Input Domain Testing

Software Testing

Trainer: Aleksandar Karamfilov
Email: alex@pragmatic.bg
Facebook: facebook.com/a.karamfilov

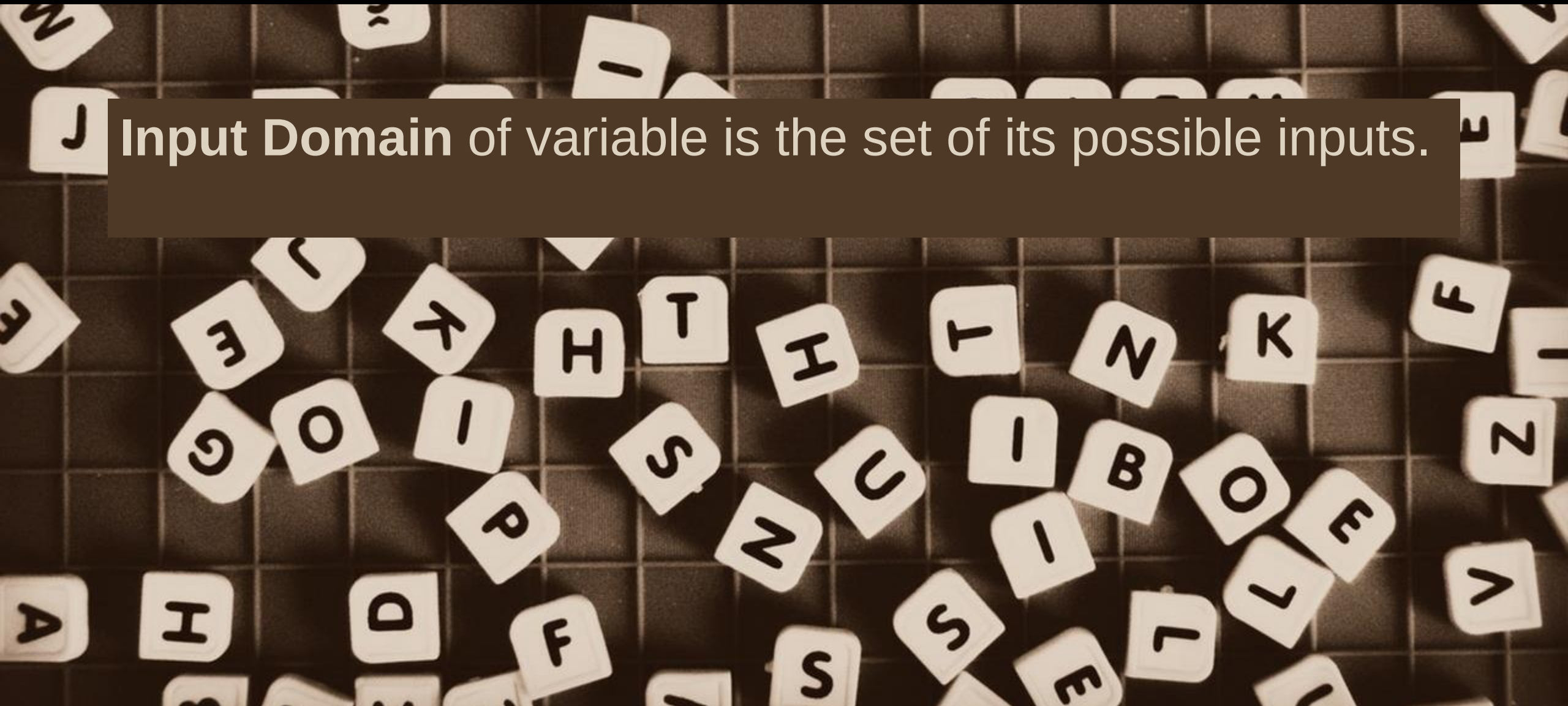
PRAGMATIC

www.pragmatic.bg

Outline

- Equivalence Class Partitioning
- Boundary Value Analysis
- N-Wise(Pairwise)

Input Domain



Input Domain of variable is the set of its possible inputs.

