Combinatorial Testing

- Introduction
- Combinatorial Coverage Criteria
- Pairwise Test Generation
- Summary

Motivation

- The behavior of a software application may be affected by many factors
  - For example, input parameters, environment configurations, and state variables.
- Techniques like equivalence partitioning and boundary-value analysis can be used to identify the possible values of each factor.
- It is impractical to test all possible combinations of values of all those factors. (Why?)

Combinatorial Explosion

- Assume that an application has 10 parameters, each of which can take 5 values. How many possible combinations?

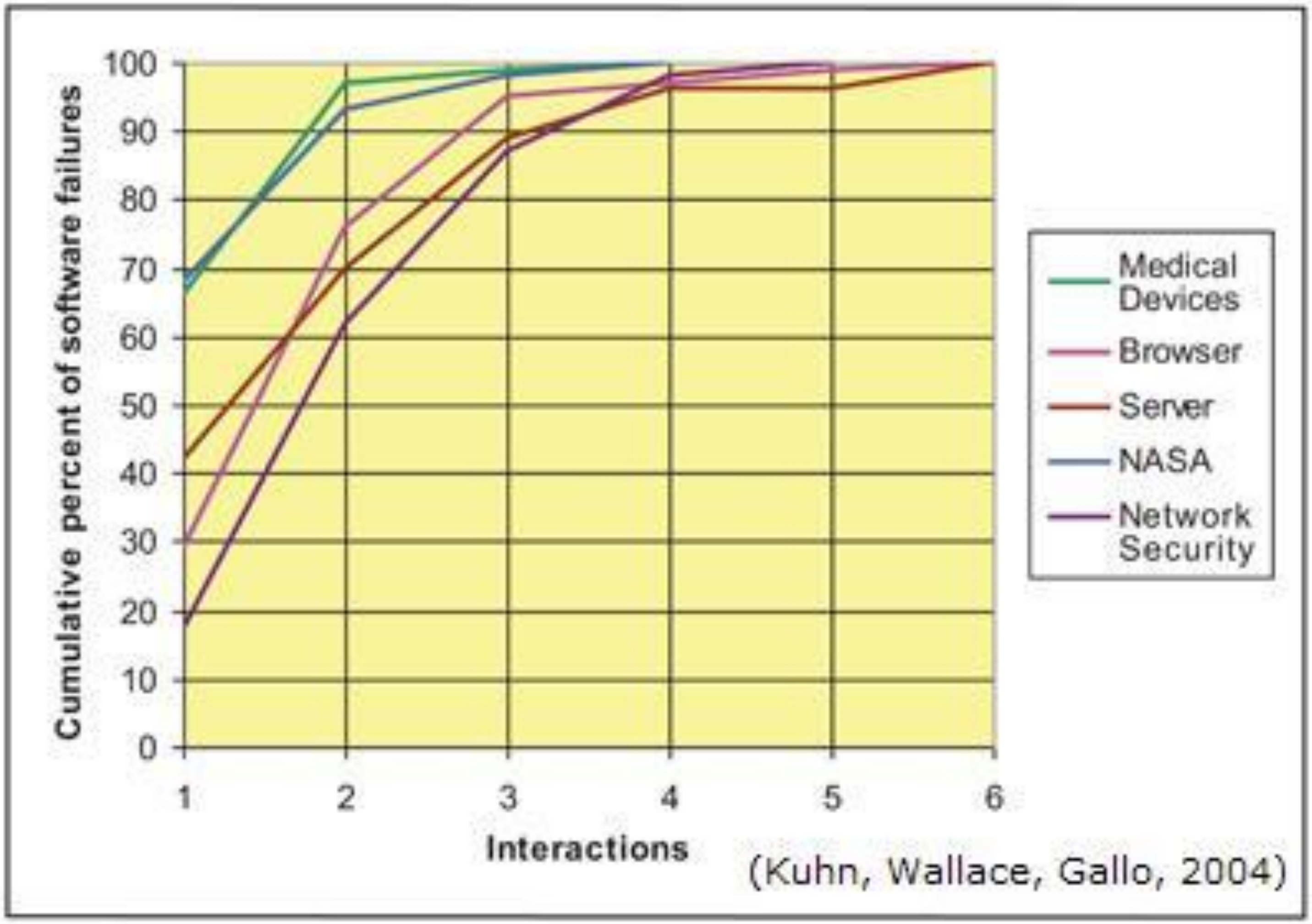
Example - sort

```
> man sort
Reformatting page.  Wait... done

User Commands                                   sort(1)

NAME
   sort - sort, merge, or sequence check text files

SYNOPSIS
   /usr/bin/sort [ -cmu ] [ -o output ] [ -T directory ]
      [ -y [ kmem ]] [ -z recsz ]  [  -dfiMnr  ]  [  -b  ]  [
   -t char ]
      [ -k keydef ] [ +pos1 [ -pos2 ]] [ file...]
…
```

