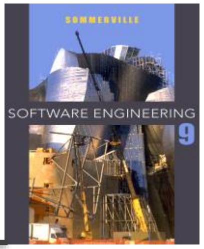
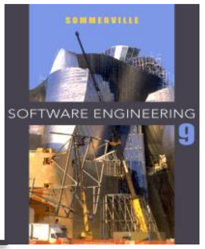
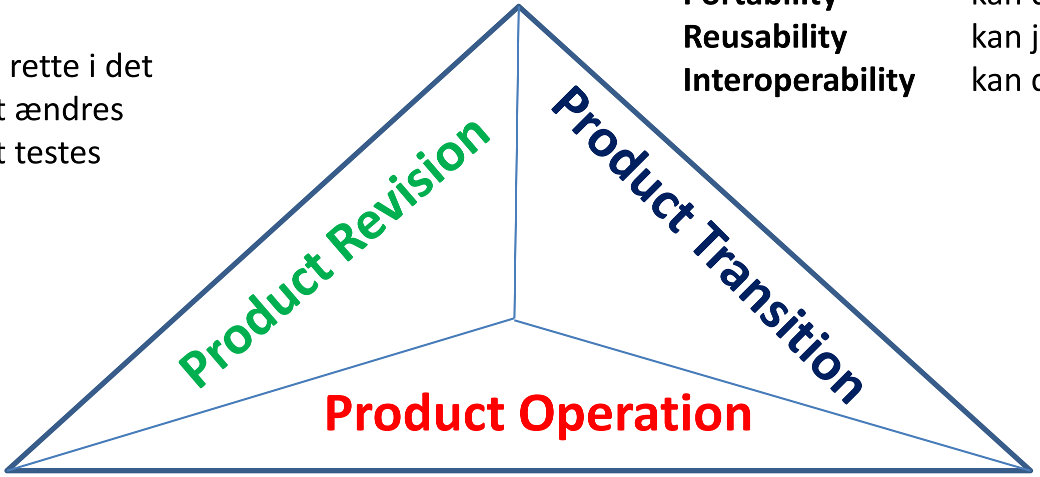


Software Testing

Kvalitets faktorer



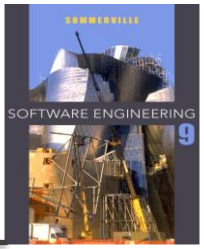


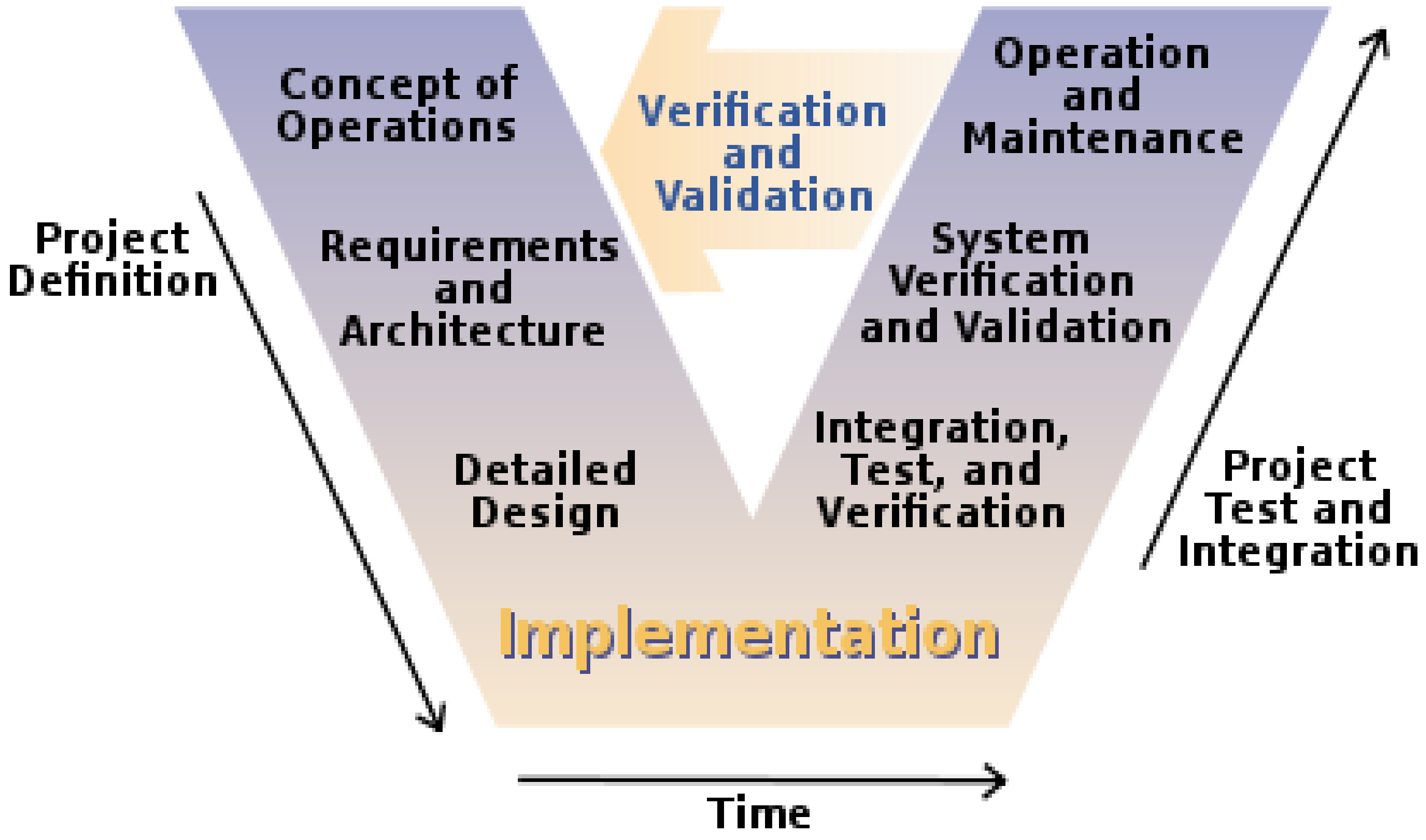
Maintability	Kan jeg rette i det
Flexsability	Kan det ændres
Testability	Kan det testes