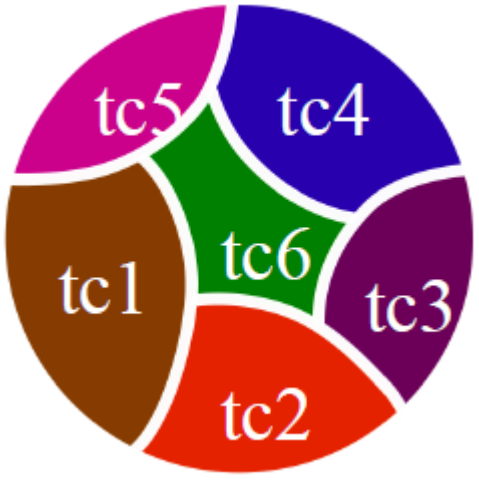
Input Domain Coverage

Input Domain Coverage means having enough test cases to test the domain of each input variable(field).

Equivalence Class Partitioning

Equivalence Class Partitioning

- Motivation: we would like to have a sense of complete testing and would hope to avoid test redundancy
- Equivalence classes: partitions of the input set in which input data have the same effect on the program
- Entire input set is covered: completeness
- Disjoint classes: to avoid redundancy
- Test cases: one element of each equivalence class



Weak & Strong Equivalence Testing

- Weak Equivalence Class Testing: Choosing one variable value from each equivalence class (one a, b, and c) such that all classes are covered.
- Strong Equivalence Class Testing: Is based on the Cartesian product of the partition subsets (A×B×C), i.e., testing all interactions of all equivalence classes.

# of test cases?

– |A| x |B| x |C|

Example of Weak Class Testing

- Number of WETCs need = Max number of equivalence classes

Test Case	A	B	C
WETC1	A1	B1	C1
WETC2	A2	B2	C2
WETC3	A3	B3	C3
WETC4	A4	B4	C4

Example of Strong Class Testing

A = 2, B = 3, C = 3

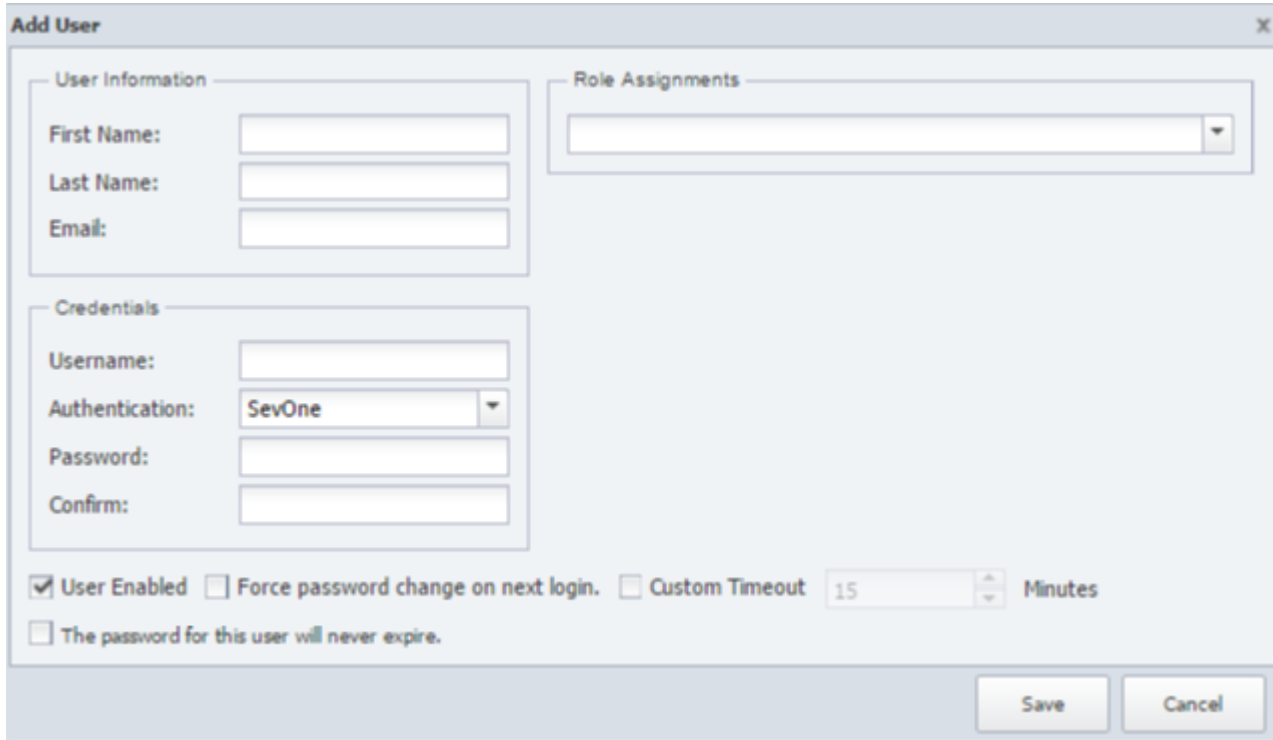
Number of tests for strong class testing = 2 x 3 x 3 => 12

