Testing Functionality.

sources of information that can be used to design test cases

1. No information

   random testing.

2. Specification

   specification-based testing.

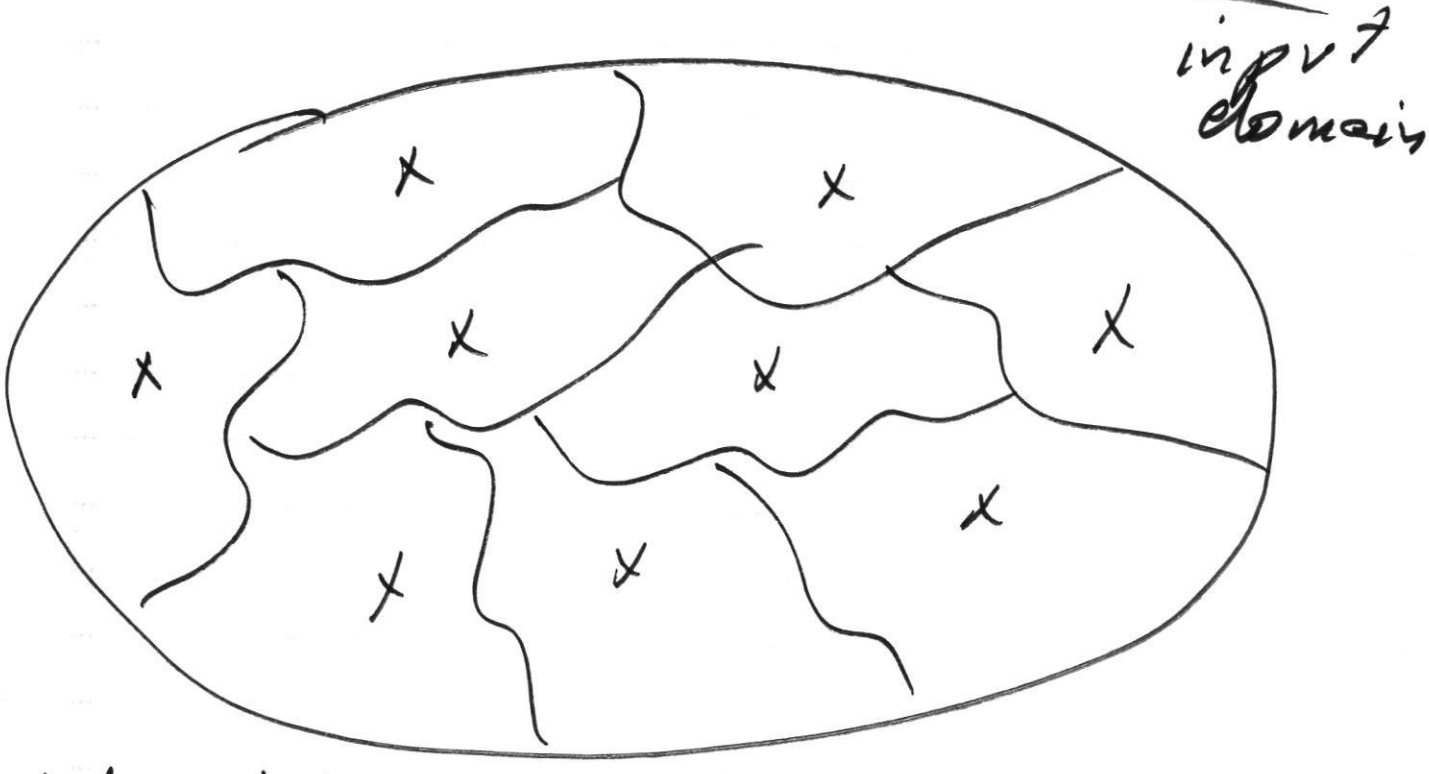
3. Source-code

   code-based testing.

Specification-based testing.

