MASPAR (the Massively Parallel computer, 65K parallel processors). This machine is used for mission-critical and life-critical applications. To test the 32-bit integer square root function, Hoffman checked all values (all 4294967296 of them). This took the computer about 6 minutes to run the tests and compare the results to an oracle. There were 2 (two) errors, neither of them near any boundary. (The underlying error was that a bit was sometimes mis-set, but in most of these cases, there was no effect on the final calculated result.) Without an exhaustive test, these errors probably wouldn't have shown up. But what about the 64-bit integer square root? How could we find the time to run all of these?*

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
1. Too Many Inputs

The domain of possible inputs is too large

- Invalid inputs
  - » Don't forget Easter Eggs.
- Edited inputs
  - » These can be quite complex. How much editing is enough?
- Variations on input timing
  - » Try testing one event just before, just after, and in the middle of processing a second event. Will they interfere with each other?

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
2. Too Many Combinations

Variables interact.

- For example, a program crashes when attempting to print preview a high resolution (say, 600x600 dpi) image on a high resolution screen. The option selections for printer resolution and screen resolution are interacting.
- For example, a program fails when the sum of a series of variables is too large.

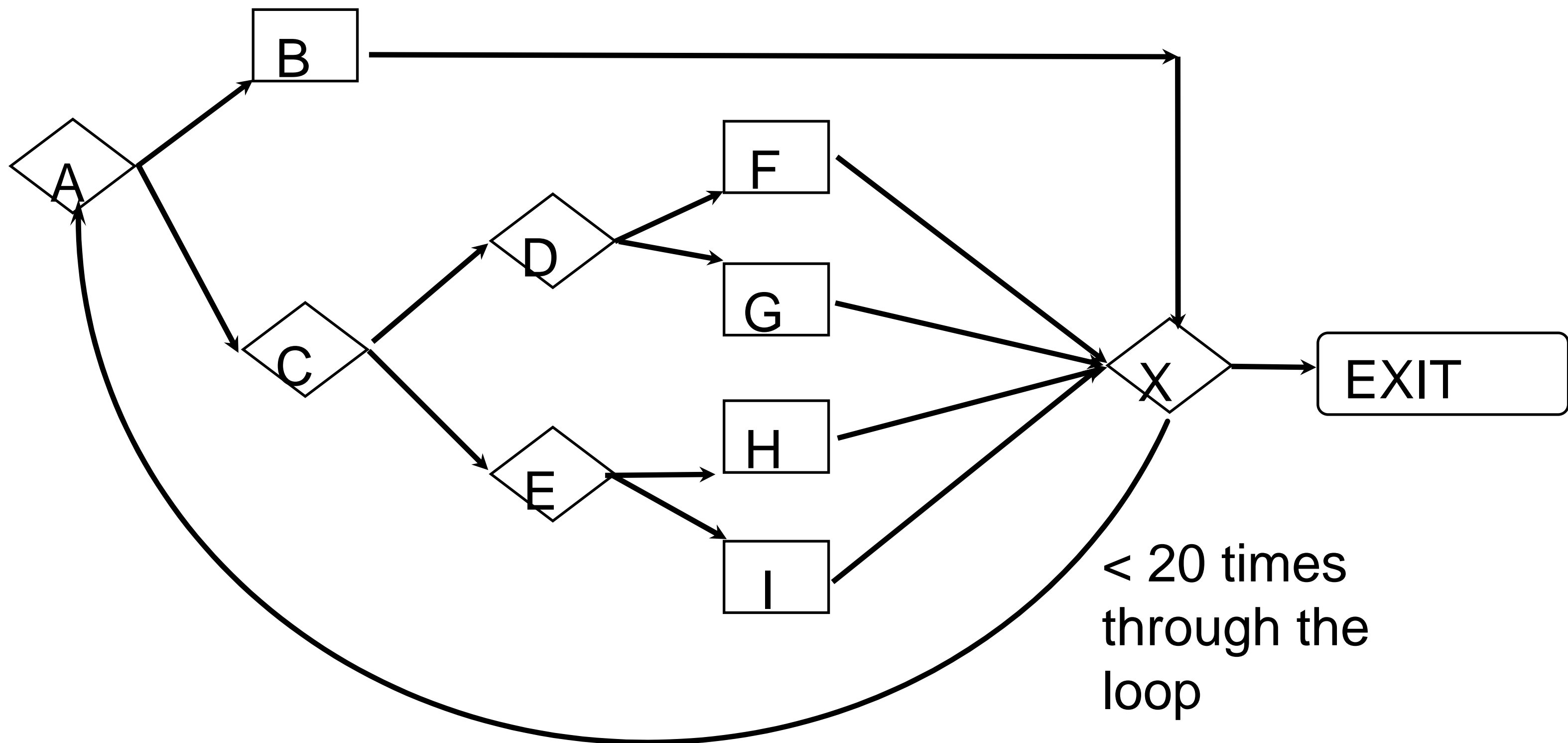
Suppose there are N variables. Suppose the number of choices for the variables are V_1, V_2, through V_N. The total number of possible combinations is $V_1 \times V_2 \times \cdots \times V_N$. This is huge.