Test Case	A	B	C
SETC1	A1	B1	C1
SETC2	A1	B1	C2
SETC3	A1	B1	C3
SETC4	A1	B2	C1
SETC5	A1	B2	C2
SETC6	A1	B2	C3
…	…	…	…

Real System Example

Example

First/Last Name: [1- 24], Latin letters only, case insensitive, apostrophes
Authentication: SevOne, TACACS, LDAP, RADIUS
Role Assignment: Single or multi selection, existing roles
Username: [1-24], Latin only, case insensitive



First Name Analysis

First/Last Name: [1- 24], Latin letters only, case insensitive

Field/Variable	Class	Exemplar	Expected
First/Last Name	All lower case [1,24]	alex	Pass
	Mixed case [1,24]	Alex, aLex, aleX	Pass
	All capital [1,24]	ALEX	Pass
	Special cases [1,24]	D'Arcy	Pass
	[0, 1)	Blank	Fail
	> 24	alexalexalexalexalexalexa	Fail
	Special chars	$<%^@#	Fail

Username Analysis

Username: [1- 24], Latin letters only, case insensitive

Field/Variable	Class	Exemplar	Expected
Username	All lower case [1,24]	alex	Pass
	Mixed case [1,24]	Alex, aLex, aleX	Pass
	All capital [1,24]	ALEX	Pass
	Special cases [1,24]	alex_admin	Pass
	[0, 1)	Blank	Fail
	> 24	alexalexalexalexalexalex	Fail
	Special chars	$<%^@#	Fail

Authentication Analysis

Authentication: SevOne, TACACS, LDAP, RADIUS

Field/Variable	Class	Exemplar	Expected
Authentication	SevOne	SevOne	Pass
	TACACS	TACACS	Pass
	LDAP	LDAP	Pass
	RADIUS	RADIUS	Pass

Example of Weak Class Testing

- LinearQ

Test Case	First Name	Last Name	Username	Auth
1	alex	alex	alex	SevOne
2	ALEX	ALEX	ALEX	TACACS
3	Alex	Alex	Alex	RADIUS
4	aLex	aLex	aLex	LDAP
5	aleX	aleX	aleX	SevOne
6	D’arcy	D’arcy	Alex_admin	TACACS
7	Blank	Blank	Blank	RADIUS
8	#%#$	#%#$	#%#$	LDAP

Boundary Value Analysis

Motivations

- We have partitioned input domains into suitable classes, on the assumption that behavior of the program is “similar”
- Some typical programming errors happen at the boundary between different classes
- This is what boundary value testing focuses on
- Simpler but complementary to previous techniques

