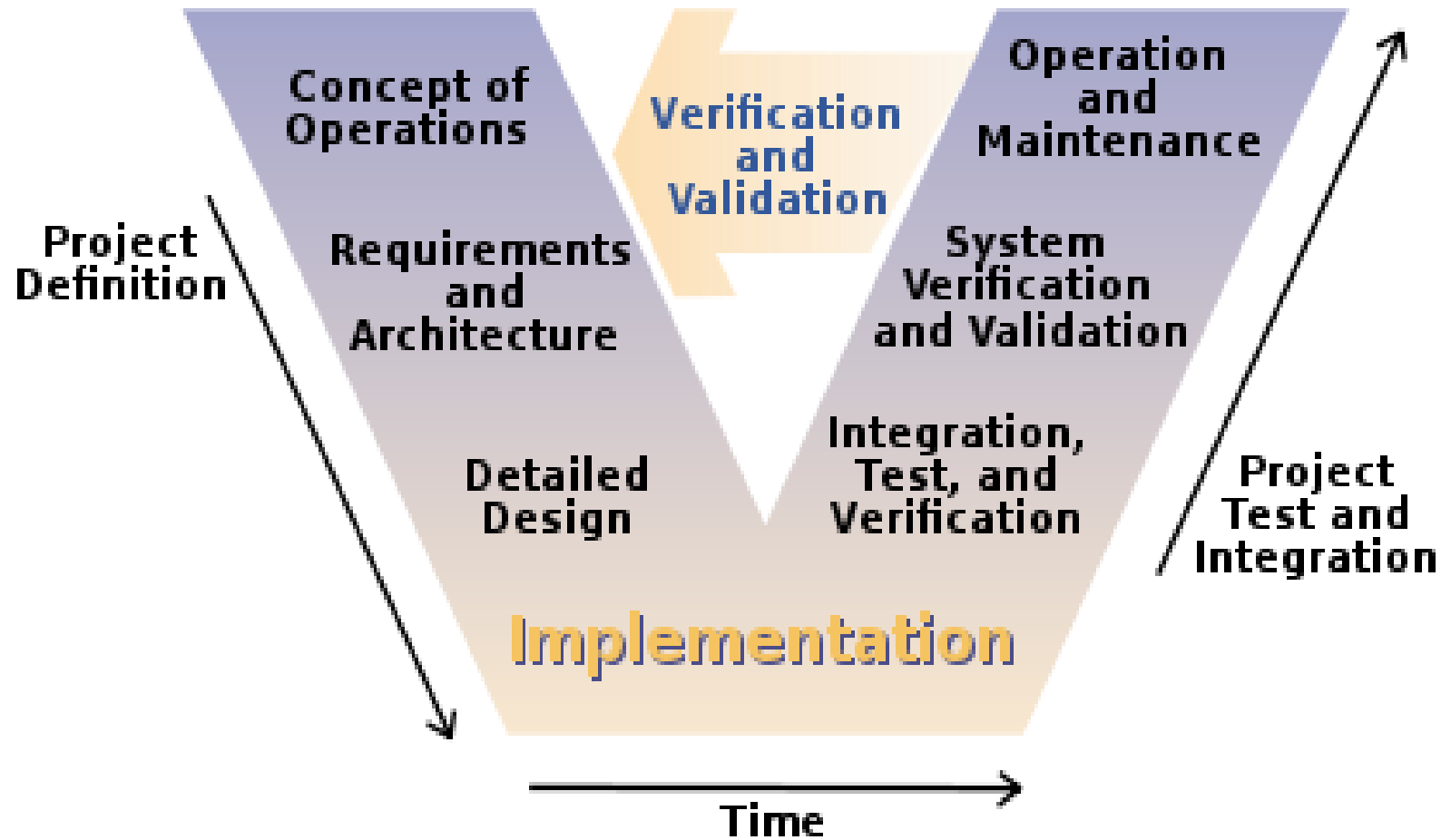


Software Testing

Peter Levinsky IT Roskilde

05.02.2024

The V Model



Program testing goals

- To demonstrate to the developer and the customer that the software **meets its requirements**.
=> leads to **validation testing**
- To discover situations in which the behavior of the software is incorrect, undesirable or does **not conform to its specification**.
=> leads to **defect testing**

Verification vs validation

- **Verification:** (testing)

"Are we building the product right".