Portability	kan det køre på andre platforme
Reusability	kan jeg genbruge dele af SW
Interoperability	kan det kobles til andre systemer

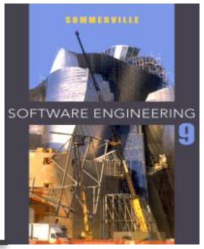
Correctness	Gør vi det rigtigt
Reliability	Gør vi det nøjagtigt hele tiden
Efficiency	Kører det optimalt på min HW
Integrity	Er det sikkert
Usability	Kan jeg køre det (brugervenligt)

V model


SOMMERVILLE
SOFTWARE ENGINEERING
9


Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

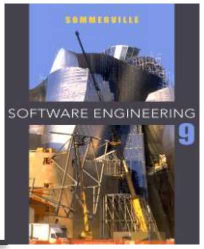
Program testing goals



- ✧ 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



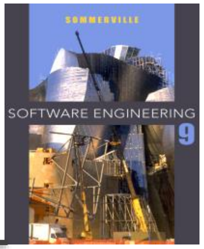
✧ **Verification**: (testing)
"Are we building the product right”.

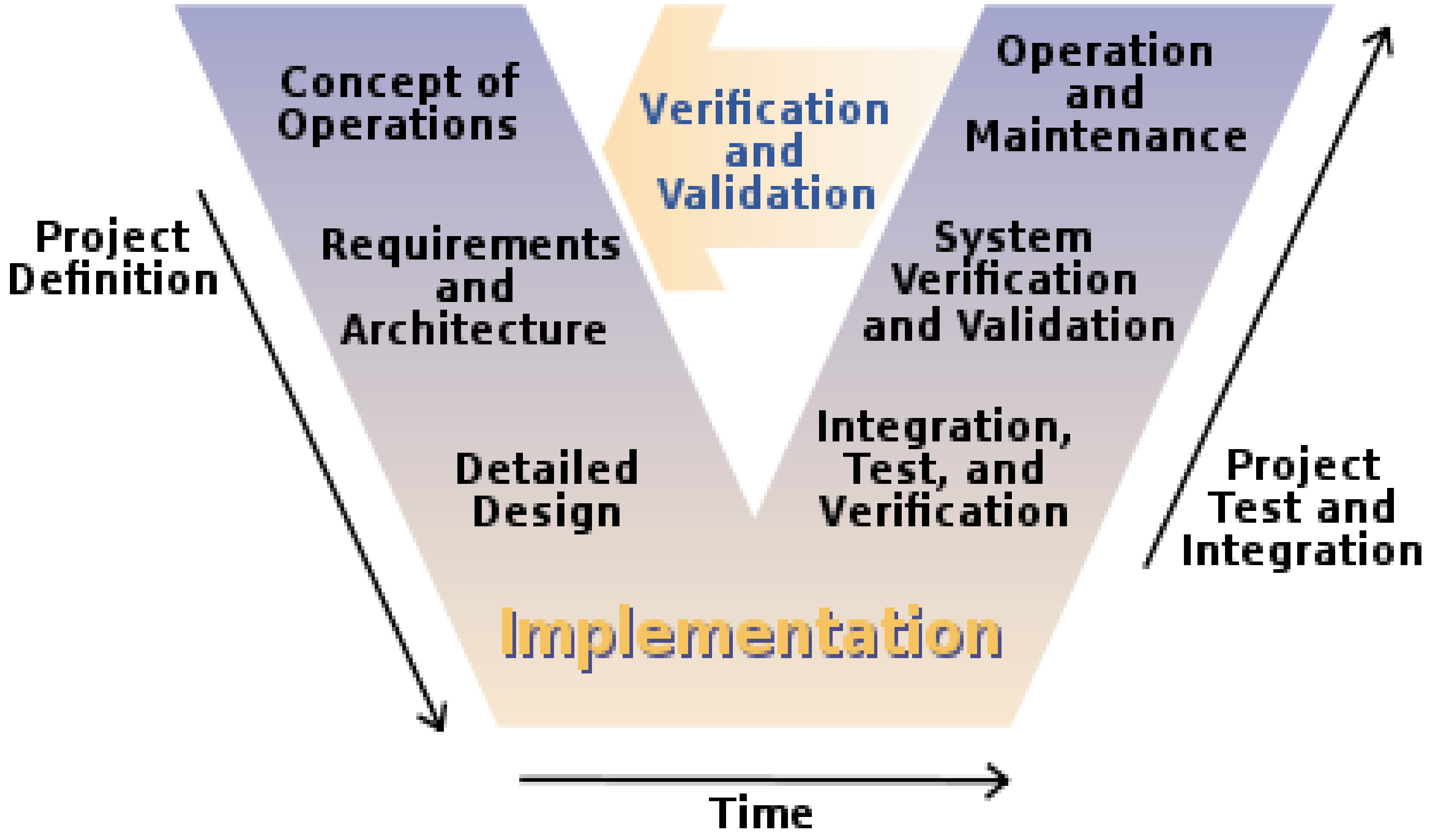
- The software should conform to its specification.

✧ **Validation**: (checking)
"Are we building the right product”.

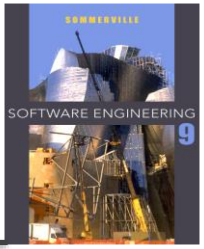
- The software should do what the user really requires.

