

# CS1699 - Lecture 17 - Pairwise & Combinatorial Testing

*\* ACTS and NIST imagery used with permission*

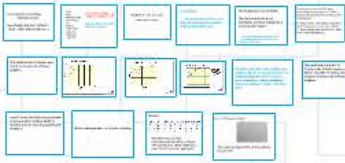
**But How Do We Make Those Tables?**



**Combinatorial Testing**




**The Problem**



**Real-World Uses**



**Benefits and Drawbacks**



# CS1699 - Lecture 17 - Pairwise & Combinatorial Testing

\* ACTS and NIST imagery used with permission

### But How Do We Make Those Tables?

This diagram illustrates the process of creating test tables from requirements. It shows a flow from 'Requirements' to 'Test Cases' and then to 'Test Results'. The 'Requirements' box contains a list of requirements, and the 'Test Cases' box contains a list of test cases. The 'Test Results' box contains a table with columns for 'Test Case', 'Pass/Fail', and 'Comments'.

### Combinatorial Testing

This diagram illustrates the process of combinatorial testing. It shows a flow from 'Requirements' to 'Test Cases' and then to 'Test Results'. The 'Requirements' box contains a list of requirements, and the 'Test Cases' box contains a list of test cases. The 'Test Results' box contains a table with columns for 'Test Case', 'Pass/Fail', and 'Comments'.

### The Problem

This diagram illustrates the problem of combinatorial testing. It shows a flow from 'Requirements' to 'Test Cases' and then to 'Test Results'. The 'Requirements' box contains a list of requirements, and the 'Test Cases' box contains a list of test cases. The 'Test Results' box contains a table with columns for 'Test Case', 'Pass/Fail', and 'Comments'.

### Real-World Uses

This diagram illustrates the real-world uses of combinatorial testing. It shows a flow from 'Requirements' to 'Test Cases' and then to 'Test Results'. The 'Requirements' box contains a list of requirements, and the 'Test Cases' box contains a list of test cases. The 'Test Results' box contains a table with columns for 'Test Case', 'Pass/Fail', and 'Comments'.

### Benefits and Drawbacks

This diagram illustrates the benefits and drawbacks of combinatorial testing. It shows a flow from 'Requirements' to 'Test Cases' and then to 'Test Results'. The 'Requirements' box contains a list of requirements, and the 'Test Cases' box contains a list of test cases. The 'Test Results' box contains a table with columns for 'Test Case', 'Pass/Fail', and 'Comments'.

# The Problem

Let's say you're testing a word processor.

Specifically, the font "effects" - (bold, italic, superscript, etc.)

Bold  
Italic  
Strikethrough  
Underlined  
3-D  
Shadow  
Superscript  
Subscript  
Embossed  
Engraved

These can be combined, e.g.,  
"bold italic underlined text"

How many tests do we need to  
fully test this feature?

Answer:  $2^{10}$ , or 1,024

(that's quite a few tests!)

For example...

... maybe a problem only occurs  
with 3-D Underlined Shadowed  
Engraved Bold Italic text.

That's possible, but unlikely.

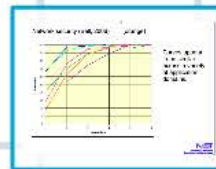
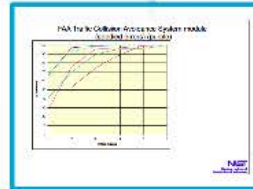
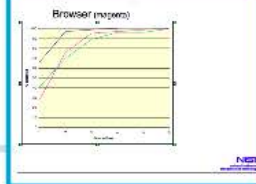
The National Institute of  
Standards and Technology did a  
study on this topic.\*

\* "Practical Combinatorial Testing", <https://csrc.nist.gov/groups/SNS/extra/documents/SP800-142-101006.pdf>

It turns out that most (50 - 90%,  
depending on the project) of defects  
come from combinations of one or two  
interactions.

In other words, most defects would be  
found if you just tested, e.g., "bold 3-  
D" (two interactions) text or just "bold  
text" (one interaction).

This distribution of defects was  
found in all sorts of software  
projects.



*The Interaction Rule: "Most failures are  
triggered by one or two parameters, and  
progressively fewer by three, four, or  
more parameters, and the maximum  
interaction degree is small." -Eric Kuhn,  
NIST*

The maximum number of  
interactions found to cause a  
defect was SIX. This was after an  
analysis of dozens of software  
projects.

Great! So we can find a large number  
of defects with minimal work by  
making sure we test all possible pairs  
of values.

This is called *pairwise*, or *all-pairs*, testing.

Example

|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|

Note that every pairwise  
combination of interactions is found  
with only 10 tests - quite a difference  
from 1,024 (2 orders of magnitude!)

So... is this good enough?



"It's cool, we found 90% of the defects!  
Hop on in!"

**Let's say you're testing a  
word processor.**

**Specifically, the font "effects" -  
(bold, italic, superscript, etc.)**

**Bold**  
**Italic**  
**Strikethrough**  
**Underlined**  
**3-D**  
**Shadow**  
**Superscript**  
**Subscript**  
**Embossed**  
**Engraved**

**These can be combined, e.g.,  
"bold italic underlined text"**

*How many tests do we need to  
fully test this feature?*

**Answer:  $2^{10}$ , or 1,024**

***(that's quite a few tests!)***



*For example...*

*... maybe a problem only occurs  
with 3-D Underlined Shadowed  
Engraved Bold Italic text.*

**That's possible, but unlikely.**

**The National Institute of Standards and Technology did a study on this topic.\***

*\* "Practical Combinatorial Testing", <http://csrc.nist.gov/groups/SNS/acts/documents/SP800-142-101006.pdf>*