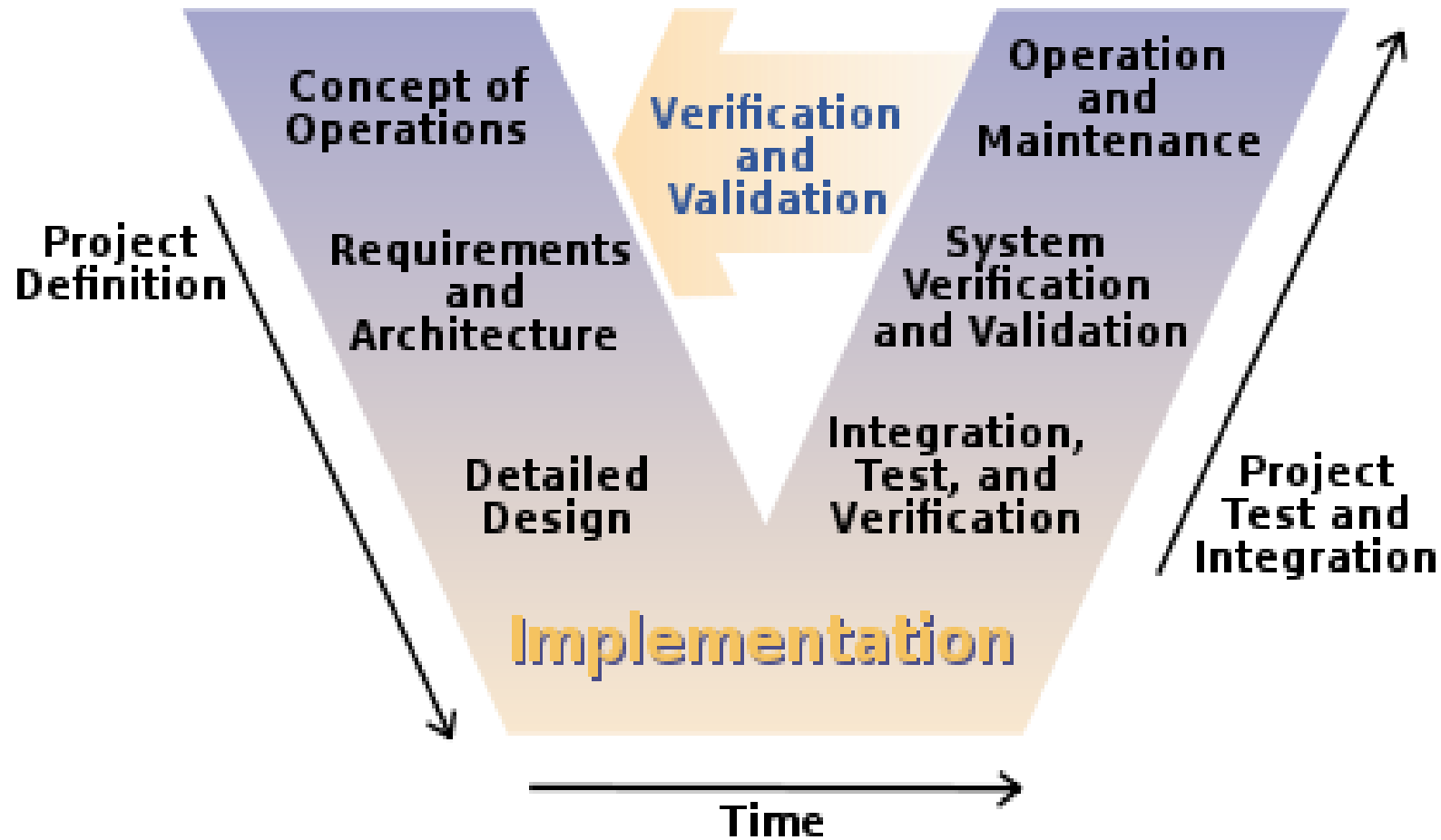
- The software should conform to its specification.

