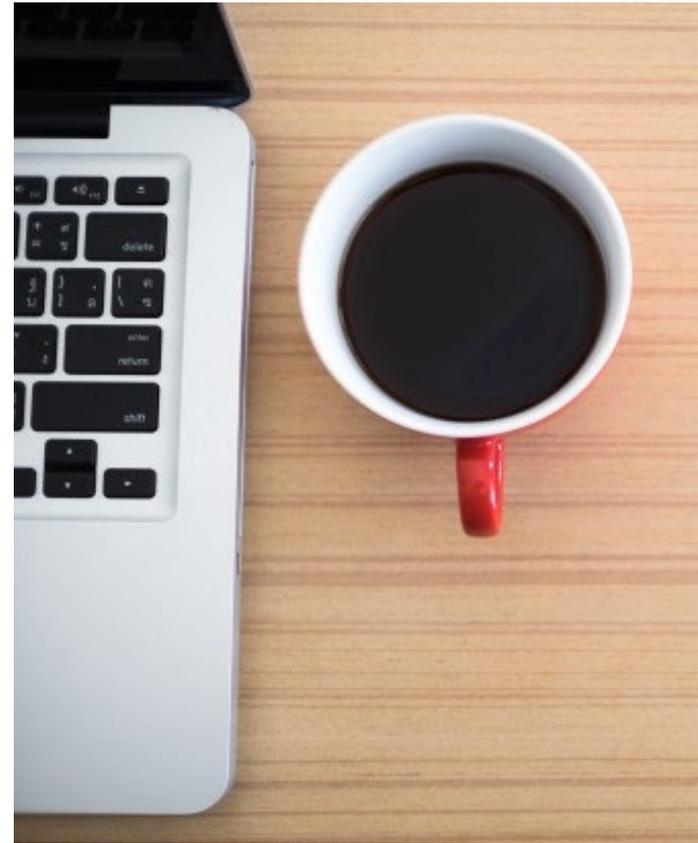


Course Topics

- ~~Introduction~~
- ~~Software Process Models~~
- ~~Requirements Engineering~~
- ~~Modeling~~
- ~~User Interface Design~~
- ~~Software Construction Techniques~~
 - Testing
 - Refactoring
 - Project Management
 - Ethical Issues

Lecture Objectives

- ✓ Software testing motivation
- ✓ What is a software testing?
- ✓ Why software fail?
- ✓ Types of testing
- ✓ Levels of testing



Software Testing Motivation



- British Airways
 - For the sixth time in a year, British Airways faced a massive global IT failure which led to the airline cancelling all flights from Heathrow and Gatwick in May 2017.
 - The IT failure affected over 1,000 flights, British Airways call centres, the website and mobile app.

Software Testing Motivation



■ Patriot Missile Error

- In February of 1991, a U.S. Patriot missile defence system in Saudi Arabia, failed to detect an attack on an Army barracks.
 - On the day of the incident, the system had been operating for more than 100 hours, and the inaccuracy was serious enough to cause the system to look in the wrong place for the incoming missile.
 - Army officials had fixed the software to improve the Patriot system's accuracy. That modified software reached the base the day **after** the attack.
- 

What is Software Testing?

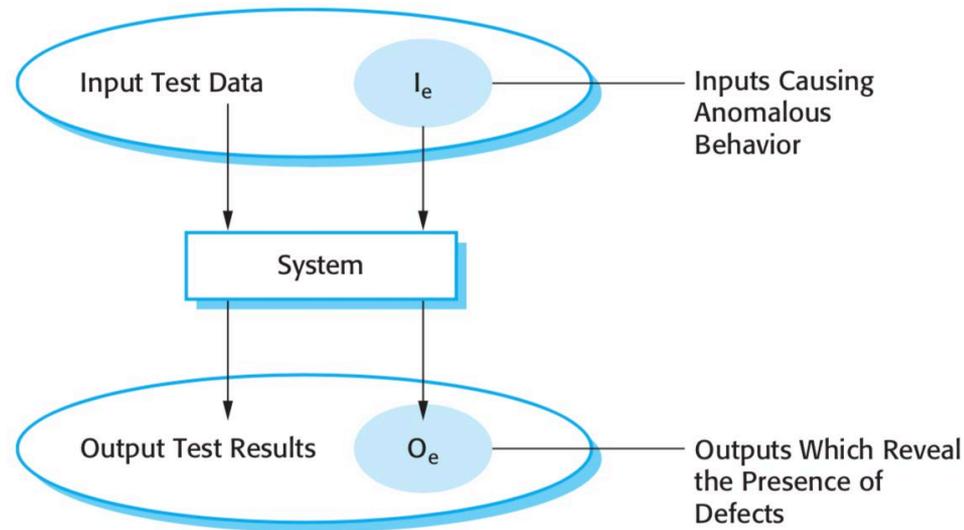
- **Testing** is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.