Error at the boundaries

- Experience indicates that programmers make mistakes in processing values at and near the boundaries of equivalence classes.
- For example, suppose that method M is required to compute a function f1 when x<=0 is true and function f2 otherwise. However, M has an error due to which it computes f1 x<0 and f2 otherwise.
- Obviously, this fault is revealed, though not necessarily, when M is tested against x = 0 but not if the input test set is, for example {-4, 7} derived using EP. In this case, the value x=0, lies at the boundary of the equivalence x<=0 and x>0

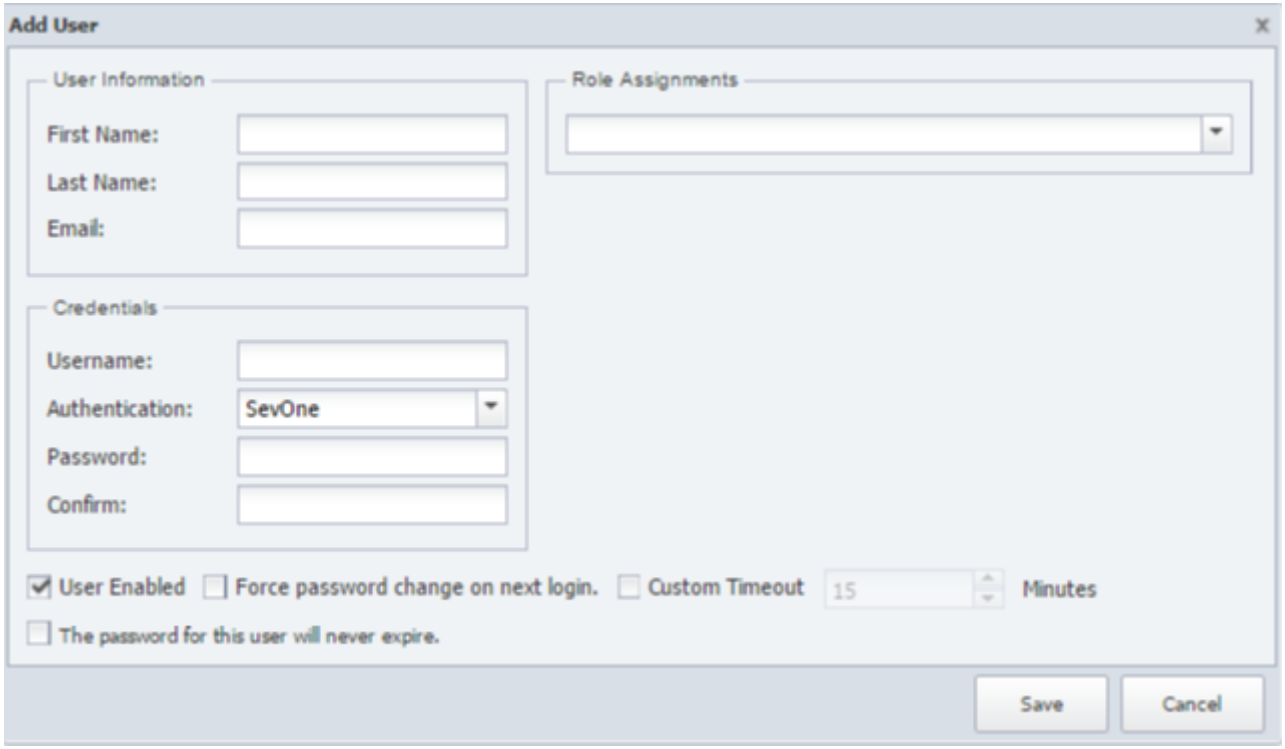
Boundary Value Analysis

- BVA is a test selection technique that targets faults in applications at the boundaries of equivalence classes.
- While EP selects tests from within equivalence classes, boundary value analysis focuses on tests at and near the boundaries of equivalence classes.
- Certainly, tests derived using either of the two techniques may overlap

Real System Example

Example

First/Last Name: [1- 24]



First Name Analysis

First/Last Name: [1- 24]