- **Validation:** (checking)

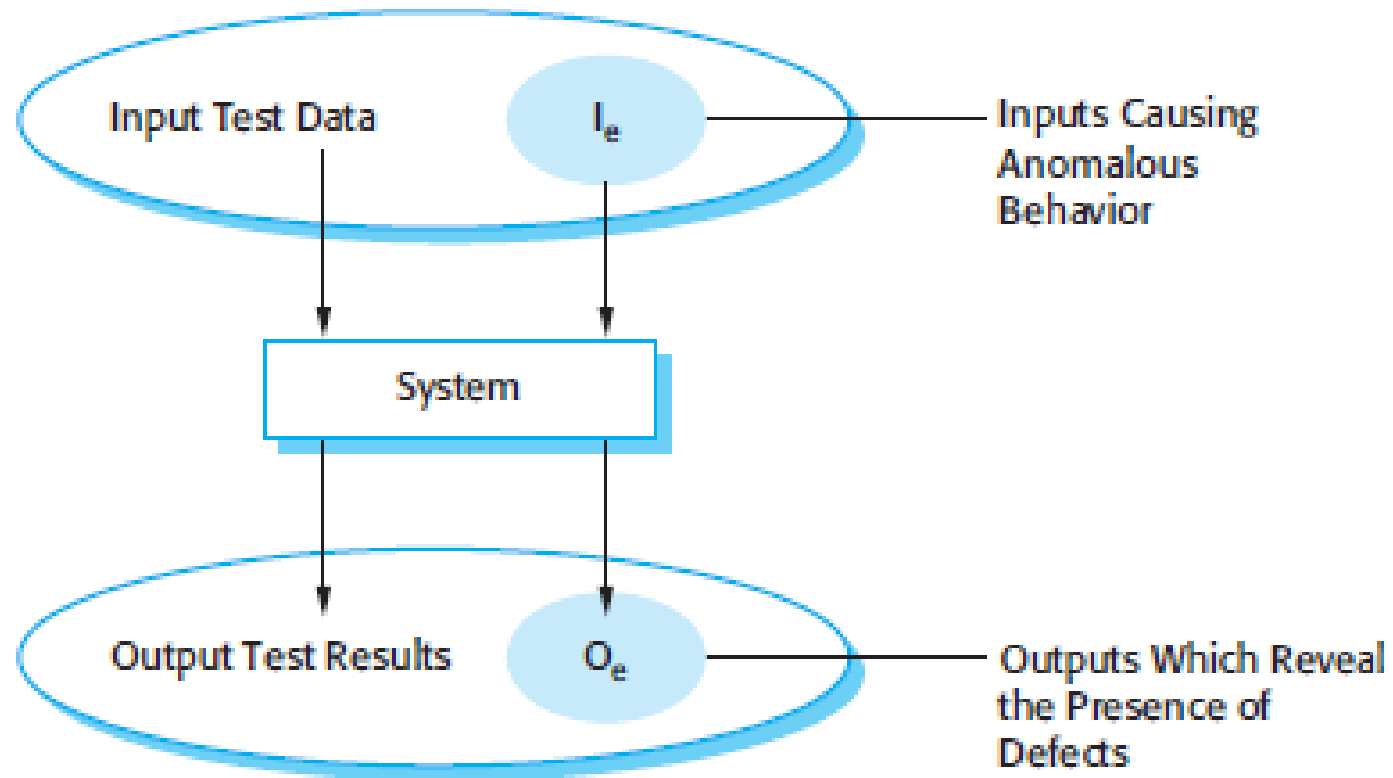
"Are we building the right product".

- The software should do what the user really requires.