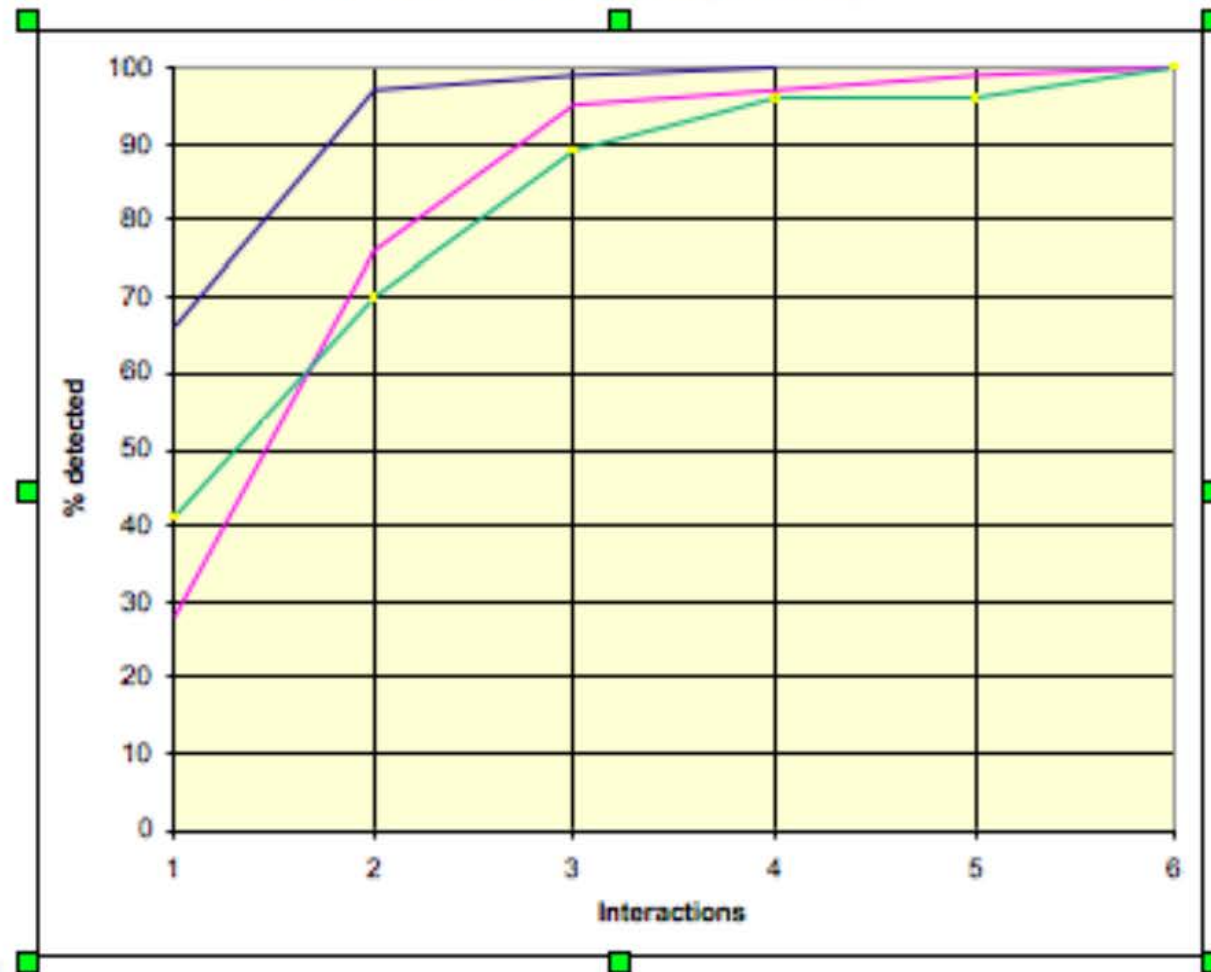


**It turns out that most (50 - 90%, depending on the project) of defects come from combinations of one or two interactions.**

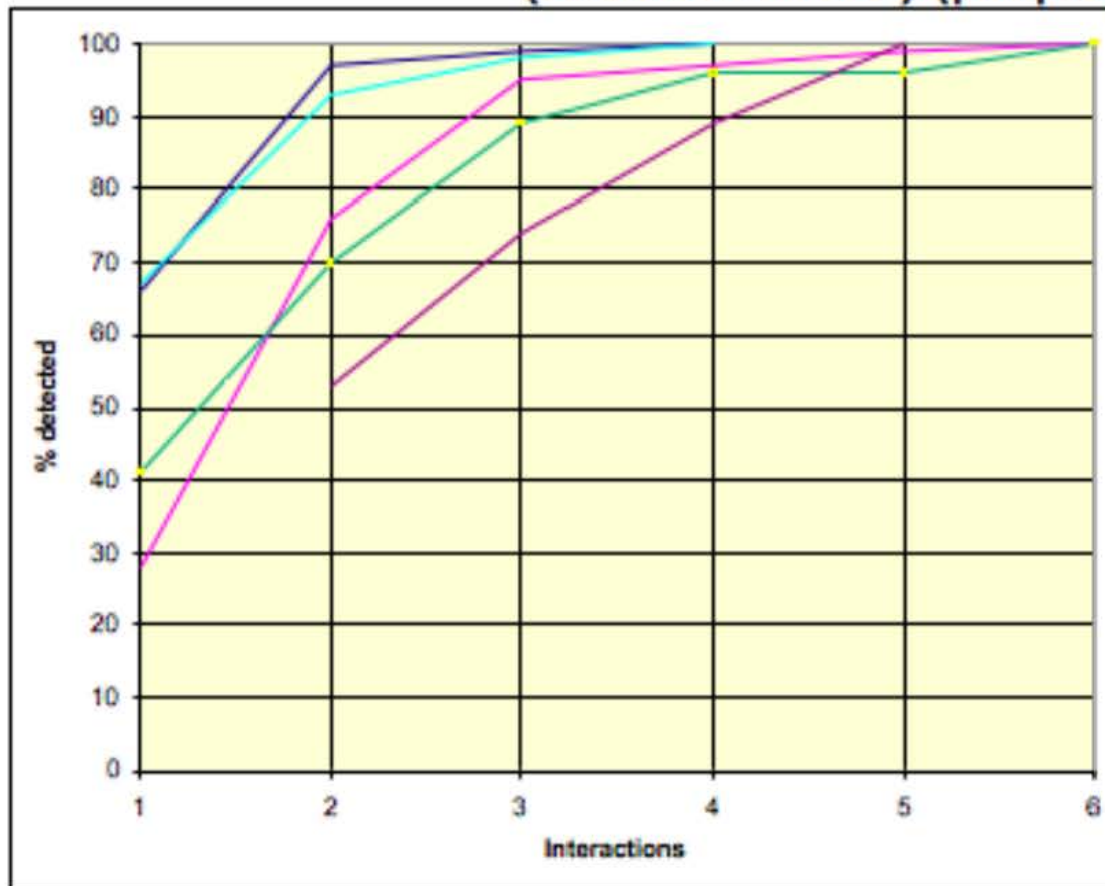
**In other words, most defects would be found if you just tested, e.g., "bold 3-D" (two interactions) text or just "bold text" (one interactions).**