- The testing process has two distinct **goals**:
 1. Demonstrate to the developer and the customer that the software meets its requirements.
 2. To discover situations in which the behavior of the software is incorrect, undesirable. ← Defect Testing

What is Software Testing?

■ Defect testing

- Priority in defect testing is to find those inputs in the set I_e because these reveal problems with the system.





“Testing can only show the presence of errors, not their absence”

Dijkstra et al .

What is Software Testing?



- **Correctness** of software with respect to requirements or intent
 - **Performance** of software under various conditions
 - **Robustness** of software, its ability to handle erroneous input and unanticipated conditions
 - **Installation** and other facets of a software release
- 

Verification and Validation (V&V)



- Testing is part of a broader process of software **verification** and **validation** (V & V).
- **Verification**
 - The aim of is to check that the software meets its stated functional and non-functional requirements.
- **Validation**
 - Is a more general process, with the aim of ensuring that the software meets the customer's expectations.

Basic Definitions

■ Failure

- There is a deviation of the observed behavior of a program or a system from its specification.

■ Fault

- An incorrect step, process or data definition.

Example: for any integer n , $\text{square}(n) = n * n$.

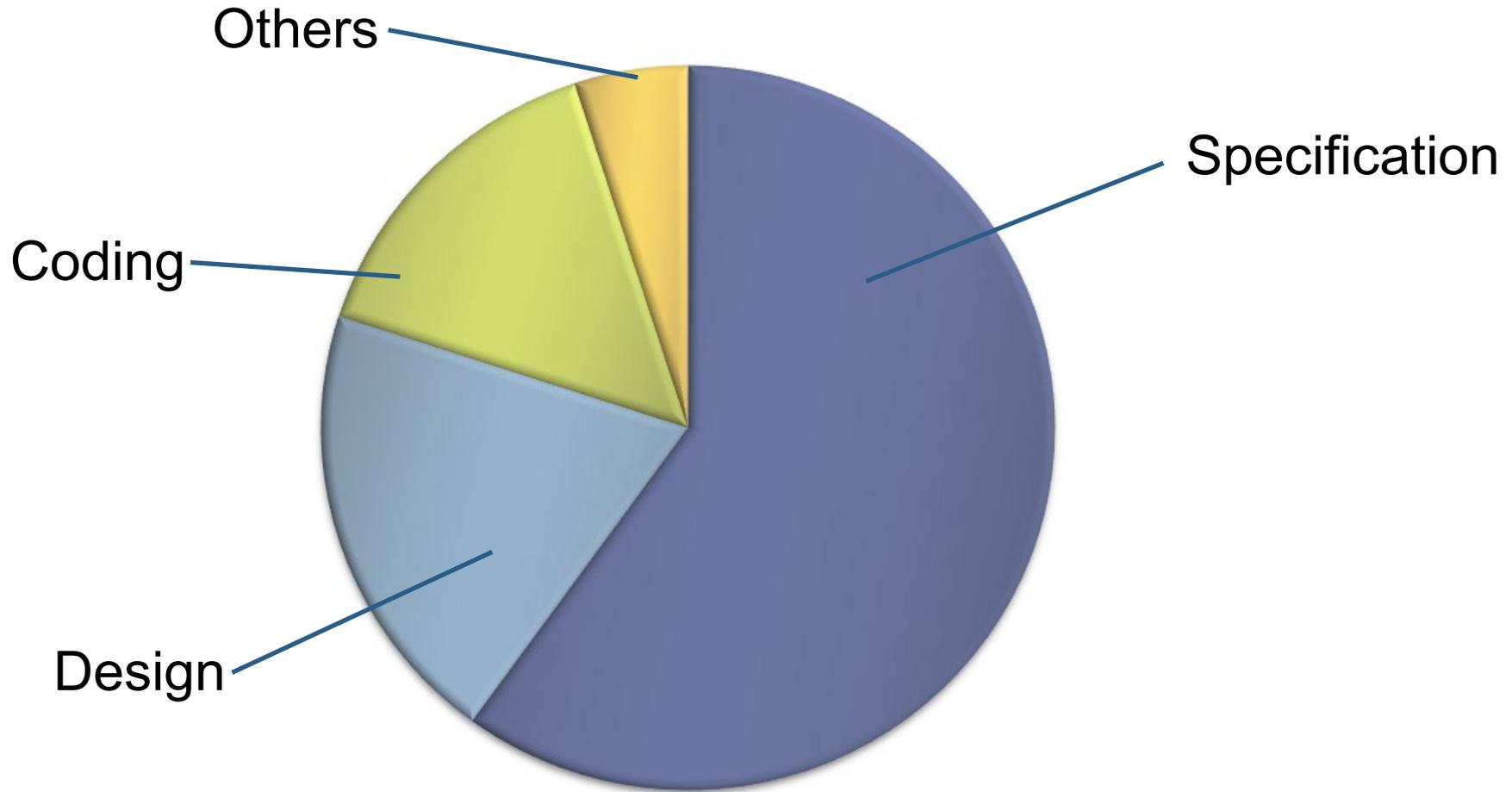
Implementation	Test cases
<pre>int square (int x) { return x*2; }</pre> <p>Fault</p>	<p>Square (2) = 4 Correct</p> <p>Square (3) = 6 Failure</p>