V model



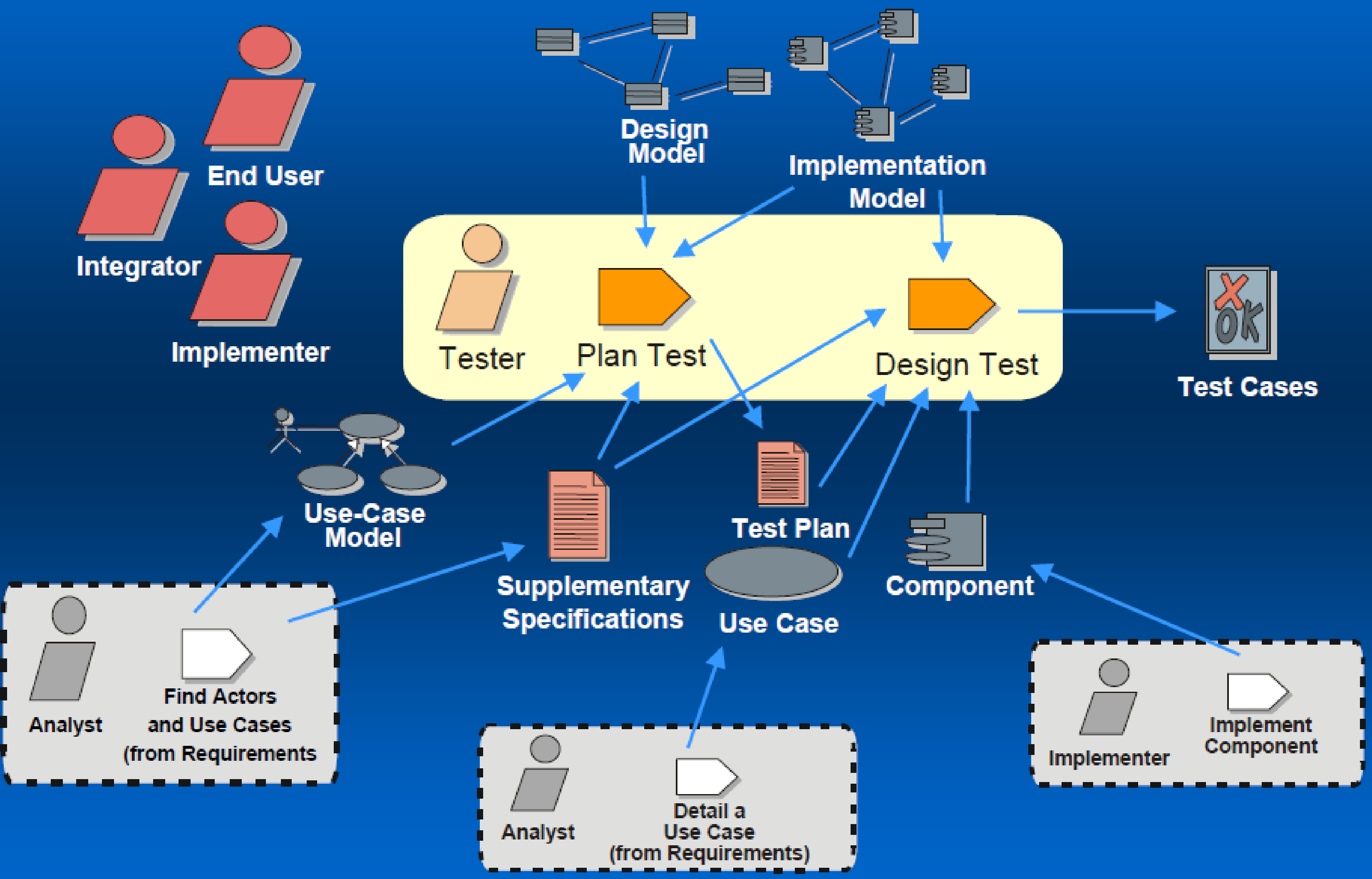

Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

Different levels of testing
related to the V-model