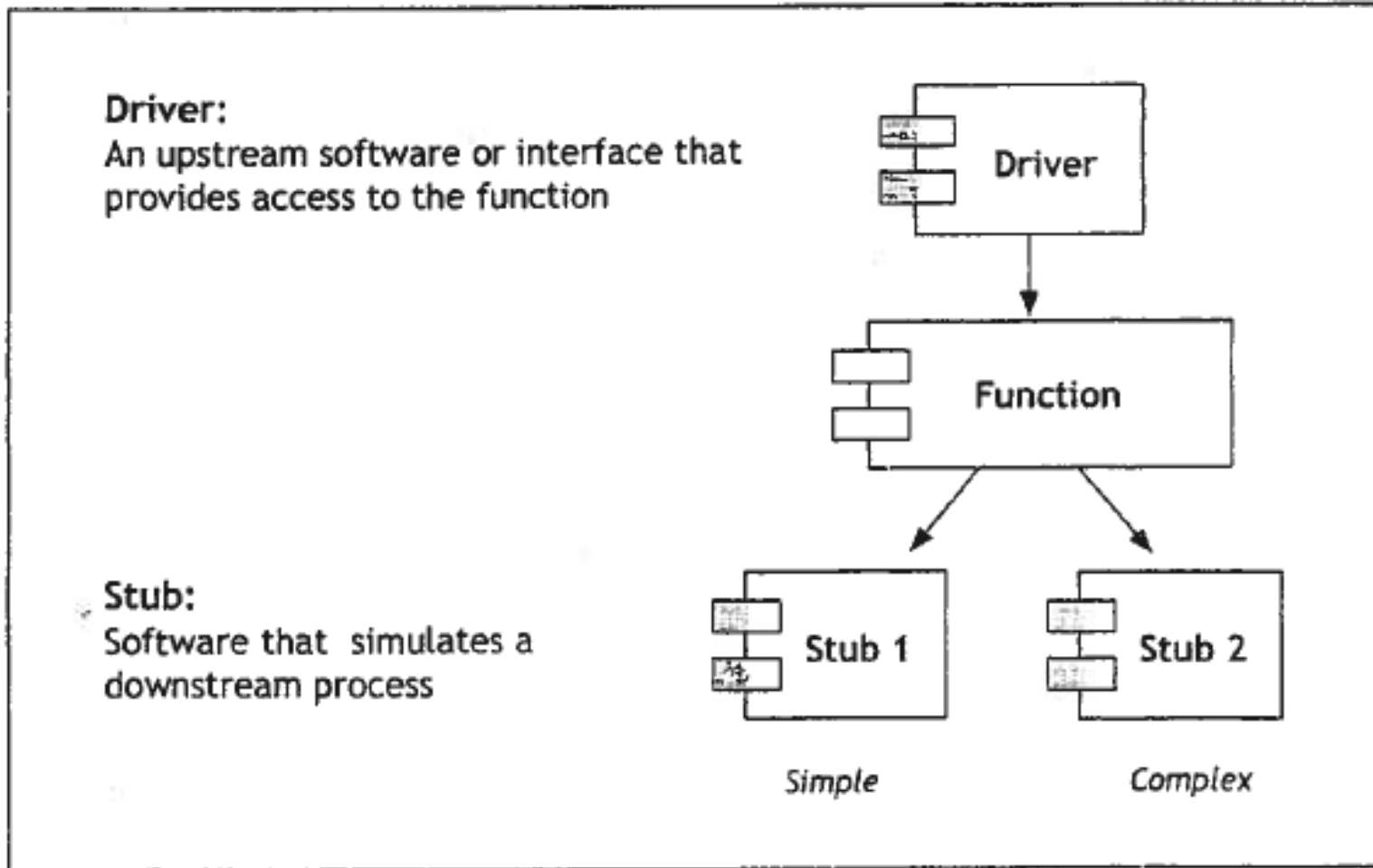
The V Model



Testing - principles



Set up Test



Different levels of testing

related to the V-model

- Validation of the concepts and requirements
 - e.g. Are the domain model right? The use stories? (the users)
- Validation of the design
 - e.g. design class diagrams and design sequence diagrams
(Reviews, Technical walkthrough by the project team)
- Component Verification
 - e.g. **unit test** and test cases (implementer)
- System and integration Verification
 - e.g. system/integration test
- Operation Verification
 - e.g. acceptance test