We saw 39,601 combinations of just two variables whose values could range only between -99 and 99.

Here's a case that isn't so trivial. There are 318,979,564,000 possible combinations of the first four moves in chess.

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
3. Too Many Paths



Here's an example that shows that there are too many paths to test in even a fairly simple program. This one is from Myers, *The Art of Software Testing.*

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
3. Too Many Paths

The program starts at A.

From A it can go to B or C

From B it goes to X

From C it can go to D or E

From D it can go to F or G

From F or from G it goes to X

From E it can go to H or I

From H or from I it goes to X

From X the program can go to EXIT or back to A. It can go back to A no more than 19 times.

One path is **ABX-Exit**. There are 5 ways to get to X and then to the EXIT in one pass.

Another path is **ABXACDFX-Exit**. There are 5 ways to get to X the first time, 5 more to get back to X the second time, so there are 5 x 5 = 25 cases like this.

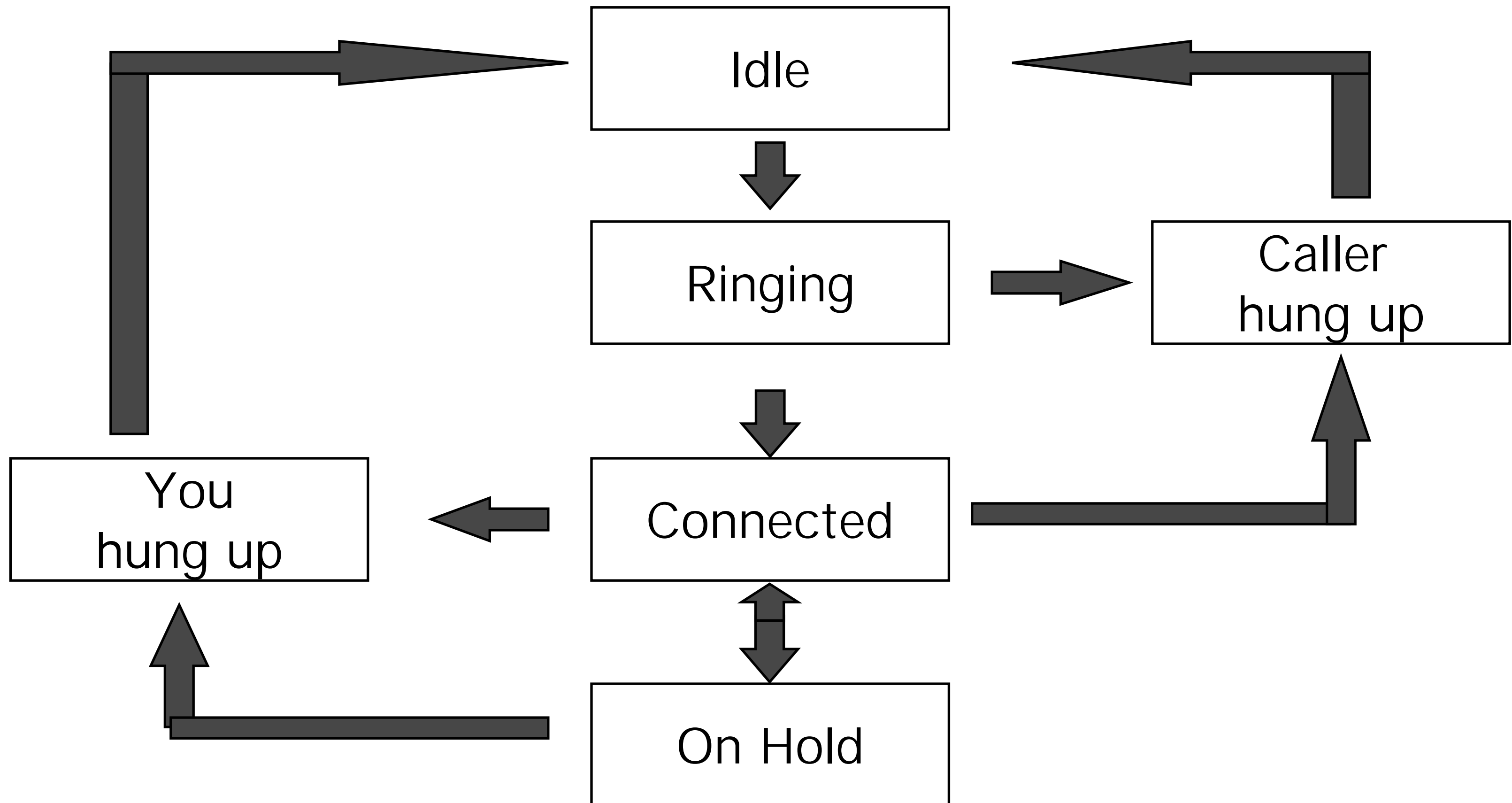
Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
3. Too Many Paths

There are $5^1 + 5^2 + ... + 5^{19} + 5^{20} = 10^{14}$ = 100 trillion paths through the program to test or approximately one billion years to try every path (if one could write, execute and verify a test case every five minutes).