Important Considerations



- Detect system failures by choosing test inputs carefully.
- Determine the faults leading to the failures detected.
- Repair the faults leading to the failures detected; and
- Re-test the module/system.

Why do Failure Occur?

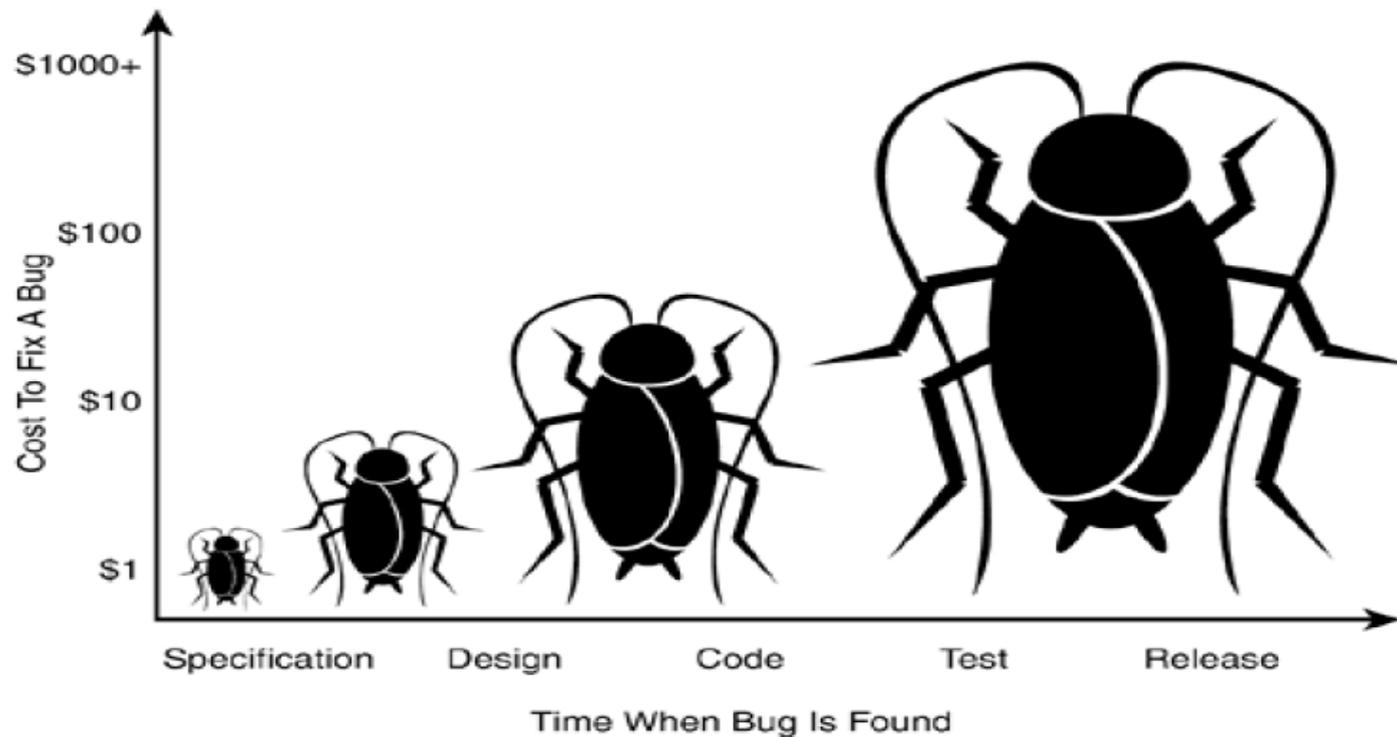


Root Causes of Failures