Field/Variable	Boundary	Exemplar	Expected
First/Last Name	1	A(1)	Pass
		AL(2)	Pass
		Blank(0)	Fail
	24	Alexalexalexalexalexalex(24)	Pass
		Alexalexalexalexalexale(23)	Pass
		AlexalexalexalexalexalexA(25)	Fail

Task

Perform EP Analysis

Function *getPrice(int itemCode, int itemQuantity);*
itemCode [99, 999]
itemQuantity [1, 100]

Field/Variable	Class	Exemplar	Expected
itemCode/itemQuantity			

Solution

Field/Variable	Class	Exemplar	Expected
itemCode	[99, 999]	200	Pass
	(-infinity, 98]	50	Fail
	[1000, +infinity)	5000	Fail

Field/Variable	Class	Exemplar	Expected
itemQuantity	[1, 100]	50	Pass
	[-infinity, 0]	-5	Fail
	[101, +infinity)	200	Fail

Perform BVA Analysis

Function *getPrice(int itemCode, int itemQuantity);*
itemCode [99, 999]
itemQuantity [1, 100]

Field/Variable	Boundary	Exemplars	Expected
itemCode/itemQuantity			

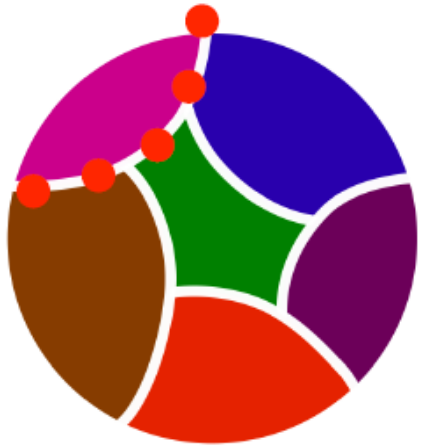
Solution

Field/Variable	Boundary	Exemplar	Expected
itemCode	99	99	Pass
		100	Pass
		98	Fail
	999	999	Pass
		998	Pass
		1000	Fail

