CS1632: Lecture 18

Pairwise and Combinatorial Testing

Bill Laboon/Dustin Iser

Let’s test a word processor

10 font effects

- Italic
- Bold
- Underline
- Strikethrough
- Superscript
- Shadow
- Embossed
- 3D
- Outline
- Inverse

Fonts may be combined

Plain text

Superscript

Bold

~~*Italic and strikethrough*~~

Bold and underlined

~~***Bold italic strikethrough superscript***~~

Comprehensive testing

You would need to execute 2^{10} tests to comprehensively test all the possible font combinations.

1,024 tests

Comprehensive testing is not necessary

Source: “Practival Combinatorial Testing”, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-142.pdf

- Think of each font effect as a boolean variable. For example, bold vs. not bold, italic vs. not italic, etc.
- Most (50%-90%) defects come from combinations of one or two interactions.
- In other words, most defects would be found if you just tested bold 3D text (two interactions) or just bold text (one interaction).
- The maximum number of interactions found to cause a defect was six.

Combinatorial testing

By carefully selecting test cases, we can provide a reasonable assurance of quality with a subset of the tests required for comprehensive testing.

Pairwise testing (t = 2)

A type of combinatorial testing in which the tester tests all possible pairs of interactions.

Test #	Bold	Italic	Underlined
1	TRUE	TRUE	FALSE
2	TRUE	FALSE	TRUE
3	FALSE	TRUE	TRUE
4	FALSE	FALSE	FALSE

What about pairwise testing all 10 font effects?

It was 1,024 tests to comprehensively test.

How many tests would it require to test all pairs of interactions?

https://github.com/asphaltpanthers/IS2545/blob/master/examples/Lecture16/FontEffectsPairwise-output.xls

Rule of nines

Pairwise testing finds 90% of bugs.

Three way 99% of bugs.

Four way 99.9% of bugs.

Five way 99.99% of bugs

Six way 99.999% of bugs.

...

Six way testing of 10 font effects (t = 6)

https://github.com/asphaltpanthers/IS2545/blob/master/examples/Lecture16/FontEffectsSixWay-output.xls

The number of tests we need to add to increase t increase exponentially.

As we increase t, our tests become less and less valuable because they find less and less defects.

Covering arrays