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

Test Planning and Design Activity



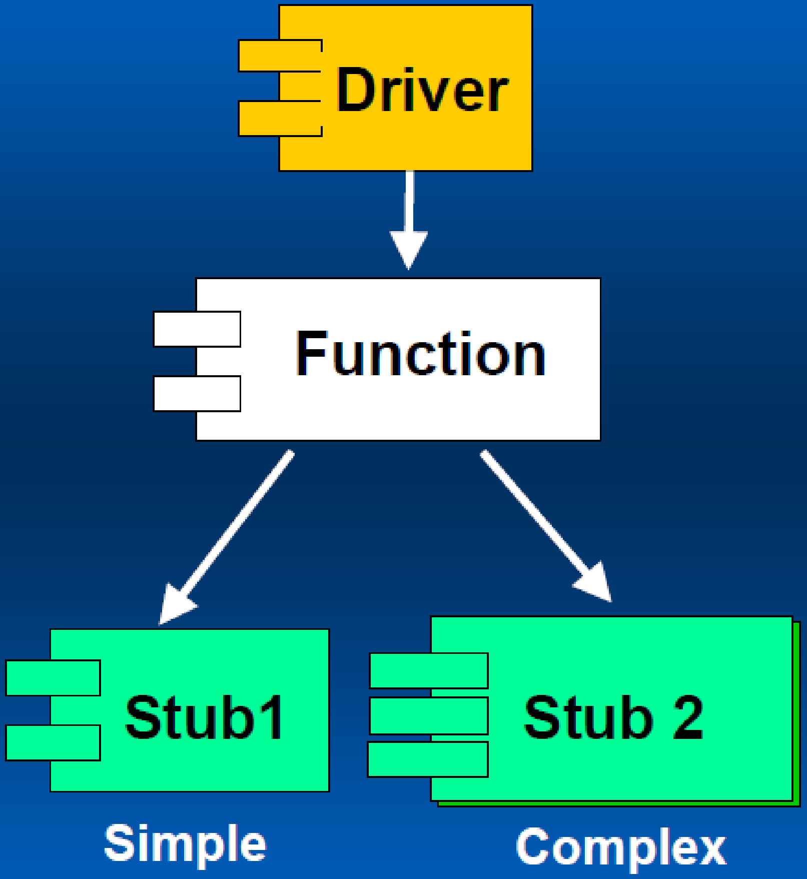
The Functions of the Stubs & Drivers

Driver:

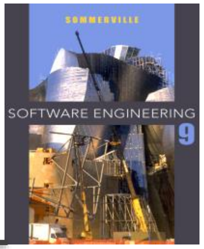
An upstream software or interface that provides access to the Function

Stub:

Software that simulates a downstream process

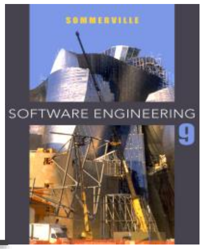


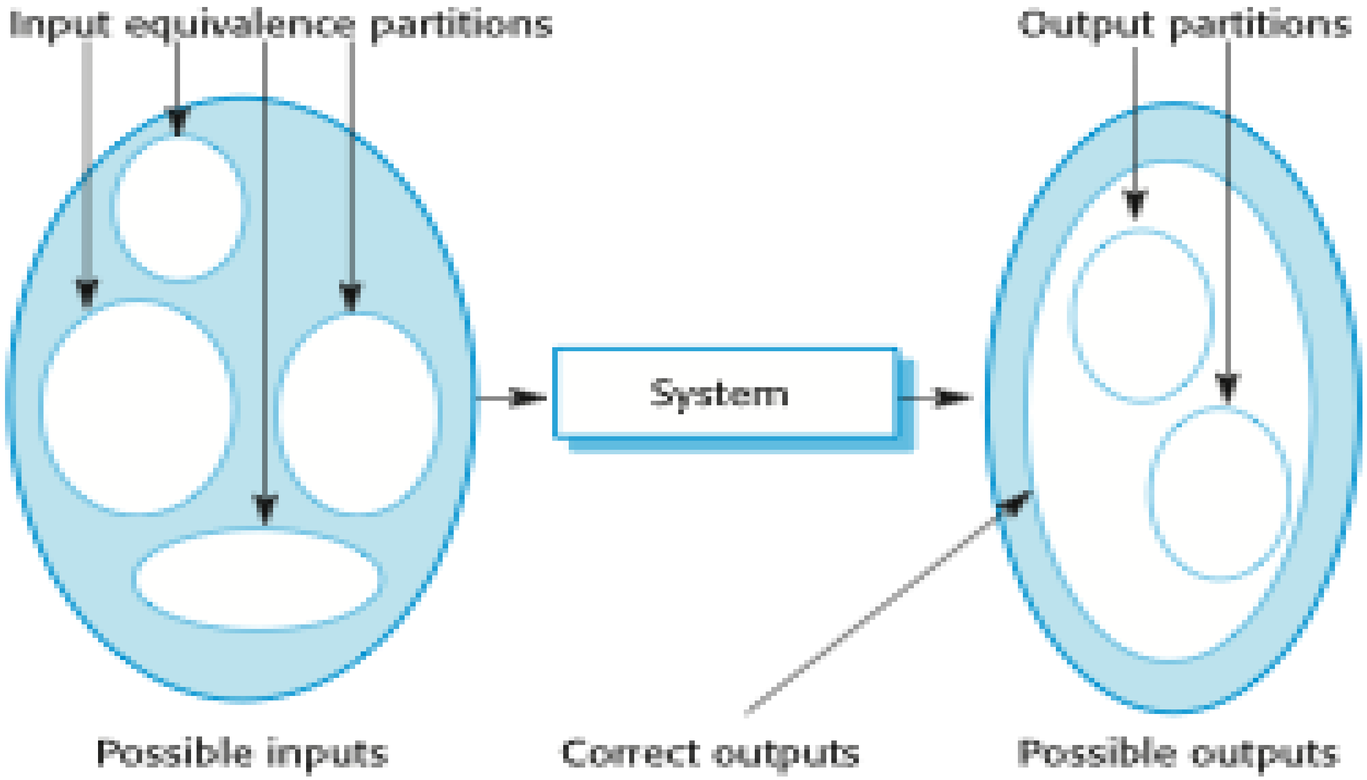
Black box testing



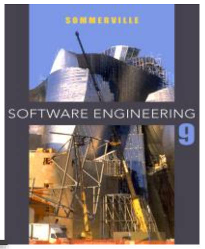
- ✧ The system code is ‘unknown’ -> a black box
- ✧ Look only at the methods signatures
- ✧ **Testing all kind of possible input and output**
- ✧ In C# create a Unit Test