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
3. Too Many Paths



Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

3. Too Many Paths--The Telephone Example

Why are we spending so much time on this crazy example?

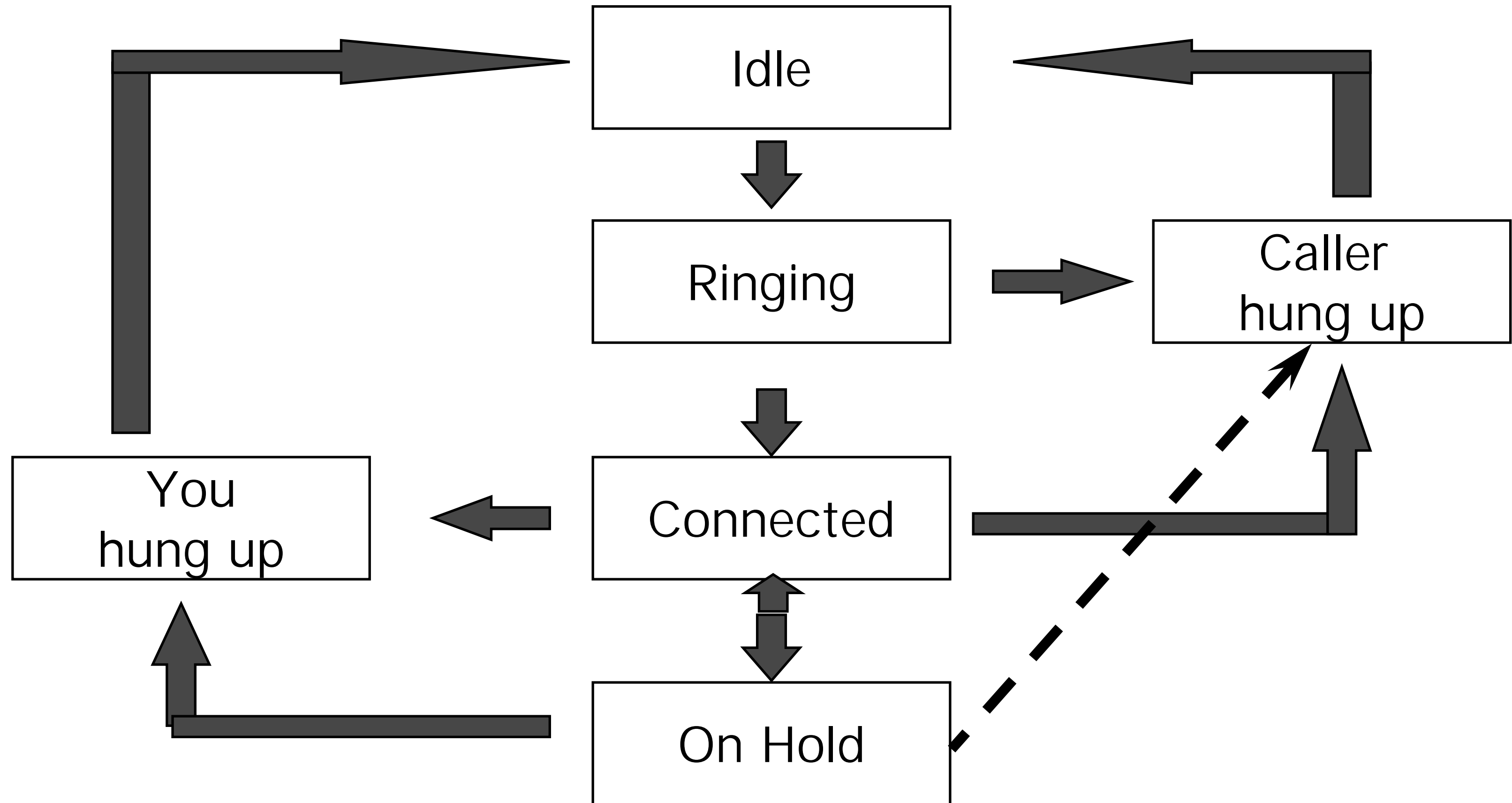
Because it illustrates several important points:

- Simplistic approaches to path testing can miss critical defects.
- Critical defects can arise under circumstances that appear (in a test lab) so specialized that you would never intentionally test for them.
- When (in some future course or book) you hear a new methodology for combination testing or path testing, I want you to test it against this defect. ***If you have no suspicion that there is a stack corruption problem in this program, will the new method lead you to find this bug?***

This example also lays a foundation for our introduction to random/statistical testing.

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

The Impossibility of Complete Testing
3. Too Many Paths



Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.

Notes

Copyright (c) 1994-2001 Cem Kaner. All Rights Reserved.