A Bug’s Perspective

- As a whole, the behavior of a system could be affected by many factors.
- However, individual bugs are often affected by only a few factors.
  - A widely-cited NIST study suggests no more than 6 factors.
- But how do we know “what” parameters affect “what” bugs?

NIST Study



Cumulative percent of software failures
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
Interactions
Medical Devices
Browser
Server
NASA
Network Security
(Kuhn, Wallace, Gallo, 2004)

Combinatorial Design

- A t-way test set covers all the t-way combinations, instead of all possible combinations (of all the parameters)
  - No need to know “what” parameters cause “what” faults.
- Extremely effective yet substantially reduces the number of tests
  - 10 5-value parameters (about 10M possible tests): 2-way testing – 49 tests; 3-way testing – 307 tests; 4-way testing – 1865 tests

An Example T-Way Test Set

Consider a system that has three parameters, each having two values 0 and 1.

P1	P2	P3
0	0	0
0	1	1
1	0	1
1	1	0

Pick ANY two parameters, all combinations 00, 01, 10, 11 are covered.

Fault Model

- A t-way interaction fault is a fault that is triggered by a certain combination of t input values.
- A simple fault is a t-way fault where t = 1; a pairwise fault is a t-way fault where t = 2.
- In practice, the majority of faults in a software application consist of simple and pairwise faults.

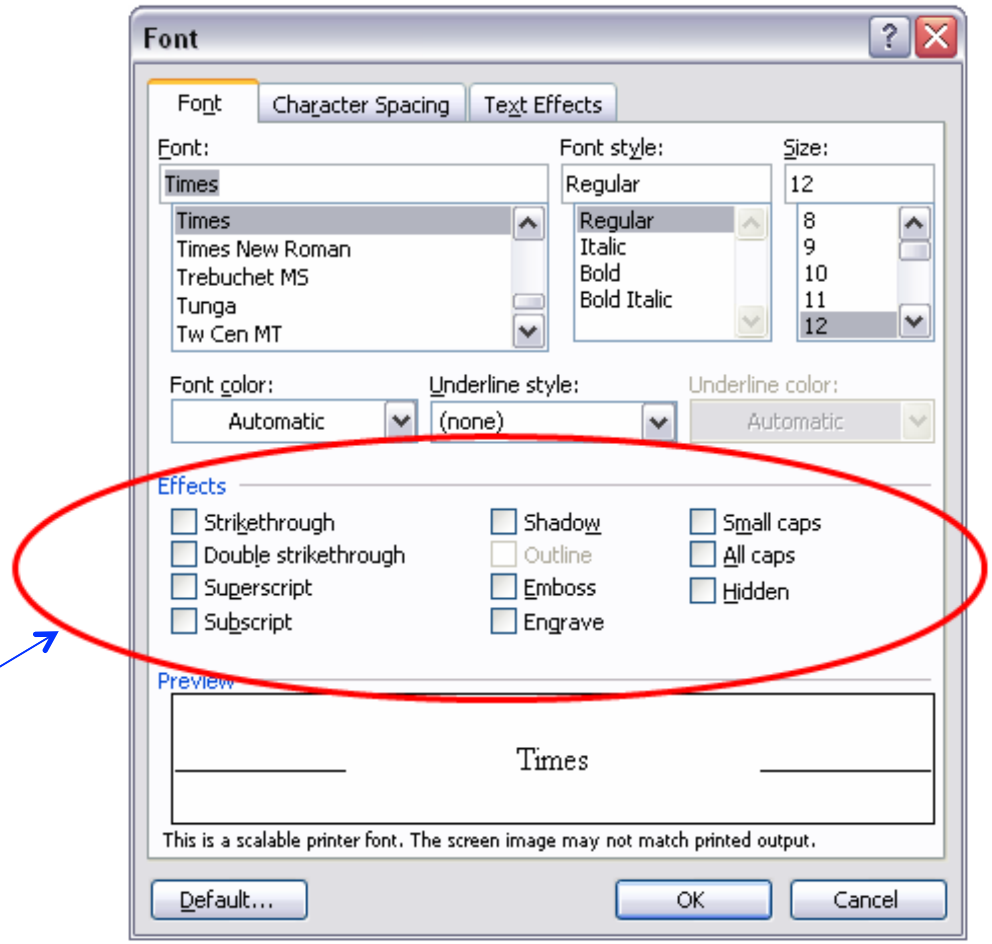
Example – Pairwise Fault

```
begin
    int x, y, z;
    input (x, y, z);
    if (x == x1 and y == y2)
        output (f(x, y, z));
    else if (x == x2 and y == y1)
        output (g(x, y));
    else
        output (f(x, y, z) + g(x, y))
end
```