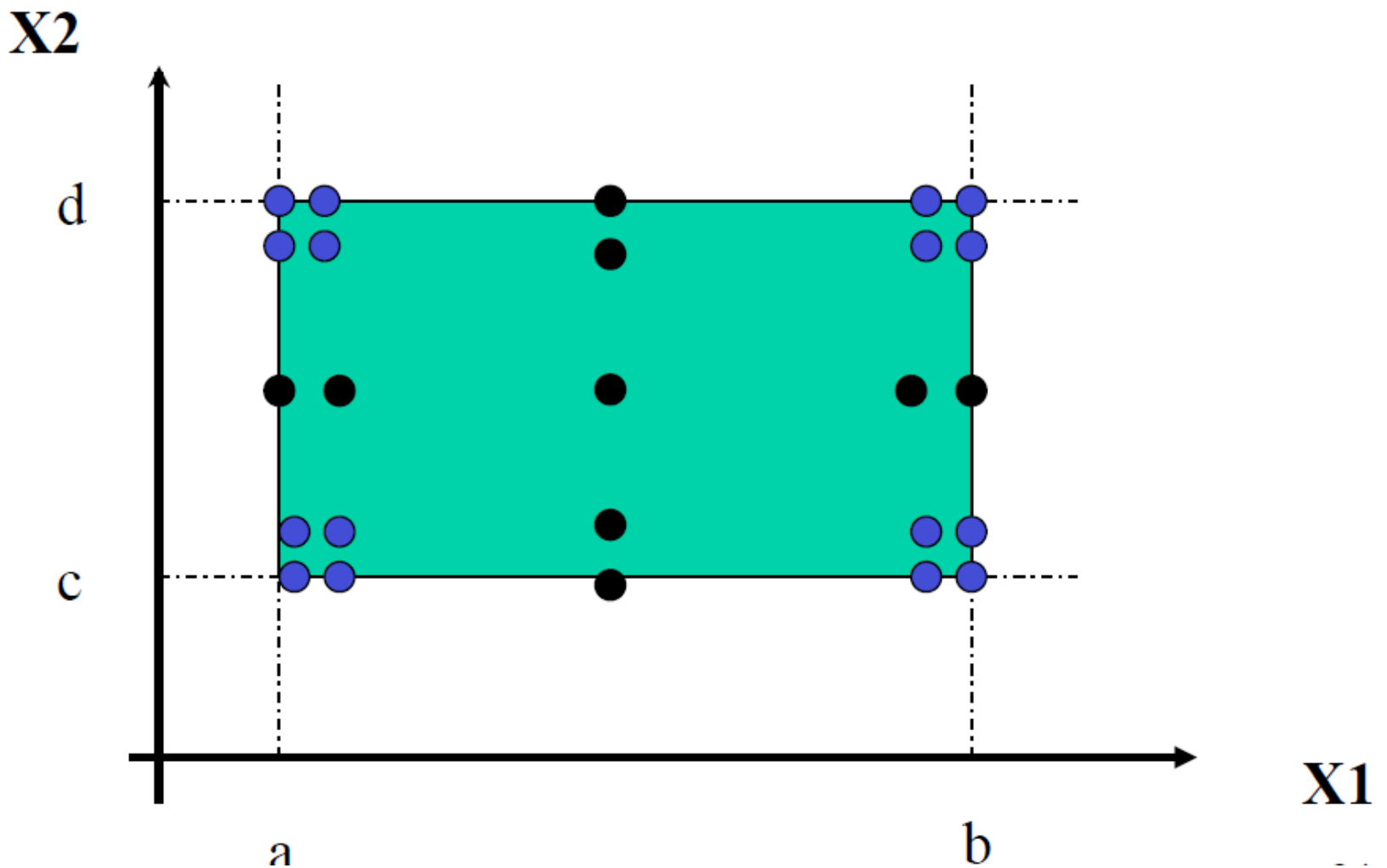
Field/Variable	Boundary	Exemplar	Expected
itemQuantity	1	1	Pass
		2	Pass
		0	Fail
	100	100, 99	Pass
		101	Fail

Worst Case Testing (WCT)

- BVA makes the common assumption that failures, most of the time, originate from one fault related to an extreme value.
- What happens when more than one variable has an extreme value?
- Idea comes from electronics in circuit analysis
- Cartesian product of {min, min+, nom, max-, max}
- Clearly more thorough than boundary value analysis, but much more effort. Good strategy when failure is costly.



WCT for 2 variables



Conclusions

- Identifying parameters and environments conditions, and categories, is heavily relying on the experience of the tester.
- Makes testing decision explicit(e.g., constraints), open for review
- Combine BVA and EP
- Techniques for test case reduction makes it useful for practical testing

N-Wise

N-Wise

N-wise testing has the aim of testing all the possibilities of any random combination of N factors.