1. Equivalence partitioning testing.
2. Boundary-value analysis testing.
3. decision-table testing.
4. . . .

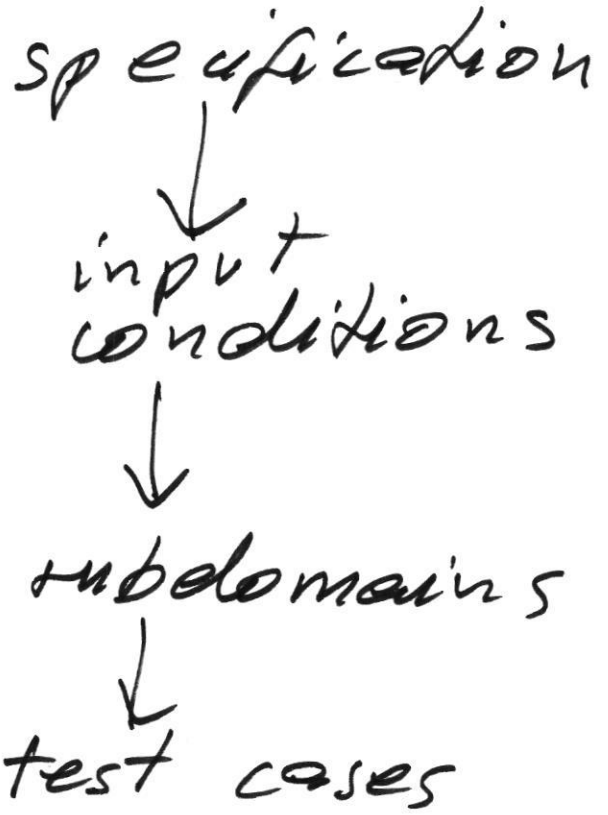
Equivalence-partitioning testing.



Identify subdomains (equivalence partitions)

A subdomain contains a set of inputs that have "common" property.

Test cases are selected from each subdomain.


specification
input conditions
subdomains
test cases

Given input condition

Two types of subdomains

1. valid subdomain

   contains inputs that satisfy the input condition

2. invalid subdomain

   contains inputs that violate the input condition

1. Normal Equivalence testing.

select test cases from valid subdomains

2. Robust Equivalence testing.

select test cases from invalid subdomains.