**This distribution of defects was found in all sorts of software projects.**

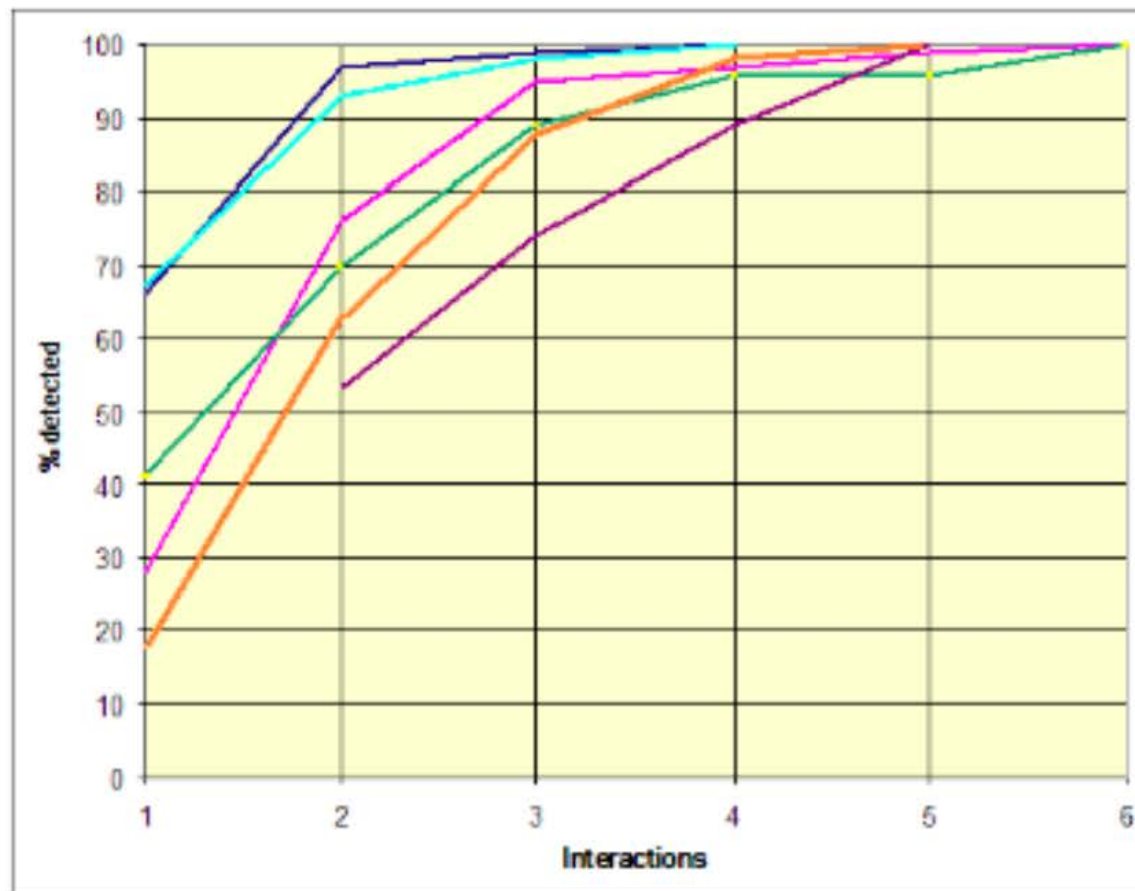
## Browser (magenta)



## FAA Traffic Collision Avoidance System module (seeded errors) (purple)



## Network security (Bell, 2006) (orange)



Curves appear to be similar across a variety of application domains.

***The Interaction Rule: "Most failures are triggered by one or two parameters, and progressively fewer by three, four, or more parameters, and the maximum interaction degree is small." -Eric Kuhn, NIST***



**The maximum number of interactions found to cause a defect was SIX. This was after an analysis of dozens of software projects.**

**Great! So we can find a large number of defects with minimal work by making sure we test all possible pairs of values.**

**This is called *pairwise*, or *all-pairs*, testing.**

# Example

|    | BOLD  | ITALIC | STRIKETHROUGH | UNDERLINE | THREED | SHADOW | SUPERSCRIPT | SUBSCRIPT | EMBOSSSED | ENGRAVED |
|----|-------|--------|---------------|-----------|--------|--------|-------------|-----------|-----------|----------|
| 1  | true  | true   | false         | false     | false  | false  | false       | false     | false     | false    |
| 2  | true  | false  | true          | true      | true   | true   | true        | true      | true      | true     |
| 3  | false | true   | true          | false     | true   | false  | true        | false     | true      | false    |
| 4  | false | false  | false         | true      | false  | true   | false       | true      | false     | true     |
| 5  | false | true   | false         | true      | true   | false  | true        | true      | false     | false    |
| 6  | false | false  | true          | false     | false  | true   | false       | false     | true      | true     |
| 7  | true  | true   | false         | false     | false  | true   | true        | true      | true      | false    |
| 8  | false | false  | true          | true      | true   | false  | false       | false     | false     | true     |
| 9  | false | true   | true          | false     | true   | false  | false       | true      | true      | true     |
| 10 | true  | false  | false         | false     | false  | false  | true        | false     | true      | false    |

***Note that every pairwise combination of interactions is found with only 10 tests - quite a difference from 1,024 (2 orders of magnitude!)***

**So... is this good enough?**



**"It's cool, we found 90% of the defects!  
Hop on in!"**

# Combinatorial Testing

OK, then, the maximum number of interactions causing a defect found in the NIST studies was six. So let's test all six-way combinations.

How many tests would that be?

Well, that was a bit of a trick question. Determining the exact number necessary is an NP-Hard problem. But there are some good algorithms out there that approximate it.

The best answer my software could come up with is 178, by the way.

Interesting, though...

- 10 tests catch 90% of defects
- 178 tests catch 99.99999999-ish of defects
- 1024 tests catch 100% of defects

... IF they are done right!

**Pareto Principle:**

\*80% of effects come from 20% of causes.\*

Examples:  
 "80% of your sales come from 20% of your customers."  
 "80% of your code execution time is in 20% of your code."  
 etc.