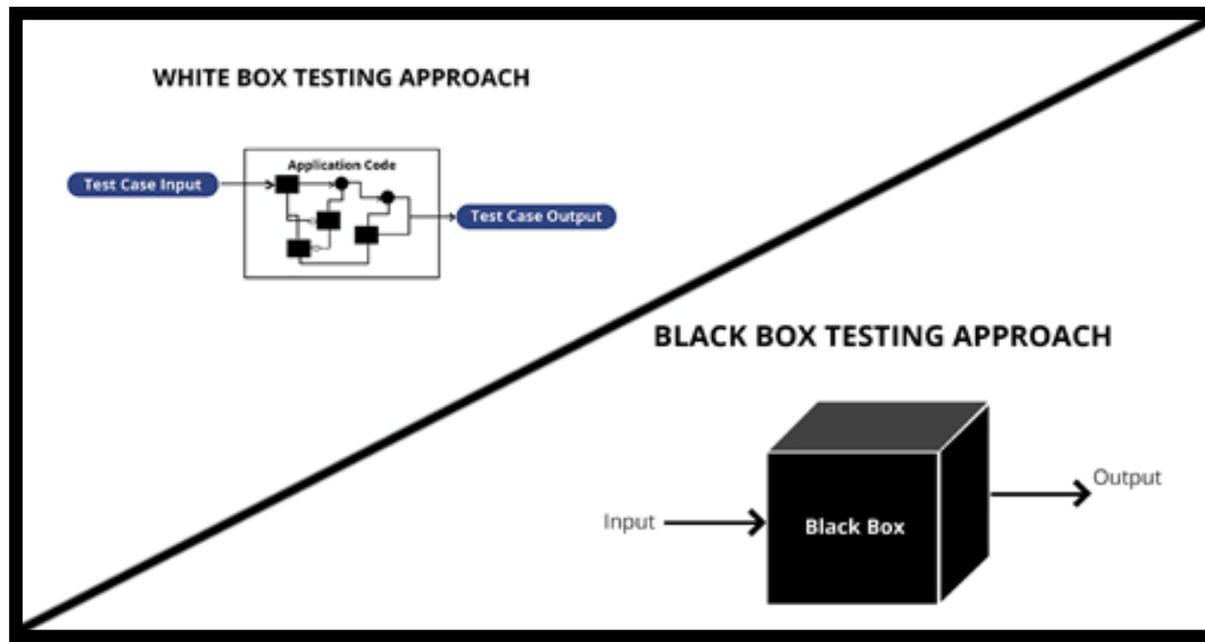
- Inaccurate understanding of end user requirements.
 - Inability to deal with changing requirements.
- Late discovery of serious project flaws.
 - For example, modules that do not fit together.
- Untrustworthy build & release process.
 - Implementation team's chaos.

