COM 3529

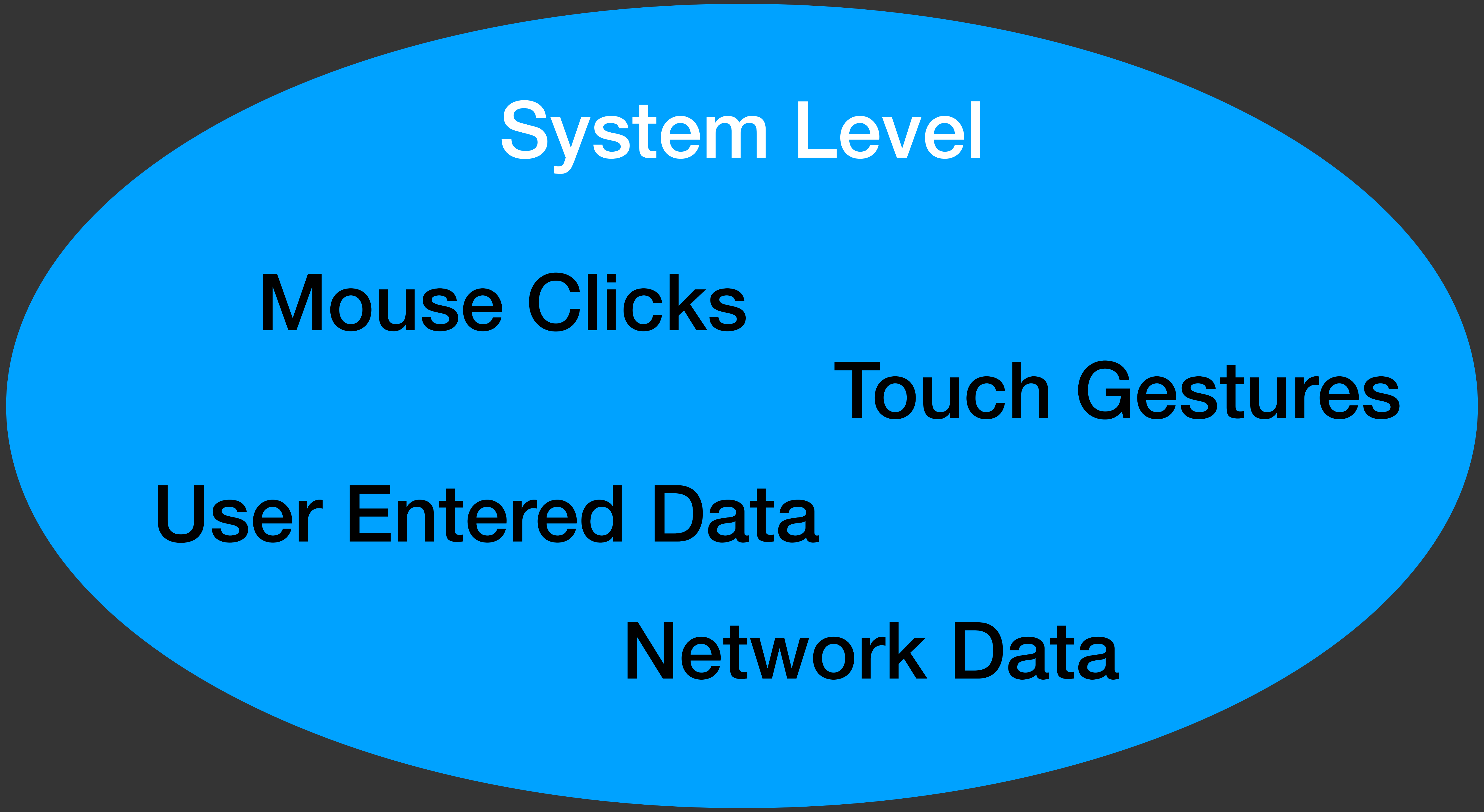
Software Testing & Analysis

Professor Phil McMinn

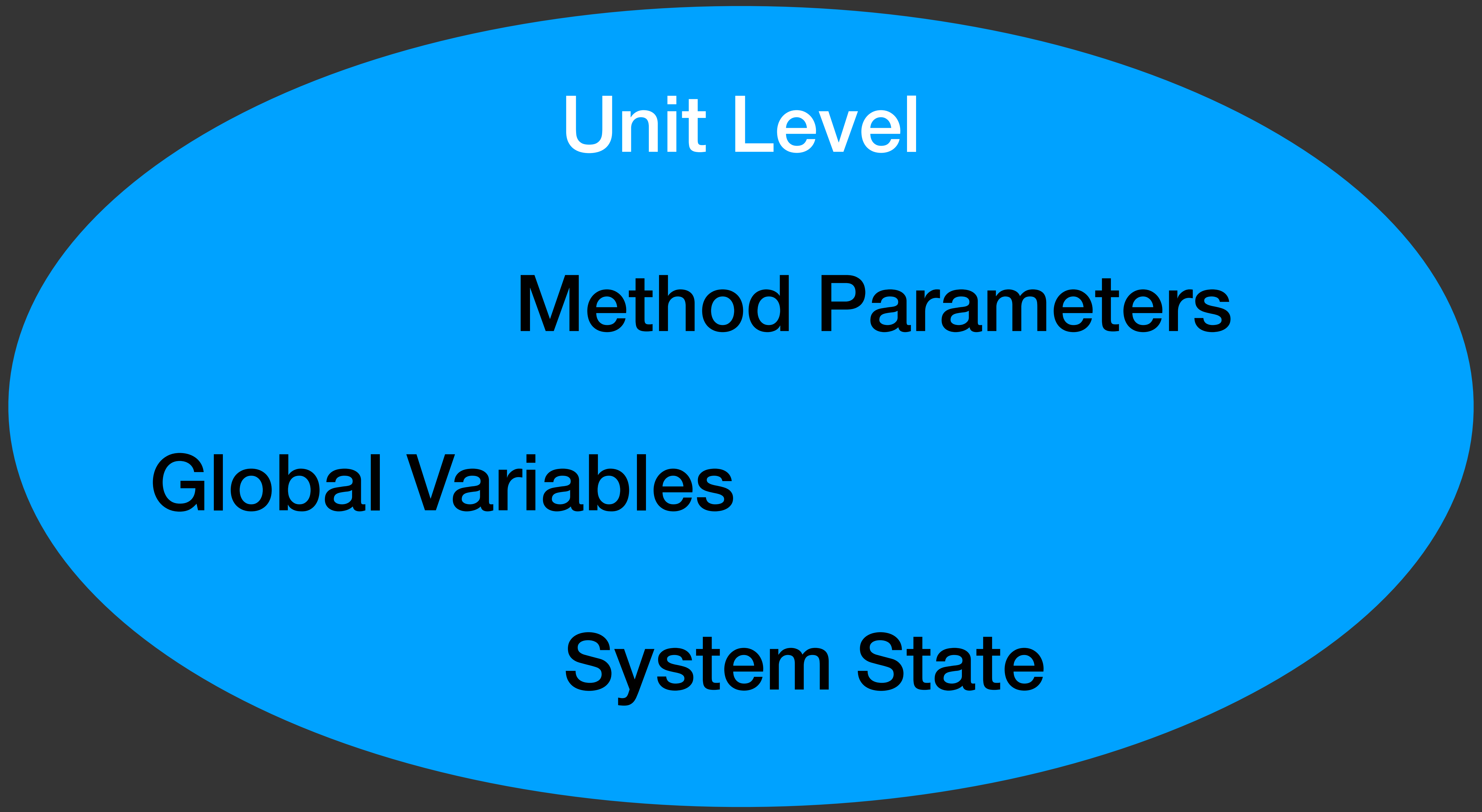
4.1 Grey-Box Coverage Criteria based on

Input Domain Analysis

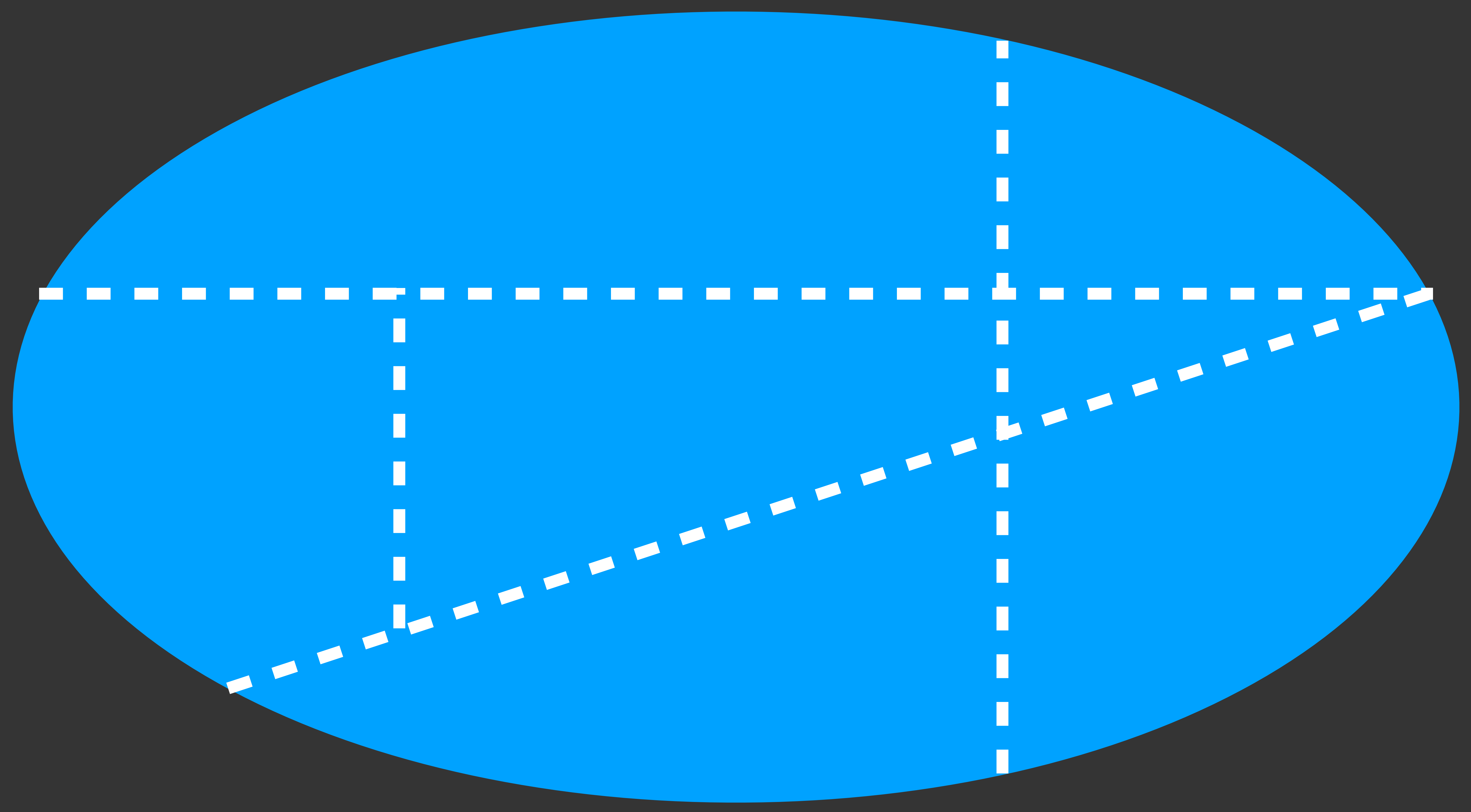
The Input Domain of a Program



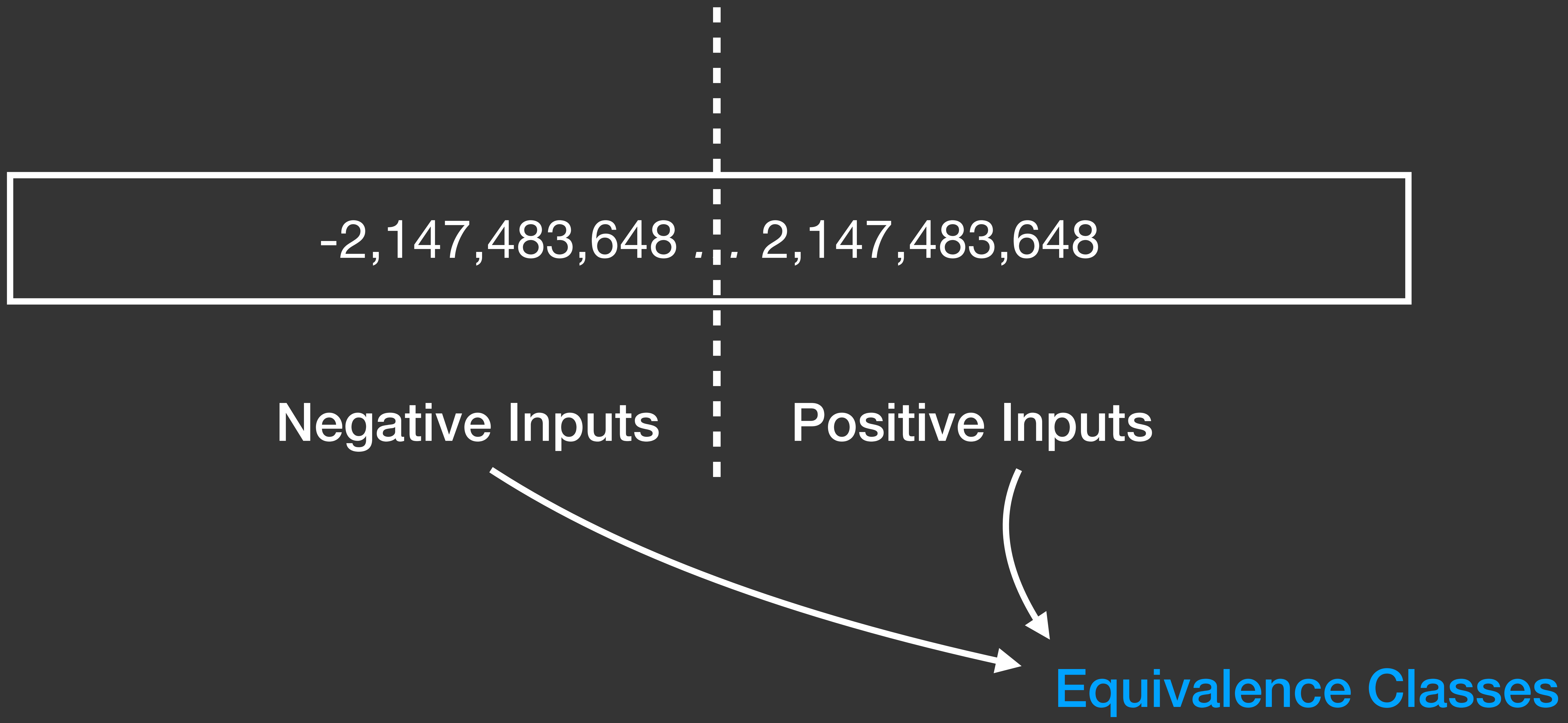
The Input Domain of a Program



The Input Domain of a Program



Input Domain for `isPositive(int n)`



Properties of an Input Domain Partition

1. The partition must cover the entire domain (i.e., it is **complete**)

2. The equivalence classes must not overlap (i.e., they are **disjoint**)

Input Domain for `isPositive(int n)`

-2,147,483,648 ... 2,147,483,648

Negative Inputs

Positive Inputs

Equivalence Classes

Input Domain for `isPositive(int n)`