For a new feature, 80% of your bugs will come from 20% of your test cases.

You can see how much more expensive it becomes to test depending on how arbitrarily close to "100% free of defects" you want to be.

It is NOT a linear relationship.

These arrays we are using to make tests are called "covering arrays".

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

Still a relatively new concept - developed in the 1990s  
Very good approximation algorithms now, despite NP hardness.

Our problem earlier seemed simple, but how does this concept of covering arrays handle larger test spaces?

Pretty well, actually! Let's imagine an airplane cockpit console. There are 34 switches. Thus  $1.7 \times 10^{10}$  (17 billion) possible combinations to test.

To test all three-way interactions: 33 tests!

To test all four-way interactions: 85 tests!

Remember at the beginning of the term when I talked about the impossibility of testing every combination of inputs? This is a possible amelioration.

You could also use combinatorial testing for ordering of events:  
a, b, c, d, e, f

Pairs become "a then b", "b then a", etc.

**Note that you can also use combinatorial testing for setup configurations, e.g. operating system, web browser, CPU, etc.**

Any time when you have a large variety of configurations or inputs to test, you can use combinatorial testing to:

1. Find all combinations to capture n-way interactions
2. Maximize testing efficiency



*OK, then, the maximum number of interactions causing a defect found in the NIST studies was six. So let's test all six-way combinations.*

*How many tests would that be?*

**Well, that was a bit of a trick question. Determining the exact number necessary is an NP-Hard problem. But there are some good algorithms out there that approximate it.**

**The best answer my software  
could come up with is 178, by  
the way.**

**Interesting, though...**

**10 tests catch 90% of defects**

**178 tests catch 99.9999999%-ish of defects**

**1024 tests catch 100% of defects**

**... IF they are done right!**

## **Pareto Principle:**

**"80% of effects come from 20% of causes."**

### **Examples:**

**"80% of your sales come from 20% of your customers."**

**"80% of your code execution time is in 20% of your code."  
etc.**

**For a new feature, 80% of your bugs will come from 20% of your test cases.**





**You can see how much more expensive it becomes to test depending on how arbitrarily close to "100% free of defects" you want to be.**

**It is NOT a linear relationship.**

These arrays we are using to make tests are called "covering arrays".

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

**Still a relatively new concept - developed in the 1990s**

**Very good approximation algorithms now, despite NP hardness.**

**Our problem earlier seemed simple, but how does this concept of covering arrays handle larger test spaces ?**

**Pretty well, actually! Let's imagine an airplane cockpit console. There are 34 switches. Thus  $1.7 \times 10^{10}$  (17 billion) possible combinations to test.**

**To test all three-way interactions: 33 tests!**

**To test all four-way interactions: 85 tests!**

**Remember at the beginning of the term when I talked about the impossibility of testing every combination of inputs? This is a possible amelioration.**



**Note that you can also use combinatorial testing for setup configurations, e.g. operating system, web browser, CPU, etc.**

**Any time when you have a large variety of configurations or inputs to test, you can use combinatorial testing to:**

- 1. Find all combinations to capture n-way interactions**
- 2. Maximize testing efficiency**

# *But How Do We Make Those Tables?*

Definitely NOT by hand.

These are not artisanal, hand-crafted tables.

One possible tool - NIST ACTS (Advanced Combinatorial Testing System)

There are other ones specialized for certain domains, e.g. security access settings or web testing.

*Let's Take a Look!*

**Definitely NOT by hand.**

**These are not artisanal, hand-crafted tables.**

**One possible tool - NIST ACTS (Advanced Combinatorial Testing System)**

**There are other ones specialized for certain domains,  
e.g. security access settings or web testing.**

***Let's Take a Look!***



# Real-World Uses

Testing the DOM for the WWW Consortium



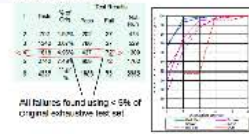
Document Object Model Events  
Original test set:

| Test Case      | Pass/Fail | Test Case        | Pass/Fail |
|----------------|-----------|------------------|-----------|
| 1. Test Case 1 | Pass      | 10. Test Case 10 | Pass      |
| 2. Test Case 2 | Pass      | 11. Test Case 11 | Pass      |
| 3. Test Case 3 | Pass      | 12. Test Case 12 | Pass      |
| 4. Test Case 4 | Pass      | 13. Test Case 13 | Pass      |
| 5. Test Case 5 | Pass      | 14. Test Case 14 | Pass      |
| 6. Test Case 6 | Pass      | 15. Test Case 15 | Pass      |
| 7. Test Case 7 | Pass      | 16. Test Case 16 | Pass      |
| 8. Test Case 8 | Pass      | 17. Test Case 17 | Pass      |
| 9. Test Case 9 | Pass      | 18. Test Case 18 | Pass      |

38,626 tests!

Exhaustive testing of equivalence class values

Combinatorial test set:



All failures found using < 5% of original exhaustive test set

Problem: unknown factors causing failures of F-16 ventral fin



Figure 2. F-16 ventral fin damage on flight with LANTIRN

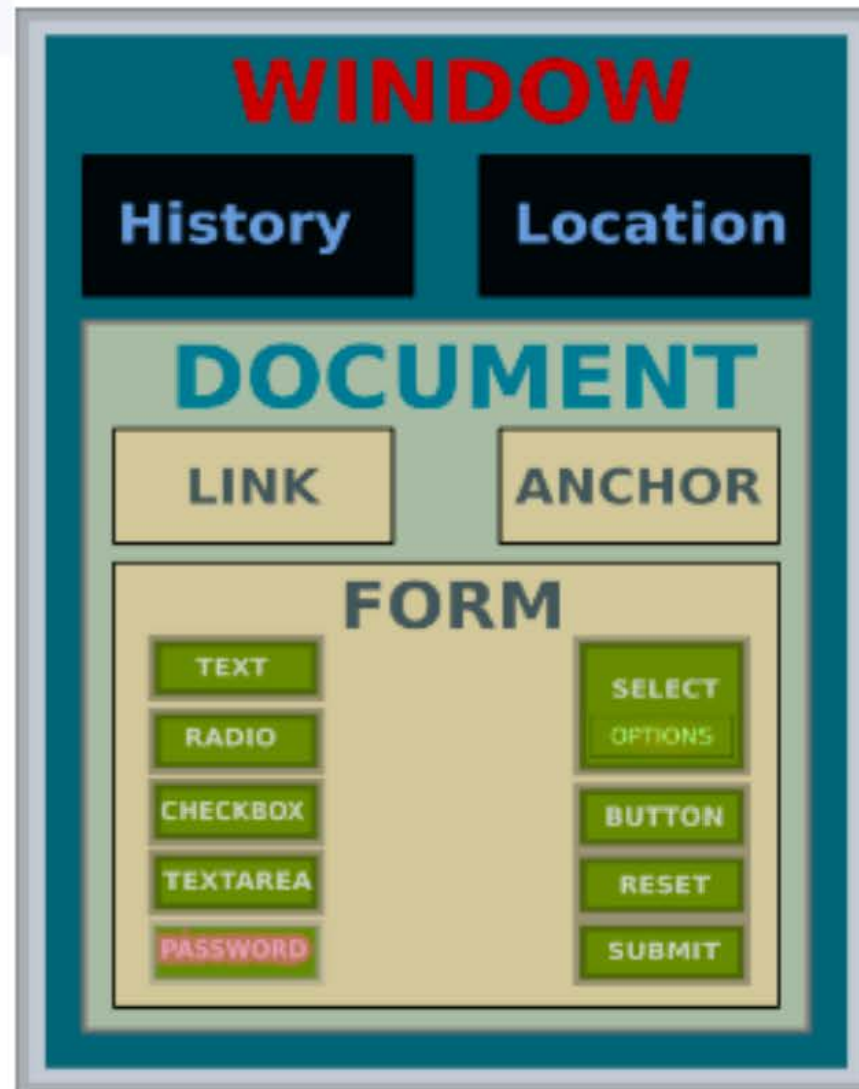
It's not supposed to look like this:



Figure 2. F-16 ventral fin damage on flight with LANTIRN

| Parameter | Value   |
|-----------|---|
| Altitude  | 15, 40  |
| Altitude  | 51, 104, 150, 204, 260, 316, 372, 428   |
| Altitude  | 484, 540, 596, 652, 708, 764, 820, 876, 932, 988, 1044, 1100, 1156, 1212, 1268, 1324, 1380, 1436, 1492, 1548, 1604, 1660, 1716, 1772, 1828, 1884, 1940, 1996, 2052, 2108, 2164, 2220, 2276, 2332, 2388, 2444, 2500, 2556, 2612, 2668, 2724, 2780, 2836, 2892, 2948, 3004, 3060, 3116, 3172, 3228, 3284, 3340, 3396, 3452, 3508, 3564, 3620, 3676, 3732, 3788, 3844, 3900, 3956, 4012, 4068, 4124, 4180, 4236, 4292, 4348, 4404, 4460, 4516, 4572, 4628, 4684, 4740, 4796, 4852, 4908, 4964, 5020, 5076, 5132, 5188, 5244, 5300, 5356, 5412, 5468, 5524, 5580, 5636, 5692, 5748, 5804, 5860, 5916, 5972, 6028, 6084, 6140, 6196, 6252, 6308, 6364, 6420, 6476, 6532, 6588, 6644, 6700, 6756, 6812, 6868, 6924, 6980, 7036, 7092, 7148, 7204, 7260, 7316, 7372, 7428, 7484, 7540, 7596, 7652, 7708, 7764, 7820, 7876, 7932, 7988, 8044, 8100, 8156, 8212, 8268, 8324, 8380, 8436, 8492, 8548, 8604, 8660, 8716, 8772, 8828, 8884, 8940, 8996, 9052, 9108, 9164, 9220, 9276, 9332, 9388, 9444, 9500, 9556, 9612, 9668, 9724, 9780, 9836, 9892, 9948, 10004, 10060, 10116, 10172, 10228, 10284, 10340, 10396, 10452, 10508, 10564, 10620, 10676, 10732, 10788, 10844, 10900, 10956, 11012, 11068, 11124, 11180, 11236, 11292, 11348, 11404, 11460, 11516, 11572, 11628, 11684, 11740, 11796, 11852, 11908, 11964, 12020, 12076, 12132, 12188, 12244, 12300, 12356, 12412, 12468, 12524, 12580, 12636, 12692, 12748, 12804, 12860, 12916, 12972, 13028, 13084, 13140, 13196, 13252, 13308, 13364, 13420, 13476, 13532, 13588, 13644, 13700, 13756, 13812, 13868, 13924, 13980, 14036, 14092, 14148, 14204, 14260, 14316, 14372, 14428, 14484, 14540, 14596, 14652, 14708, 14764, 14820, 14876, 14932, 14988, 15044, 15100, 15156, 15212, 15268, 15324, 15380, 15436, 15492, 15548, 15604, 15660, 15716, 15772, 15828, 15884, 15940, 15996, 16052, 16108, 16164, 16220, 16276, 16332, 16388, 16444, 16500, 16556, 16612, 16668, 16724, 16780, 16836, 16892, 16948, 17004, 17060, 17116, 17172, 17228, 17284, 17340, 17396, 17452, 17508, 17564, 17620, 17676, 17732, 17788, 17844, 17900, 17956, 18012, 18068, 18124, 18180, 18236, 18292, 18348, 18404, 18460, 18516, 18572, 18628, 18684, 18740, 18796, 18852, 18908, 18964, 19020, 19076, 19132, 19188, 19244, 19300, 19356, 19412, 19468, 19524, 19580, 19636, 19692, 19748, 19804, 19860, 19916, 19972, 20028, 20084, 20140, 20196, 20252, 20308, 20364, 20420, 20476, 20532, 20588, 20644, 20700, 20756, 20812, 20868, 20924, 20980, 21036, 21092, 21148, 21204, 21260, 21316, 21372, 21428, 21484, 21540, 21596, 21652, 21708, 21764, 21820, 21876, 21932, 21988, 22044, 22100, 22156, 22212, 22268, 22324, 22380, 22436, 22492, 22548, 22604, 22660, 22716, 22772, 22828, 22884, 22940, 22996, 23052, 23108, 23164, 23220, 23276, 23332, 23388, 23444, 23500, 23556, 23612, 23668, 23724, 23780, 23836, 23892, 23948, 24004, 24060, 24116, 24172, 24228, 24284, 24340, 24396, 24452, 24508, 24564, 24620, 24676, 24732, 24788, 24844, 24900, 24956, 25012, 25068, 25124, 25180, 25236, 25292, 25348, 25404, 25460, 25516, 25572, 25628, 25684, 25740, 25796, 25852, 25908, 25964, 26020, 26076, 26132, 26188, 26244, 26300, 26356, 26412, 26468, 26524, 26580, 26636, 26692, 26748, 26804, 26860, 26916, 26972, 27028, 27084, 27140, 27196, 27252, 27308, 27364, 27420, 27476, 27532, 27588, 27644, 27700, 27756, 27812, 27868, 27924, 27980, 28036, 28092, 28148, 28204, 28260, 28316, 28372, 28428, 28484, 28540, 28596, 28652, 28708, 28764, 28820, 28876, 28932, 28988, 29044, 29100, 29156, 29212, 29268, 29324, 29380, 29436, 29492, 29548, 29604, 29660, 29716, 29772, 29828, 29884, 29940, 29996, 30052, 30108, 30164, 30220, 30276, 30332, 30388, 30444, 30500, 30556, 30612, 30668, 30724, 30780, 30836, 30892, 30948, 31004, 31060, 31116, 31172, 31228, 31284, 31340, 31396, 31452, 31508, 31564, 31620, 31676, 31732, 31788, 31844, 31900, 31956, 32012, 32068, 32124, 32180, 32236, 32292, 32348, 32404, 32460, 32516, 32572, 32628, 32684, 32740, 32796, 32852, 32908, 32964, 33020, 33076, 33132, 33188, 33244, 33300, 33356, 33412, 33468, 33524, 33580, 33636, 33692, 33748, 33804, 33860, 33916, 33972, 34028, 34084, 34140, 34196, 34252, 34308, 34364, 34420, 34476, 34532, 34588, 34644, 34700, 34756, 34812, 34868, 34924, 34980, 35036, 35092, 35148, 35204, 35260, 35316, 35372, 35428, 35484, 35540, 35596, 35652, 35708, 35764, 35820, 35876, 35932, 35988, 36044, 36100, 36156, 36212, 36268, 36324, 36380, 36436, 36492, 36548, 36604, 36660, 36716, 36772, 36828, 36884, 36940, 36996, 37052, 37108, 37164, 37220, 37276, 37332, 37388, 37444, 37500, 37556, 37612, 37668, 37724, 37780, 37836, 37892, 37948, 38004, 38060, 38116, 38172, 38228, 38284, 38340, 38396, 38452, 38508, 38564, 38620, 38676, 38732, 38788, 38844, 38900, 38956, 39012, 39068, 39124, 39180, 39236, 39292, 39348, 39404, 39460, 39516, 39572, 39628, 39684, 39740, 39796, 39852, 39908, 39964, 40020, 40076, 40132, 40188, 40244, 40300, 40356, 40412, 40468, 40524, 40580, 40636, 40692, 40748, 40804, 40860, 40916, 40972, 41028, 41084, 41140, 41196, 41252, 41308, 41364, 41420, 41476, 41532, 41588, 41644, 41700, 41756, 41812, 41868, 41924, 41980, 42036, 42092, 42148, 42204, 42260, 42316, 42372, 42428, 42484, 42540, 42596, 42652, 42708, 42764, 42820, 42876, 42932, 42988, 43044, 43100, 43156, 43212, 43268, 43324, 43380, 43436, 43492, 43548, 43604, 43660, 43716, 43772, 43828, 43884, 43940, 43996, 44052, 44108, 44164, 44220, 44276, 44332, 44388, 44444, 44500, 44556, 44612, 44668, 44724, 44780, 44836, 44892, 44948, 45004, 45060, 45116, 45172, 45228, 45284, 45340, 45396, 45452, 45508, 45564, 45620, 45676, 45732, 45788, 45844, 45900, 45956, 46012, 46068, 46124, 46180, 46236, 46292, 46348, 46404, 46460, 46516, 46572, 46628, 46684, 46740, 46796, 46852, 46908, 46964, 47020, 47076, 47132, 47188, 47244, 47300, 47356, 47412, 47468, 47524, 47580, 47636, 47692, 47748, 47804, 47860, 47916, 47972, 48028, 48084, 48140, 48196, 48252, 48308, 48364, 48420, 48476, 48532, 48588, 48644, 48700, 48756, 48812, 48868, 48924, 48980, 49036, 49092, 49148, 49204, 49260, 49316, 49372, 49428, 49484, 49540, 49596, 49652, 49708, 49764, 49820, 49876, 49932, 49988, 50044, 50100, 50156, 50212, 50268, 50324, 50380, 50436, 50492, 50548, 50604, 50660, 50716, 50772, 50828, 50884, 50940, 50996, 51052, 51108, 51164, 51220, 51276, 51332, 51388, 51444, 51500, 51556, 51612, 51668, 51724, 51780, 51836, 51892, 51948, 52004, 52060, 52116, 52172, 52228, 52284, 52340, 52396, 52452, 52508, 52564, 52620, 52676, 52732, 52788, 52844, 52900, 52956, 53012, 53068, 53124, 53180, 53236, 53292, 53348, 53404, 53460, 53516, 53572, 53628, 53684, 53740, 53796, 53852, 53908, 53964, 54020, 54076, 54132, 54188, 54244, 54300, 54356, 54412, 54468, 54524, 54580, 54636, 54692, 54748, 54804, 54860, 54916, 54972, 55028, 55084, 55140, 55196, 55252, 55308, 55364, 55420, 55476, 55532, 55588, 55644, 55700, 55756, 55812, 55868, 55924, 55980, 56036, 56092, 56148, 56204, 56260, 56316, 56372, 56428, 56484, 56540, 56596, 56652, 56708, 56764, 56820, 56876, 56932, 56988, 57044, 57100, 57156, 57212, 57268, 57324, 57380, 57436, 57492, 57548, 57604, 57660, 57716, 57772, 57828, 57884, 57940, 57996, 58052, 58108, 58164, 58220, 58276, 58332, 58388, 58444, 58500, 58556, 58612, 58668, 58724, 58780, 58836, 58892, 58948, 59004, 59060, 59116, 59172, 59228, 59284, 59340, 59396, 59452, 59508, 59564, 59620, 59676, 59732, 59788, 59844, 59900, 59956, 60012, 60068, 60124, 60180, 60236, 60292, 60348, 60404, 60460, 60516, 60572, 60628, 60684, 60740, 60796, 60852, 60908, 60964, 61020, 61076, 61132, 61188, 61244, 61300, 61356, 61412, 61468, 61524, 61580, 61636, 61692, 61748, 61804, 61860, 61916, 61972, 62028, 62084, 62140, 62196, 62252, 62308, 62364, 62420, 62476, 62532, 62588, 62644, 62700, 62756, 62812, 62868, 62924, 62980, 63036, 63092, 63148, 63204, 63260, 63316, 63372, 63428, 63484, 63540, 63596, 63652, 63708, 63764, 63820, 63876, 63932, 63988, 64044, 64100, 64156, 64212, 64268, 64324, 64380, 64436, 64492, 64548, 64604, 64660, 64716, 64772, 64828, 64884, 64940, 64996, 65052, 65108, 65164, 65220, 65276, 65332, 65388, 65444, 65500, 65556, 65612, 65668, 65724, 65780, 65836, 65892, 65948, 66004, 66060, 66116, 66172, 66228, 66284, 66340, 66396, 66452, 66508, 66564, 66620, 66676, 66732, 66788, 66844, 66900, 66956, 67012, 67068, 67124, 67180, 67236, 67292, 67348, 67404, 67460, 67516, 67572, 67628, 67684, 67740, 67796, 67852, 67908, 67964, 68020, 68076, 68132, 68188, 68244, 68300, 68356, 68412, 68468, 68524, 68580, 68636, 68692, 68748, 68804, 68860, 68916, 68972, 69028, 69084, 69140, 69196, 69252, 69308, 69364, 69420, 69476, 69532, 69588, 69644, 69700, 69756, 69812, 69868, 69924, 69980, 70036, 70092, 70148, 70204, 70260, 70316, 70372, 70428, 70484, 70540, 70596, 70652, 70708, 70764, 70820, 70876, 70932, 70988, 71044, 71100, 71156, 71212, 71268, 71324, 71380, 71436, 71492, 71548, 71604, 71660, 71716, 71772, 71828, 71884, 71940, 71996, 72052, 72108, 72164, 72220, 72276, 72332, 72388, 72444, 72500, 72556, 72612, 72668, 72724, 72780, 72836, 72892, 72948, 73004, 73060, 73116, 73172, 73228, 73284, 73340, 73396, 73452, 73508, 73564, 73620, 73676, 73732, 73788, 73844, 73900, 73956, 74012, 74068, 74124, 74180, 74236, 74292, 74348, 74404, 74460, 74516, 74572, 74628, 74684, 74740, 74796, 74852, 74908, 74964, 75020, 75076, 75132, 75188, 75244, 75300, 75356, 75412, 75468, 75524, 75580, 75636, 75692, 75748, 75804, 75860, 75916, 75972, 76028, 76084, 76140, 76196, 76252, 76308, 76364, 76420, 76476, 76532, 76588, 76644, 76700, 76756, 76812, 76868, 76924, 76980, 77036, 77092, 77148, 77204, 77260, 77316, 77372, 77428, 77484, 77540, 77596, 77652, 77708, 77764, 77820, 77876, 77932, 77988, 78044, 78100, 78156, 78212, 78268, 78324, 78380, 78436, 78492, 78548, 78604, 78660, 78716, 78772, 78828, 78884, 78940, 78996, 79052, 79108, 79164, 79220, 79276, 79332, 79388, 79444, 79500, 79556, 79612, 79668, 79724, 79780, 79836, 79892, 79948, 80004, 80060, 80116, 80172, 80228, 80284, 80340, 80396, 80452, 80508, 80564, 80620, 80676, 80732, 80788, 80844, 80900, 80956, 81012, 81068, 81124, 81180, 81236, 81292, 81348, 81404, 81460, 81516, 81572, 81628, 81684, 81740, 81796, 81852, 81908, 81964, 82020, 82076, 82132, 82188, 82244, 82300, 82356, 82412, 82468, 82524, 82580, 82636, 82692, 82748, 82804, 82860, 82916, 82972, 83028, 83084, 83140, 83196, 83252, 83308, 83364, 83420, 83476, 83532, 83588, 83644, 83700, 83756, 83812, 83868, 83924, 83980, 84036, 84092, 84148, 84204, 84260, 84316, 84372, 84428, 84484, 84540, 84596, 84652, 84708, 84764, 84820, 84876, 84932, 84988, 85044, 85100, 85156, 85212, 85268, 85324, 85380, 85436, 85492, 85548, 85604, 85660, 85716, 85772, 85828, 85884, 85940, 85996, 86052, 86108, 86164, 86220, 86276, 86332, 86388, 86444, 86500, 86556, 86612, 86668, 86724, 86780, 86836, 86892, 86948, 87004, 87060, 87116, 87172, 87228, 87284, 87340, 87396, 87452, 87508, 87564, 87620, 87676, 87732, 87788, 87844, 87900, 87956, 88012, 88068, 88124, 88180, 88236, 88292, 88348, 88404, 88460, 88516, 88572, 88628, 88684, 88740, 88796, 88852, 88908, 88964, 89020, 89076, 89132, 89188, 89244, 89300, 89356, 89412, 89468, 89524, 89580, 89636, 89692, 8 |