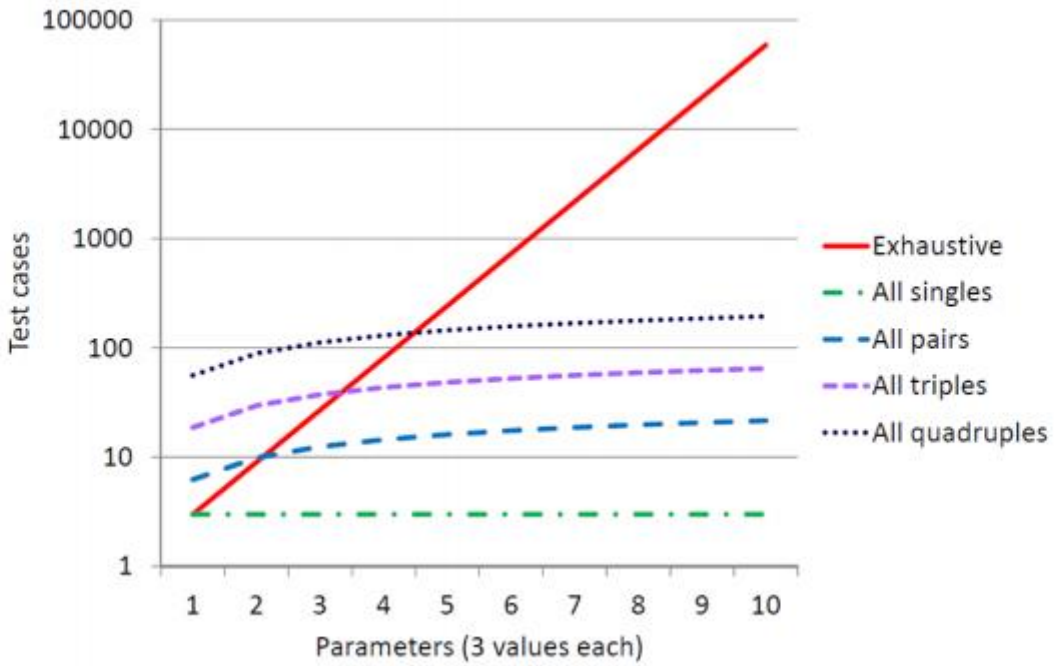
The maximum value for N is equal to the number of parameters. In that case, the result is equal to the testing of the complete decision table: all the combinations of all the values of all the parameters. In practice, a value of 4 or higher is seldom applied. In order to apply N-wise testing tools are required.

Pairwise Testing

The most common application of N-wise testing is pairwise testing. Pairwise testing is based on the phenomenon that most faults in software are the consequence of one particular factor or the combination of 2 factors. The number of faults that are caused by a specific combination of more than 2 factors becomes exponentially smaller. Instead of testing all the possible combinations of all the factors, it is very effective if every combination of 2 factors is tested.

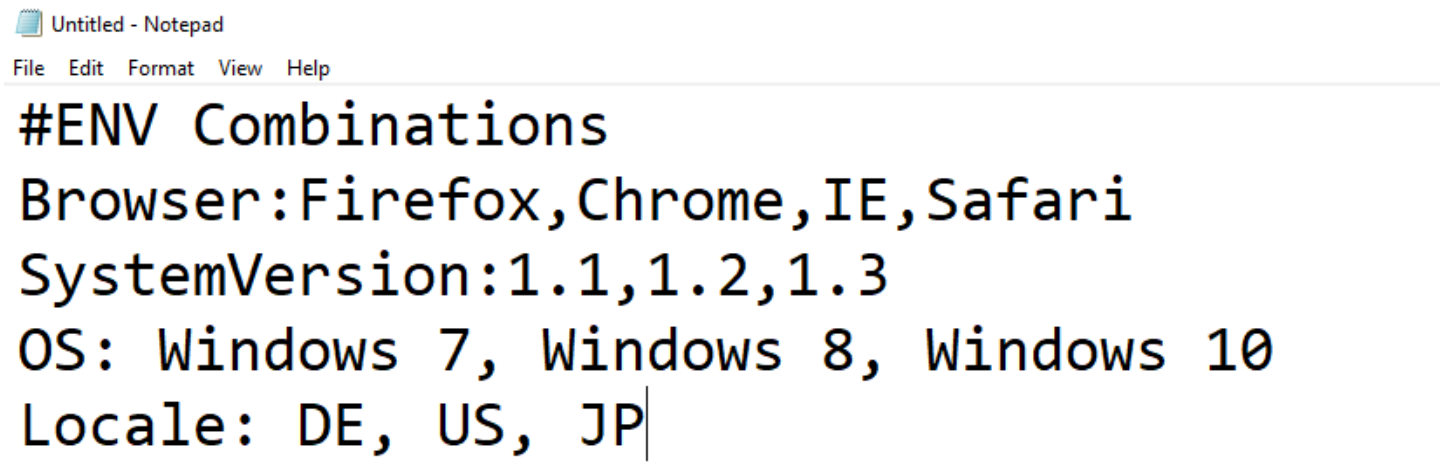
The aim of pairwise testing is to test all the possibilities of any combination of 2 factors.