-2,147,483,648 … -1	0	1 … 2,147,483,648
e_1	e_2	e_3

Input Domain for `isPositive(int n)`

-2,147,483,648 … -1	0	1 … 2,147,483,648
e_1	e_2	e_3

Test Case	Equivalence Class	Example Input	Expected Outcome
1	Less than zero	-100	`false`
2	Zero	0	`false`
3	Greater than zero	100	`true`

Boundary Value Analysis

-2,147,483,648 … -1	0	1	2 … 2,147,483,648
e_1	e_2	e_3	e_4

```
if (n > 1) return true   😢

if (n >= 1) return true  🤩
```

Characteristics

- Whether `elements` is `null` (true, false)
- The *size* of `elements` (0, 1, greater than 1)
- The *initial order* of `elements` (ascending, descending, arbitrary)

The *initial order* of `elements` (ascending, descending, arbitrary)

e_1 e_2 e_3

???

Length of `elements`
is zero or one

The *initial order* of `elements`

- Sorted in ascending order (true, false)
- Sorted in descending order (true, false)

Applying Input Domain Analysis

1. Identify the Functions to Test
2. Identify the Input Domain
3. Identify Characteristics
    - Interface-Based Input Domain Modelling
    - Functionality-Based Input Domain Modelling
4. Identify Equivalence Classes

Identifying Equivalence Classes

- Valid sets of values
- Invalid sets of values
- Normal sets of values
- Abnormal sets of values
- Boundary values
- Special values

Ensure the partitioning is valid:

- Missing sets of values
- Overlapping sets of values

Case Study: The Triangle Classification Program

	Characteristic	Equivalence Classes		
		e_1	e_2	e_3
1	Relation of `side1` to 0	greater than 0	equal to 0	less than 0