Failure Costs



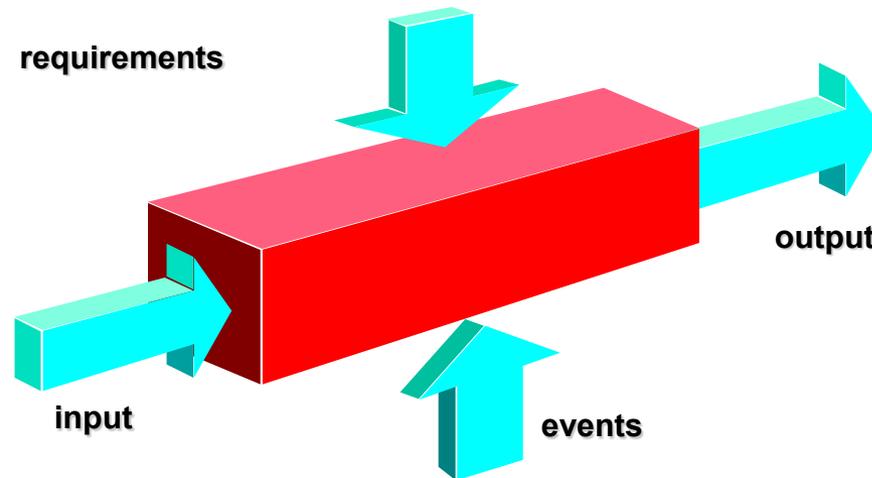
Types of Testing

- Black-box Testing
- White-Box Testing



Black-Box Testing

- Test cases are derived from formal specification of the system.
- Test case selection can be done without any reference to the program design or code.
- Only tests the functionality and features of the program.
 - Not the internal operation.



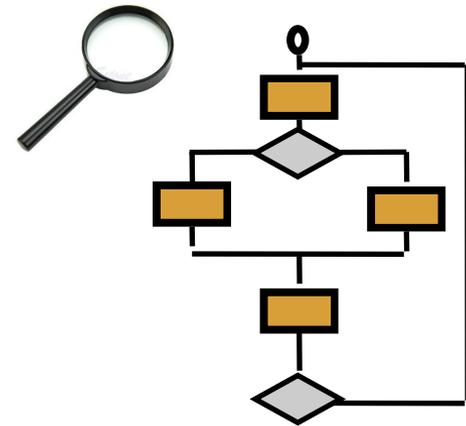
Black-Box Testing



- Advantages
 - Test case selection is done before the implementation of a program.
 - Help in getting the design and coding correct with respect to the specification.

White-Box Testing

- Test cases are derived from the internal design specification or actual code for the program.
- Advantages
 - Tests the internal details of the code
 - Checks all paths that a program can execute
- Limitations
 - Wait until after designing and coding the program under test in order to select test cases