Expected: x = x1 and y = y1 => f(x, y, z) – g(x, y); x = x2, y = y2 => f(x, y, z) + g(x, y)

A Real-World Example

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	0	1	0	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



0 = effect on

1 = effect off

13 tests for **all 3-way combinations**

2^{10} = **1,024** tests for **all combinations**

(From Rick Kuhn)

Combinatorial Testing

- Introduction
- Combinatorial Coverage Criteria
- Pairwise Test Generation
- Summary

All Combinations Coverage

- Every possible combination of values of the parameters must be covered
- For example, if we have three parameters P1 = (A, B), P2 = (1, 2, 3), and P3 = (x, y), then all combinations coverage requires 12 tests:
  - {(A, 1, x), (A, 1, y), (A, 2, x), (A, 2, y), (A, 3, x), (A, 3, y), (B, 1, x), (B, 1, y), (B, 2, x), (B, 2, y), (B, 3, x), (B, 3, y)}

Each Choice Coverage

- Each parameter value must be covered in at least one test case.
- Consider the previous example, a test set that satisfies each choice coverage is the following: {(A, 1, x), (B, 2, y), (A, 3, x)}

P1	P2	P3
A	1	x
B	2	y
A	3	x

Pairwise Coverage

- Given any two parameters, every combination of values of these two parameters is covered in at least one test case.
- A pairwise test set of the previous example is the following:

P1	P2	P3
A	1	x
A	2	x
A	3	x
A	-	y
B	1	y
B	2	y
B	3	y
B	-	x

T-Wise Coverage

- Given any t parameters, every combination of values of these t parameters must be covered in at least one test case.
- For example, a 3-wise coverage requires every triple be covered in at least one test case.
- Note that all combinations, each choice, and pairwise coverage can be considered to be a special case of t-wise coverage.

Base Choice Coverage

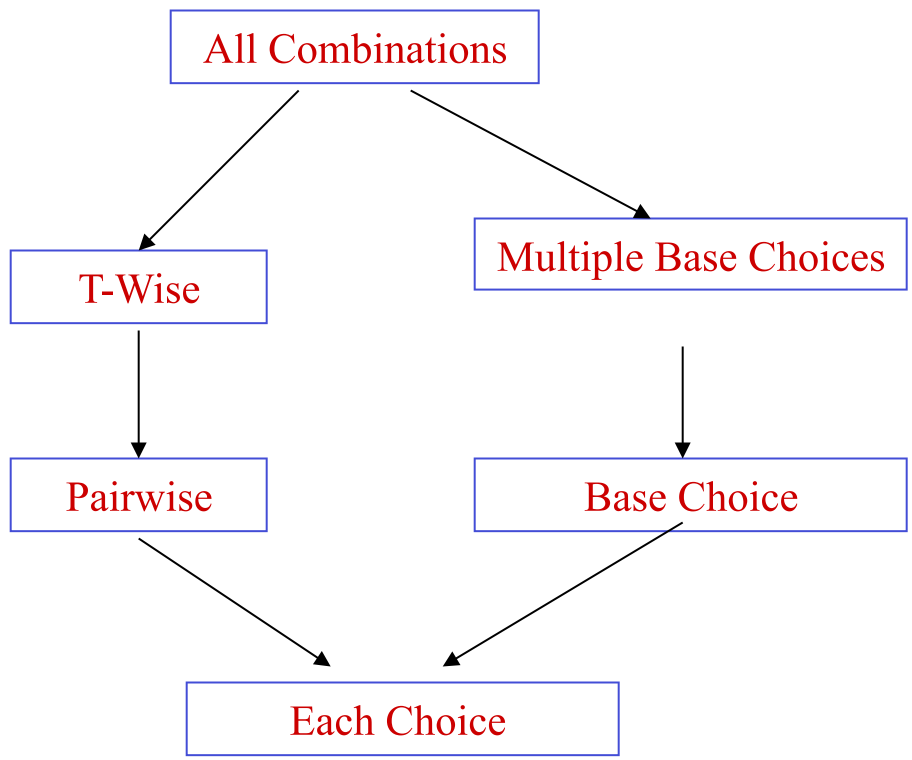
- For each parameter, one of the possible values is designated as a base choice of the parameter
- A base test is formed by using the base choice for each parameter
- Subsequent tests are chosen by holding all base choices constant, except for one, which is replaced using a non-base choice of the corresponding parameter:

P1	P2	P3
A	1	x
B	1	x
A	2	x
A	3	x
A	1	y

Multiple Base Choices Coverage

- At least one, and possibly more, base choices are designated for each parameter.
- The notions of a base test and subsequent tests are defined in the same as Base Choice.

Subsumption Relation



Combinatorial Testing

- Introduction
- Combinatorial Coverage Criteria
- Pairwise Test Generation
- Summary

Why Pairwise?

- Many faults are caused by the interactions between two parameters
- Not practical to cover all the parameter interactions
- Pairwise testing makes a good balance between test coverage and test effort

Example (1)

Consider a system with the following parameters and values:

- parameter A has values A1 and A2
- parameter B has values B1 and B2, and
- parameter C has values C1, C2, and C3

Example (2)

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1
A2	B1	C2
A1	B2	C3

A	B	C
A1	B1	C1
A1	B2	C1
A2	B1	C2
A2	B2	C3
A2	B1	C1
A1	B2	C2
A1	B1	C3

A	B	C
A1	B1	C1
A1	B2	C1
A2	B1	C2
A2	B2	C2
A2	B1	C1
A1	B1	C2
A1	B1	C3
A2	B2	C3

The IPO Strategy

- First generate a pairwise test set for the first two parameters, then for the first three parameters, and so on
- A pairwise test set for the first n parameters is built by extending the test set for the first n – 1 parameters
  - Horizontal growth: Extend each existing test case by adding one value of the new parameter
  - Vertical growth: Adds new tests, if necessary

Algorithm IPO_H (T, p_i)

Assume that the domain of p_i contains values v_1, v_2, …, and v_q;
π = { pairs between values of p_i and values of p_1, p_2, …, and p_{i-1}
if ($|T| <= q$)
  for $1 <= j <= |T|$, extend the j^{th} test in T by adding value v_j and remove from π pairs covered by the extended test
else
  for $1 <=j <= q$, extend the j^{th} test in T by adding value v_j and remove from π pairs covered by the extended test;
  for $q < j <= |T|$, extend the j^{th} test in T by adding one value of p_i such that the resulting test covers the most number of pairs in π, and remove from π pairs covered by the extended test

Algorithm IPO_V(T, π)

let T' be an empty set;
for each pair in π
  assume that the pair contains value w of p_k, $1 \leq k < i$, and value u of p_i;
  if (T contains a test with "-" as the value of p_k and u as the value of p_i)
    modify this test by replacing the "-" with w
  else
    add a new test to T' that has w as the value of p_k, u as the value of p_i, and "-" as the value of every other parameter;
  $T = T \cup T'$

Example Revisited

Show how to apply the IPO strategy to construct the pairwise test set for the example system.

A	B
A1	B1
A1	B2
A2	B1
A2	B2

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1

Horizontal Growth

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1
A2	B1	C2
A1	B2	C3

Vertical Growth

Example Revisited (2)

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1

A1	B2	C3
A2	B1	C2

A	C
~~A1~~	~~C1~~
~~A1~~	~~C2~~
~~A1~~	~~C3~~
~~A2~~	~~C1~~
~~A2~~	~~C2~~
~~A2~~	~~C3~~

B	C
~~B1~~	~~C1~~
~~B1~~	~~C2~~
~~B1~~	~~C3~~
~~B2~~	~~C1~~
~~B2~~	~~C2~~
~~B2~~	~~C3~~

C1: 2, C2: 1, C3: 1

Combinatorial Testing

- Introduction
- Combinatorial Coverage Criteria
- Pairwise Test Generation
- Summary

Summary

- Combinatorial testing makes an excellent trade-off between test effort and test effectiveness.
- Pairwise testing can often reduce the number of tests dramatically, but it can still detect faults effectively.
- The IPO strategy constructs a pairwise test set incrementally, one parameter at a time.
- In practice, some combinations may be invalid from the domain semantics, and must be excluded, e.g., by means of constraint processing.