valid subdomain: several tests

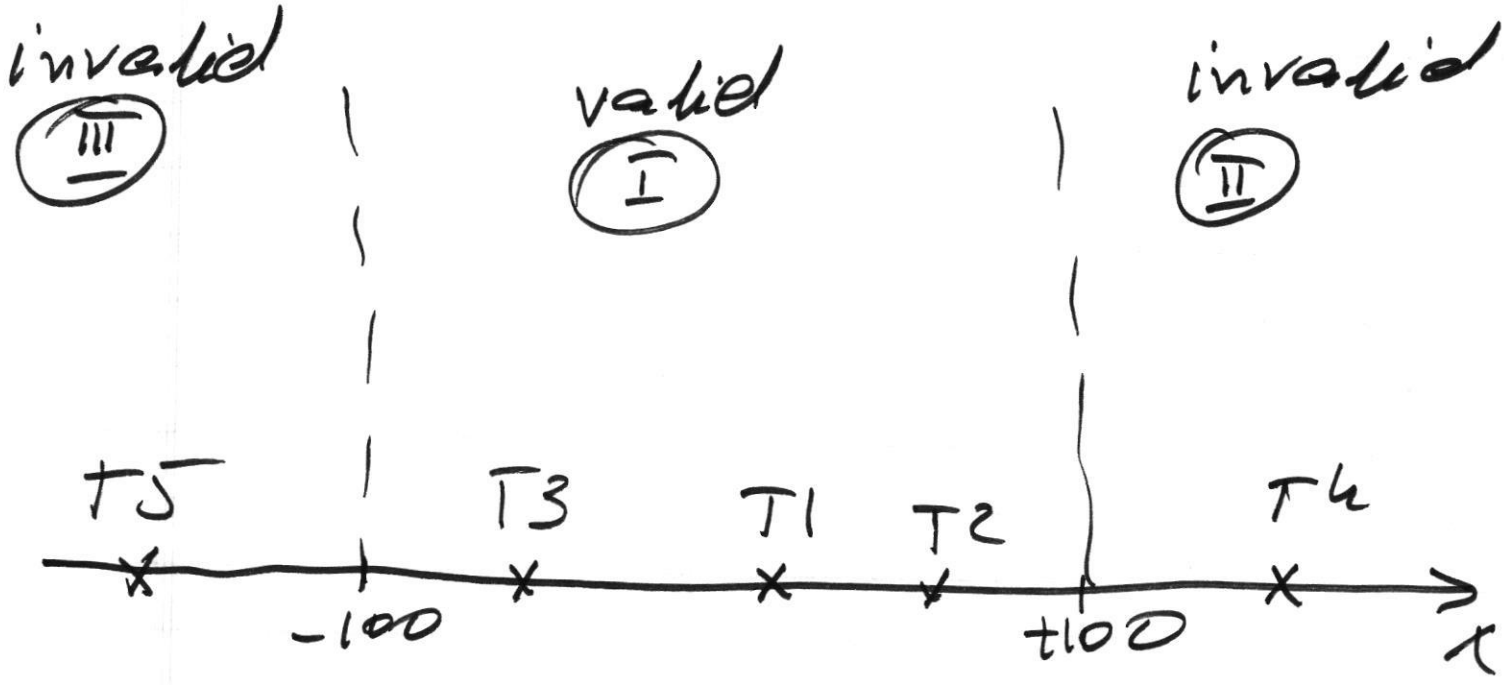
invalid —"— : one test

Input condition

range of values

input: x integer

input condition: $-100 \leq x \leq 100$



Normal Eq. tests

T1: 25
T2: 74
T3: -57

Robust Eq. tests

T4: x = 125
T5: x = -114

Input condition

The number of values

"An input file contains from 1 to 255 records"

I: valid subdomain

$1 \leq N \leq 255$

N: # of records

II: invalid
$N = 0$ empty file

III: invalid
$N > 255$

Normal Eq. tests	Robust Eq. tests
T1: N = 54 records	Th: N = 0 empty file
T2: N = 5 records	
T3: N = 205 -11-	TJ: N = 274 records

A set of values

- Type of vehicle must be bus, truck, taxi cab, passanger[4]

valid subdomain:

bus, truck, taxi cab, passenger

invalid subdomain:

anything else

Normal Eq. tests

T1: taxi cab
T2: bus

Robust Eq. test

T3: trailer

Another approach

4. valid subdomains
   1. bus
   2. truck
   3. taxi cab
   4. passanger
1 invalid subdomain
   anything else

---

1. Normal Eq. tests
   T1: bus
   T2: truck
   T3: taxi cab
   T4: passanger
2. Robust Eq. tests
   T5: trailer

"must be" condition

"The first character of input identifier must be a letter"

1. valid subdomain

   The 1st char. must be a letter

2. invalid subdomain

   1st char. is not a letter

1. Normal Eq tests	Robust Eq. test
T1: A25 T2: C47 T3: X247	T4: $A27

Integer array dimension declaration

int n(d[d]...)

n: symbolic name of the array.
d: dimension declaration.

symbolic name can have 1-6 ~~charact~~ letters or digits where the first character must be a letter.

The min # of dimensions: 1
The max # of dimensions: 7

The format of dim. declaration

[lb:]ub

lb: lower bound; ub: upper bound
The ~~b~~bound can be in a range

$-65,534 \leq lb, ub \leq 65,535$

if lb is not declared, lb=1

$ub \geq lb$

input condition	valid subdomain(s)	invalid subdomain
size of array name	1-6 (1) ✓	0 (2) ✓ >6 (3) ✓
name has letters/digits	has letter (4) ✓ has digit (5) ✓	sth else (6) ✓
array name starts with a letter	yes (7) ✓	no (8) ✓
# of dimensions	1-7 (9) ✓	0 (10) ✓ >7 (11) ✓
lower bound	-65,534 to +65,535 (12) ✓	<-65,534 (13) ✓ >65,535 (14) ✓
upper bound	-65,534 to +65,535 (15) ✓	<-65,534 (16) ✓ >65,535 (17) ✓
lower bound specified	yes (18) ✓ no (19) ✓	
ub to lb	ub > lb (20) ✓ ub = lb (21) ✓	ub < lb (22) ✓

Normal Eq. tests

T1: int A(1:5)

①, ④, ⑦, ⑨, ⑫, ⑮, ⑱, ⑳

T2: int A2(1:5) ⑤

T3: int A(5) ⑲

T4: int A(5:5) ㉑

Robust Eq. tests

T5: int C(1:5) (2)

T6: int abcdefgh(1:5) (3)

T7: int A$7(1:5) (6)

T8: int 5A(1:5) (8)

T9: int A() (10)

T10: int A(1:5,B,B,B,2:4,1:5,B,B,B) (11)

T11: int A(-65724:10) (13)

T12: int A(65572:10) (14) (22)

T13: int A(10:-65724) (16) (22)

T14: int A(1:67242) (17)

T15: int A(5:2) (22)

Advantages

* systematic approach
* high quality tests

Disadvantages

* subjective
* labor intensive
* test explosion.!!!