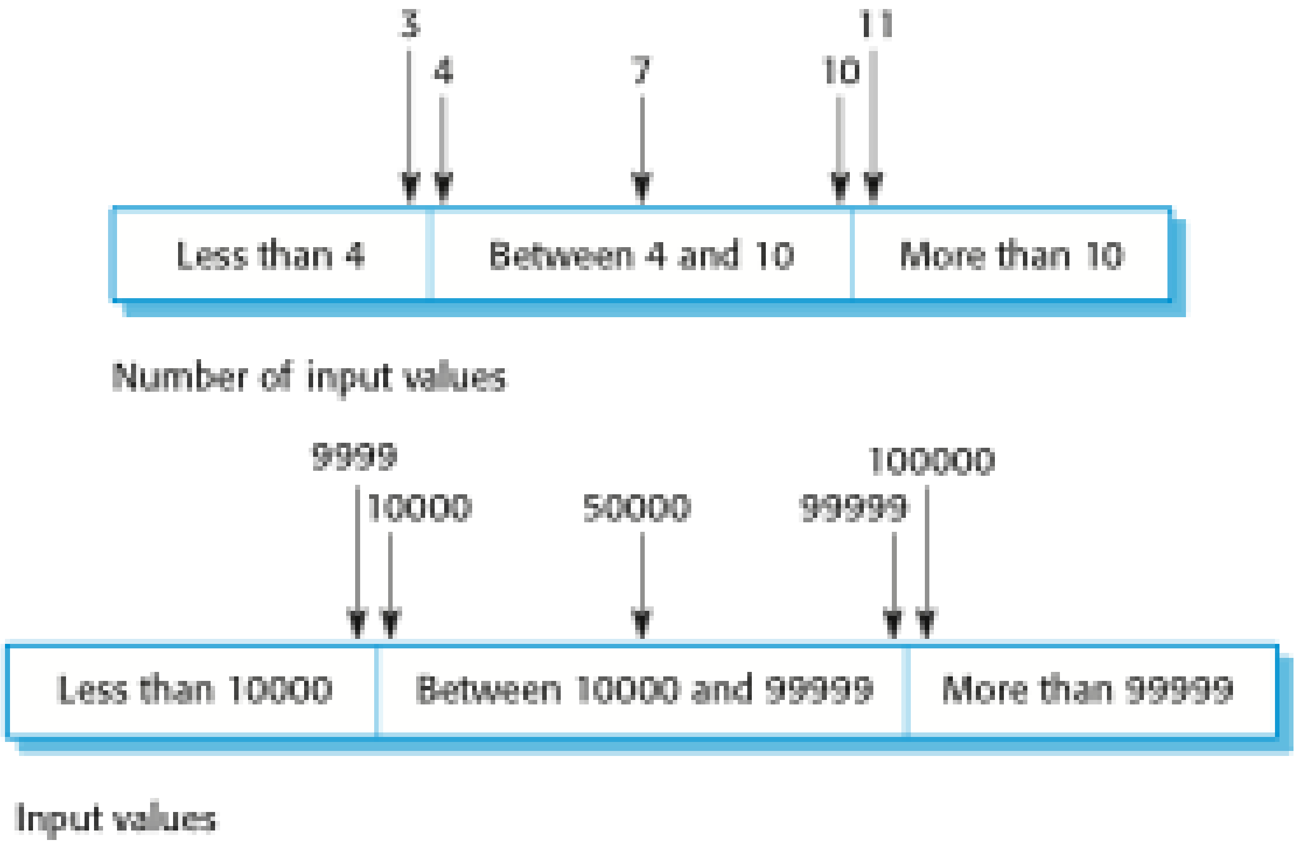
Equivalence partitioning



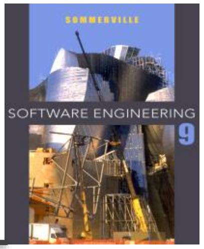


Equivalence partitions



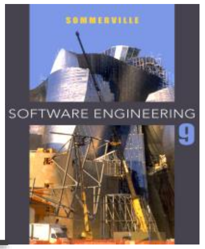


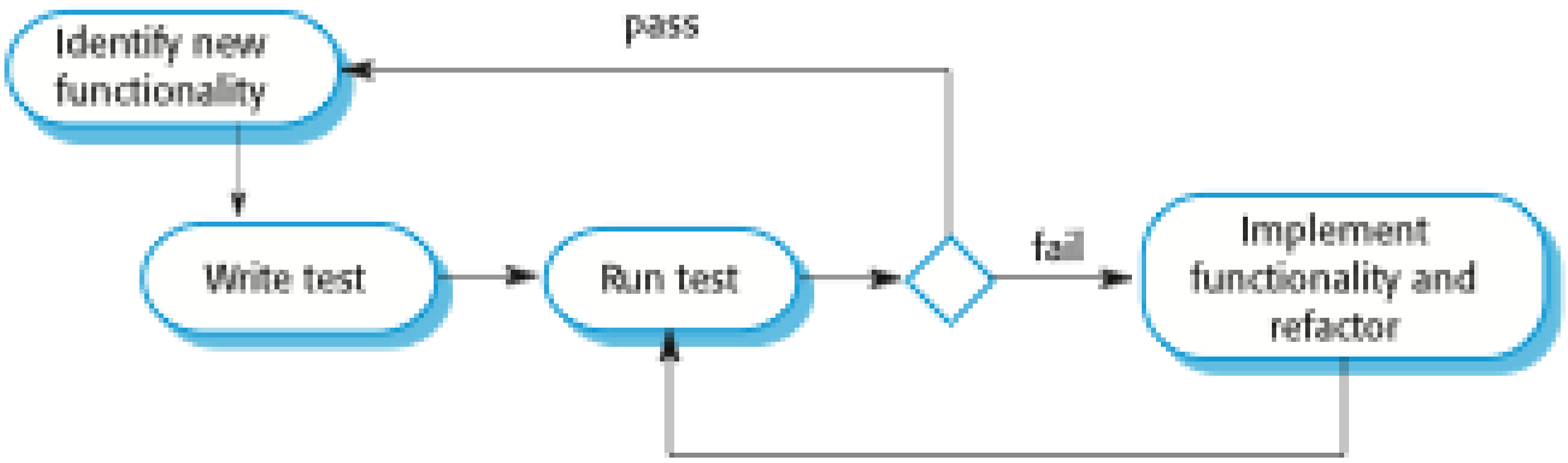
Test-driven development

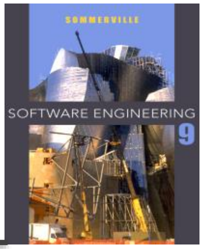


- ✧ Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- ✧ **Tests are written before code** and ‘passing’ the tests is the critical driver of development.
- ✧ You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- ✧ TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.

Test-driven development




Identify new functionality
pass
Write test
Run test
fail
Implement functionality and refactor



Unit test in c#

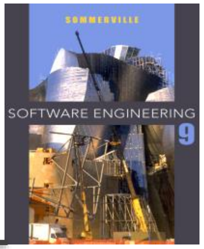
✧ Console Programs

- Create a test unit project,
- Add reference to the project,
- Remember to have the class to be tested **public.**
  (in resharper set cursor at the class – right click choose generate unit test)
- Make a test method for each test case

✧ App Programs

- Create a unit test app (universal windows),
- Add reference to the project,
- Remember to have the class to be tested **public.**
  (in resharper set cursor at the class – right click choose generate unit test)
- Make a test method for each test case

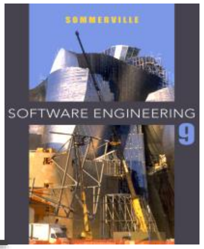
What can we do in in a test unit



- ✧ **Annotations**
- ✧ [TestClass] : set up the test
- ✧ [TestMethod] : This is a test method to be run
- ✧ [TestInitialize] : Run this before each test method

- ✧ **Testing validation**
- ✧ Assert.AreEqual(expected, actual)
- ✧ Assert.IsTrue(actual)

Special for exception



✧ Console programs

- Make try – catch : NB! The catch is ok = green

-
  ```
  Try{
        Call method;
        Assert.Fail();
    Catch{
        //Ok
    }
  ```

- Alternative make an annotation [ExpectedException typeof (xxxException)]

✧ App programs

- Assert.ThrowsException<xxxException>(() => call method)