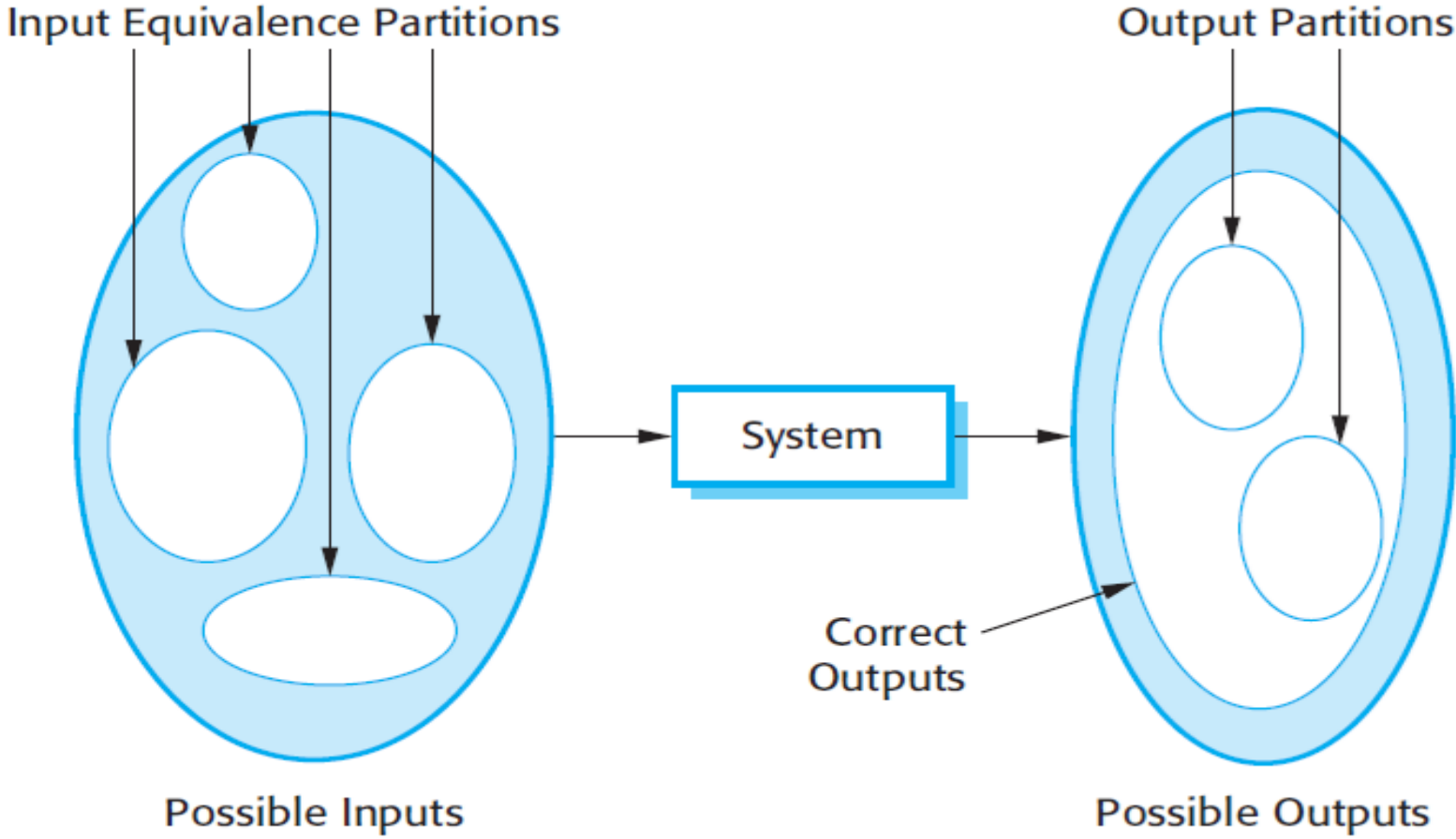
Black Box & White Box test

- Black box
 - Look at methods (system part) as a closed box
 - Know only interface
- White box
 - Look inside the methods (system part)
 - Look at all possible path through the methods