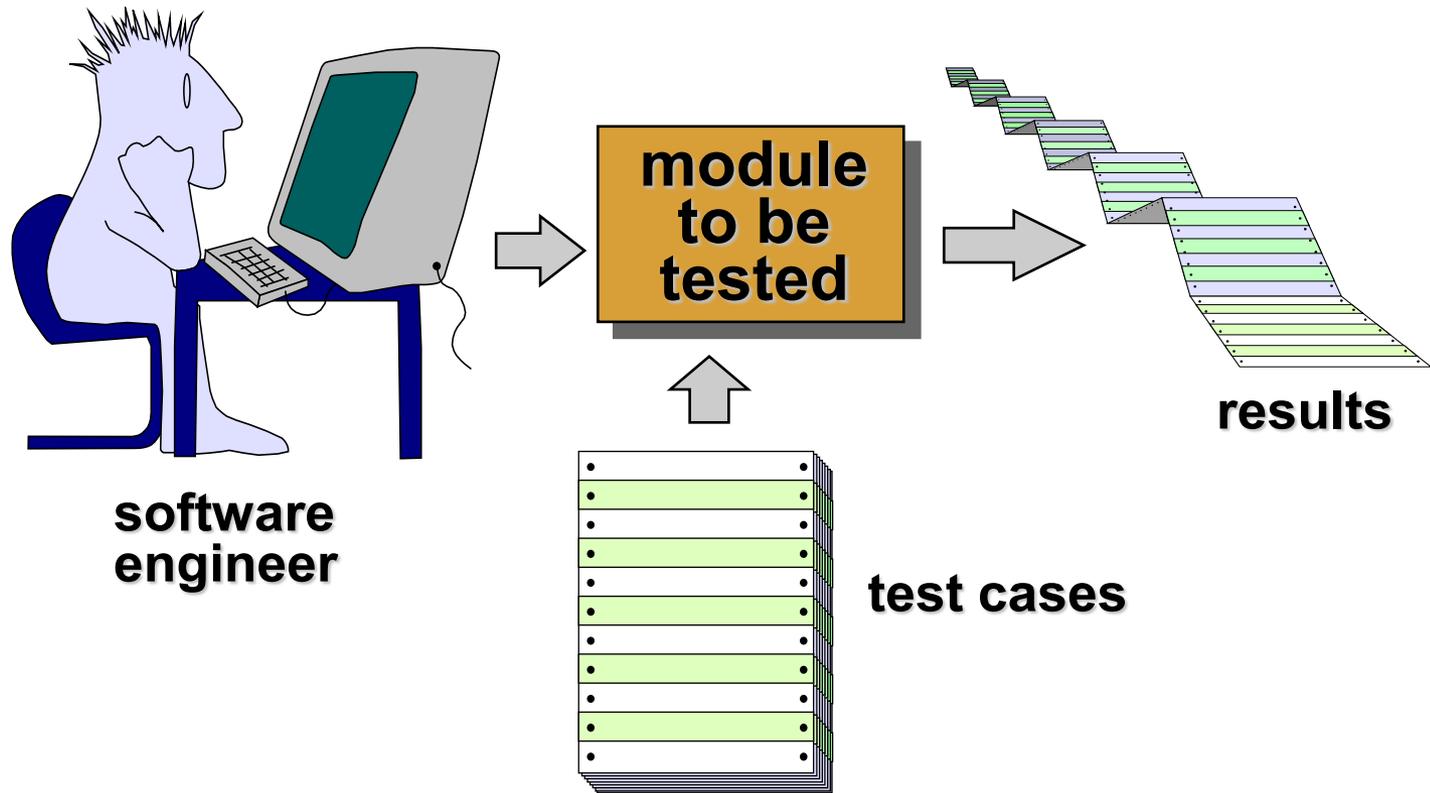


Levels of Testing



- Unit testing
 - Integration testing
 - System testing
 - Regression testing
- 

Unit Testing



Integration Testing



- Involves building a system from its components and
 - testing it for problems that arise from component interactions.

To simplify error localisation, systems should be incrementally integrated.



System Testing



- **System Functional Test**
 - Test entire system against the functional requirements.

 - **System Performance Test**
 - Test the performance of the system. For example, Response times, load testing etc.

 - **System Acceptance Test**
 - Set of tests that the software must pass before it is accepted by the client.
- 

Regression Testing

- Change do not always effect the entire program.
- Change in one part of system can effect other part.
- After each change
 - Entire test suite of a system must be run again.

Need for an automatic test suite execution.

Test Activities

- Boils down to selecting and executing test cases.
Test case consists of.....
 - Set of test inputs or a sequence of test inputs.
 - Expected results when the inputs are executed; and
 - Execution conditions or execution environment in which the inputs are to be executed.

These steps generally remain same from
unit testing to system testing.

Test Activities (cont)



- Test Evaluation
 - Compare the actual behavior with the expected behavior.
 - Generally can be automated to an extent !!!!

- Test Reporting
 - Report the outcome of the testing.
 - Developers
 - Project Mangers etc.
 - Generally can be automated to an extent !!!!

JUnit testing framework



- JUnit provides:
 - is a unit testing framework for the Java programming language.
 - Important in test-driven development (TDD) development method
 - Assertions for testing expected results.
 - Test suites for easily organizing and running tests.

- Latest version is 5. The example in the next slide is based on the previous version 4.