2	Relation of `side2` to 0	greater than 0	equal to 0	less than 0
3	Relation of `side3` to 0	greater than 0	equal to 0	less than 0

Case Study: The Triangle Classification Program

	Characteristic	Equivalence Classes			
		e_1	e_2	e_3	e_4
1	Length of `side1`	greater than 1	equal to 1	equal to 0	less than 0
2	Length of `side2`	greater than 1	equal to 1	equal to 0	less than 0
3	Length of `side3`	greater than 1	equal to 1	equal to 0	less than 0

Case Study: The Triangle Classification Program

		Equivalence Classes			
Partition	**Characteristic**	e_1	e_2	e_3	e_4
1	Geometric Classification	scalene	isosceles	equilateral	invalid

Case Study: The Triangle Classification Program

	Characteristic	Equivalence Classes			
		e_1	e_2	e_3	e_4
1	Geometric Classification	scalene	isosceles, not equilateral	equilateral	invalid

Case Study: The Triangle Classification Program

Param	e_1	e_2	e_3	e_4
Triangle	(4, 5, 6)	(3, 3, 4)	(3, 3, 3)	(3, 4, 8)

Case Study: The Triangle Classification Program

	Characteristic	Equivalence Classes	
		e_1	e_2
1	Triangle is Scalene	true	false
2	Triangle is Isosceles	true	false
3	Triangle is Equilateral	true	false
4	Triangle is Invalid	true	false

Combination Coverage Criteria

- All Combinations Coverage
- Each Choice Coverage
- Pair-Wise Coverage
- T-Wise Coverage
- Base Choice Coverage
- Multiple Base Choice Coverage

Constraints Among Partitions

- Cannot be combined
- Must be combined

Final Thoughts

- Using more than one input domain model
- Checking the input domain model