# Testing the DOM for the WWW Consortium



# Document Object Model Events

## Original test set:

| Event Name                  | Param. | Tests |
|-----------------------------|--------|-------|
| Abort                       | 3      | 12    |
| Blur                        | 5      | 24    |
| Click                       | 15     | 4352  |
| Change                      | 3      | 12    |
| dblClick                    | 15     | 4352  |
| DOMActivate                 | 5      | 24    |
| DOMAttrModified             | 8      | 16    |
| DOMCharacterDataModified    | 8      | 64    |
| DOMElementNameChanged       | 6      | 8     |
| DOMFocusIn                  | 5      | 24    |
| DOMFocusOut                 | 5      | 24    |
| DOMNodeInserted             | 8      | 128   |
| DOMNodeInsertedIntoDocument | 8      | 128   |
| DOMNodeRemoved              | 8      | 128   |
| DOMNodeRemovedFromDocument  | 8      | 128   |
| DOMSubTreeModified          | 8      | 64    |
| Error                       | 3      | 12    |
| Focus                       | 5      | 24    |
| KeyDown                     | 1      | 17    |
| KeyUp                       | 1      | 17    |
| Load                        | 3      | 24    |
| MouseDown                   | 15     | 4352  |
| MouseMove                   | 15     | 4352  |
| MouseOut                    | 15     | 4352  |
| MouseOver                   | 15     | 4352  |
| MouseUp                     | 15     | 4352  |
| MouseWheel                  | 14     | 1024  |
| Reset                       | 3      | 12    |
| Resize                      | 5      | 48    |
| Scroll                      | 5      | 48    |
| Select                      | 3      | 12    |
| Submit                      | 3      | 12    |
| TextInput                   | 5      | 8     |
| Unload                      | 3      | 24    |
| Wheel                       | 15     | 4096  |
| Total Tests                 |        | 36626 |