JUnit

Example

Class Under Test

```
package myPackage;
public class MyClass {
    public int Add (int x, int y)
    {
        return x+y;
    }
    public int Multiply (int x, int y)
    {
        return x*y;
    }
}
```

JUnit test for Add and Multiply

```
package myPackage;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
public class MyClassTest {
    MyClass tester = new MyClass();
    @Test
    public void testAdd() {
        assertEquals("Result", 8, tester.Add(6,2));
    }
    @Test
    public void testMultiply() {
        assertEquals("Result", 12, tester.Multiply(6,2));
    }
}
```

JUnit Annotations and Assertions

Annotation	Description
<code>@Test public void method()</code>	The <code>Test</code> annotation indicates that the public void method to which it is attached can be run as a test case.

Assertion	Description
<code>void assertEquals([String message], expected value, actual value)</code>	Asserts that two values are equal. Values might be type of int, short, long, byte, char or java.lang.Object. The first argument is an optional String message.
<code>void assertTrue([String message], boolean condition)</code>	Asserts that a condition is true.
<code>void assertFalse([String message],boolean condition)</code>	Asserts that a condition is false.
<code>void assertNotNull([String message], java.lang.Object object)</code>	Asserts that an object is not null.
<code>void assertNull([String message], java. lang.Object object)</code>	Asserts that an object is null.

Tutorial on Junit 4 and 5 for Eclipse



JUnit – Eclipse Tutorial (by Lars Vogel)

<http://www.vogella.com/articles/JUnit/article.html>

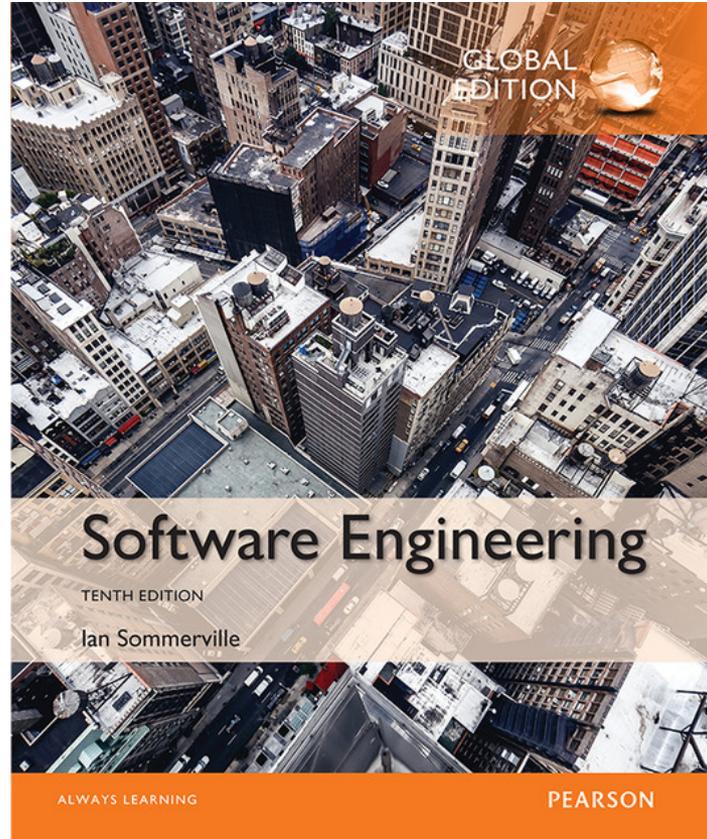
Key Points



- Software testing is the process of discovering evidence of defects and failures in software systems.
- Test early, test often, test enough.
- Testers should have good understanding of the development process, product.

Read

Chapter 8



References



- Ian Sommerville, “Software Engineering”, 10th Edition, Addison-Wesley, 2015.
- Timothy C. Lethbridge and Robert Laganière, “Object-Oriented Software Engineering: Practical Software Development using UML and Java”, 2nd Edition, McGraw Hill, 2001.
- R. S. Pressman, Software Engineering: A Practitioner’s Approach, 10th Edition, McGraw-Hill, 2005.
- <https://junit.org/junit5/docs/current/user-guide/>

Course Topics

- Introduction
- Software Process Models
- Requirements Engineering
- Modeling
- Software Construction Techniques
- Testing
 - Refactoring
 - Project Management
 - Ethical Issues