Pairwise Testing



Input Model for Pairwise

Browser	System Version	OS	Locale
Firefox	1.1	Windows 7	DE
Chrome	1.2	Windows 8	US
IE	1.3	Windows 10	JP
Safari			



PICT Availability

```
Alex@KARAMFILOV MINGW64 /
$ pict
Pairwise Independent Combinatorial Testing

Usage: pict model [options]

Options:
 /o:N    - Order of combinations (default: 2)
 /d:C    - Separator for values  (default: ,)
 /a:C    - Separator for aliases (default: |)
 /n:C    - Negative value prefix (default: ~)
 /e:file - File with seeding rows
 /r[:N]  - Randomize generation, N - seed
 /c      - Case-sensitive model evaluation
 /s      - Show model statistics

Alex@KARAMFILOV MINGW64 /
$
```

Pairwise Combinations

```
Alex@KARAMFILOV MINGW64 ~/Desktop
$ pict env.txt
Browser SystemVersion   OS         Locale
Chrome  1.1     Windows 7          DE
Safari  1.2     Windows 7          JP
Firefox 1.3     Windows 10         JP
Chrome  1.3     Windows 8          JP
IE      1.2     Windows 10         US
Safari  1.1     Windows 8          US
IE      1.1     Windows 10         JP
IE      1.3     Windows 7          DE
Chrome  1.2     Windows 10         DE
Firefox 1.3     Windows 7          US
Firefox 1.2     Windows 8          DE
Safari  1.3     Windows 10         DE
Firefox 1.1     Windows 10         US
Chrome  1.3     Windows 7          US
IE      1.1     Windows 8          US
```

Output to File

```
Alex@KARAMFILOV MINGW64 ~/Desktop
$ pict env.txt > env_combinations.xls

Alex@KARAMFILOV MINGW64 ~/Desktop
$
```

Result File

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Browser	SystemVersion	OS	Locale										
2	Chrome	1.1	Windows 7	DE										
3	Safari	1.2	Windows 7	JP										
4	Firefox	1.3	Windows 10	JP										
5	Chrome	1.3	Windows 8	JP										
6	IE	1.2	Windows 10	US										
7	Safari	1.1	Windows 8	US										
8	IE	1.1	Windows 10	JP										
9	IE	1.3	Windows 7	DE										
10	Chrome	1.2	Windows 10	DE										
11	Firefox	1.3	Windows 7	US										
12	Firefox	1.2	Windows 8	DE										
13	Safari	1.3	Windows 10	DE										
14	Firefox	1.1	Windows 10	US										
15	Chrome	1.3	Windows 7	US										
16	IE	1.1	Windows 8	US										

PICT Conditions

```
Alex@KARAMFILOV MINGW64 ~/Desktop
$ pict env.txt
Browser SystemVersion   OS      Locale
Firefox 1.1     MacOs   DE
Firefox 1.3     Windows 7       US
Chrome  1.1     Windows 7       JP
Chrome  1.2     MacOs   US
Firefox 1.2     Windows 10      JP
Firefox 1.2     Windows 8       DE
Safari  1.3     MacOs   DE
IE      1.3     Windows 8       JP
IE      1.1     Windows 10      DE
Safari  1.2     MacOs   JP
Chrome  1.3     Windows 10      US
IE      1.2     Windows 7       US
Chrome  1.1     Windows 8       US
Chrome  1.2     Windows 7       DE
Safari  1.1     MacOs   US
IE      1.1     MacOs   DE
```

IF [Browser] = "Safari" Then [OS] = "MacOs";

IF [OS] in {"Windows 7", "Windows 8"} THEN [Browser] = "Firefox";

PICT Input File

```
#ENV Combinations
Browser:Firefox,Chrome,IE,Safari
SystemVersion:1.1,1.2,1.3
OS: Windows 7, Windows 8, Windows 10, MacOs
Locale: DE, US, JP
IF [Browser] = "Safari" Then [OS] = "MacOs";
```

Questions