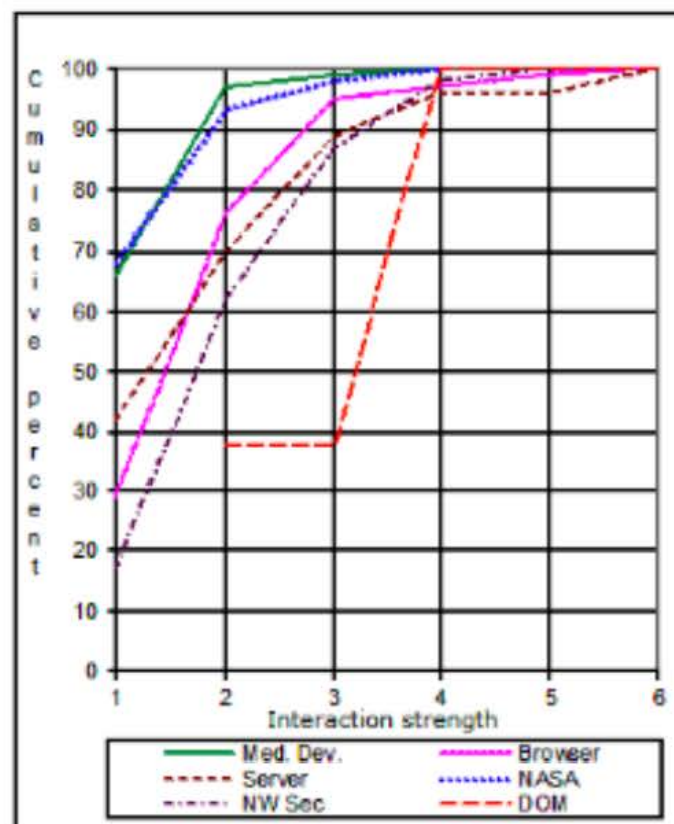
**36,626 tests!**

Exhaustive testing of  
equivalence class  
values

## Combinatorial test set:

| t | Tests | % of Orig. | Test Results |      |         |
|---|-------|------------|--------------|------|---------|
|   |       |            | Pass         | Fail | Not Run |
| 2 | 702   | 1.92%      | 202          | 27   | 473     |
| 3 | 1342  | 3.67%      | 786          | 27   | 529     |
| 4 | 1818  | 4.96%      | 437          | 72   | 1309    |
| 5 | 2742  | 7.49%      | 908          | 72   | 1762    |
| 6 | 4227  | 11.54%     | 1803         | 72   | 2352    |

All failures found using < 5% of original exhaustive test set



## **Problem: unknown factors causing failures of F-16 ventral fin**



Figure 1. LANTIRN pod carriage on the F-16.



**It's not supposed to look like this:**



A04-14639001

Figure 2. F-16 ventral fin damage on flight with LANTIRN

| Parameter                 | Values  |
|---------------------------|---|
| Aircraft                  | 15, 40  |
| Altitude                  | 5k, 10k, 15k, 20k, 30k, 40k, 50k  |
| Maneuver                  | hi-speed throttle, slow accel/dwell, L/R 5 deg side slip, L/R 360 roll, R/L 5 deg side slip, Med accel/dwell, R-L-R-L banking, Hi-speed to Low, 360 nose roll |
| Mach (100 <sup>th</sup> ) | 40, 50, 60, 70, 80, 90, 100, 110, 120   |

*Combinatorial testing was able to find the problem with less flight- and expert-time.*



## *Security Testing*

**Analysis of buffer overflows by NIST:**

**94.7% - Single variable problem (e.g. not checking for length of ftps:// string)**

**4.9% - Two-way interaction (e.g. a particular search string with a particular replacement string)**

**0.4% - Three-way interaction (e.g. directory traversal enabled, magic\_quotes enabled, and ".." in the page parameter)**

# *Benefits and Drawbacks*

## Benefits:

1. Great test coverage
2. Maximise efficiency
3. Can gauge coverage
4. Can turn dial "up to 11"
5. Very good growth rates as number of interactions increase

## Drawbacks:

1. May be overkill for small projects
2. Extra time to make tests (albeit minimal)
3. New features - may have to re-run and re-create tests instead of just adding
4. Automation?

## **Benefits:**

- 1. Great test coverage**
- 2. Maximize efficiency**
- 3. Can gauge coverage**
- 4. Can turn dial "up to 11"**
- 5. Very good growth rates as number of interactions increase**

## **Drawbacks:**

- 1. May be overkill for small projects**
- 2. Extra time to make tests (albeit minimal)**
- 3. New features - may have to re-run and re-create tests instead of just adding**
- 4. Automation?**

**You could also use combinatorial testing for ordering of events:**


**a, b, c, d, e, f**

**Pairs become "a then b", "b then a", etc.**


# CS1699 - Lecture 17 - Pairwise & Combinatorial Testing

\* ACTS and NIST imagery used with permission

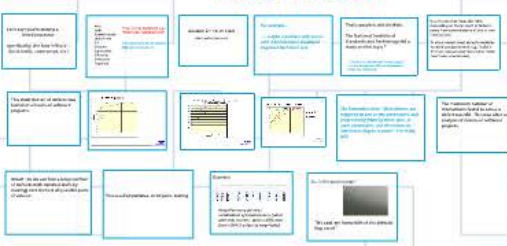
### But How Do We Make Those Tables?




### Combinatorial Testing



### The Problem



### Real-World Uses



### Benefits and Drawbacks