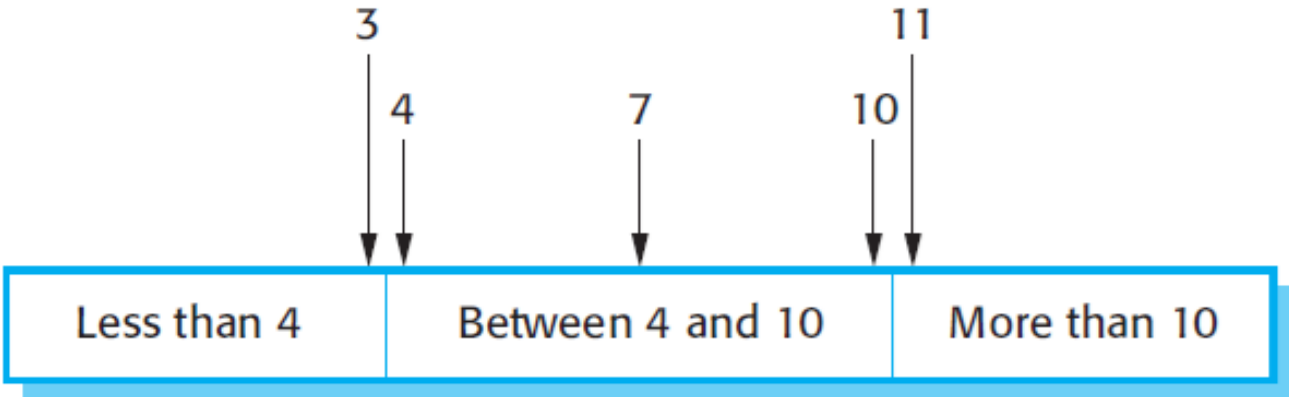
Black box testing

- The system code is 'unknown' -> a black box
- Look only at the methods signatures
- **Testing all kind of possible input and output**
- In C# create a Unit Test

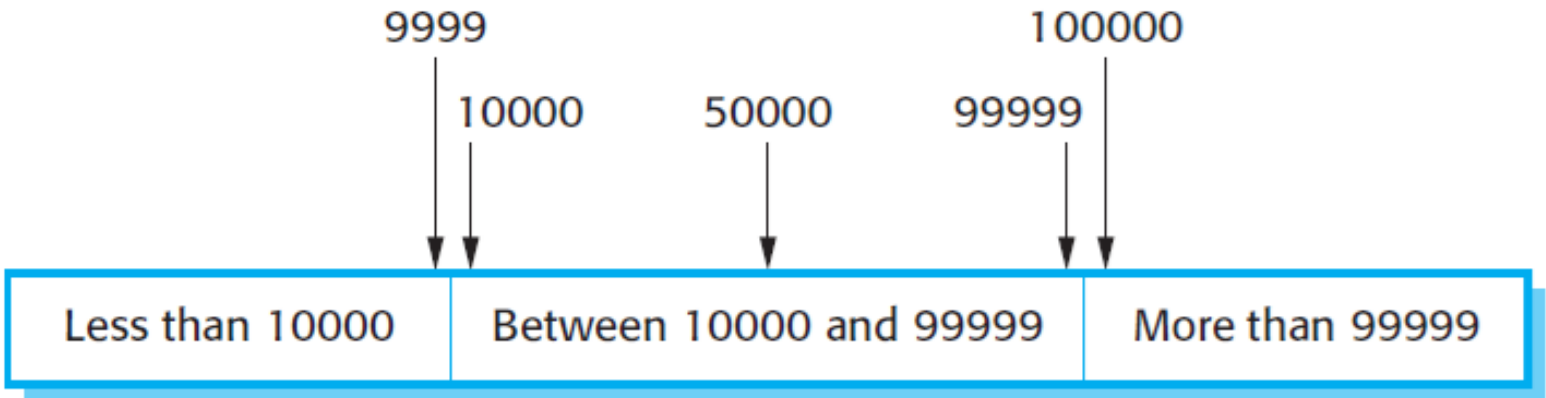
Equivalence partitioning



Equivalence partitions



Number of Input Values



Input Values

Unit test in C# - Visual Studio

- Create a test unit project,
- Add reference to the project,
- Remember to have the class to be tested **public**.
- Make a test method for each test case

What can we do in in a test unit

- **Annotations**
- [TestClass] : set up the test
- [TestMethod] : This is a test method to be run
- [TestInitialize] : Run this before each test method
- [ClassInitialize] : Run this before the test starts
- [DataRow (x,y)] : give test method parameter

- **Testing verification**
- Assert.AreEqual(expected, actual)
- Assert.IsTrue(actual)
- Assert.ThrowsException<XXException>(()=> -- act --)

Practice - Test case in UNIT test

- **Arrange**

- Set up the test (part could be in test TestInitialize)
- Give all input the testing data
- Give expected data the expected values

- **Act**

- Run the method

- **Assert**

- Check if the test have succeed

Example

```
[TestMethod]
public void TestMethod1()
{
    // Arrange
    Person p = new Person("SomeName", "SomePhone", "SomeAddress");
    String expectedPhone = "SomePhone";

    //Act
    String actualPhone = p.Phone; // do the action - here just read a property

    //Assert
    Assert.AreEqual(expectedPhone, actualPhone);
}
```