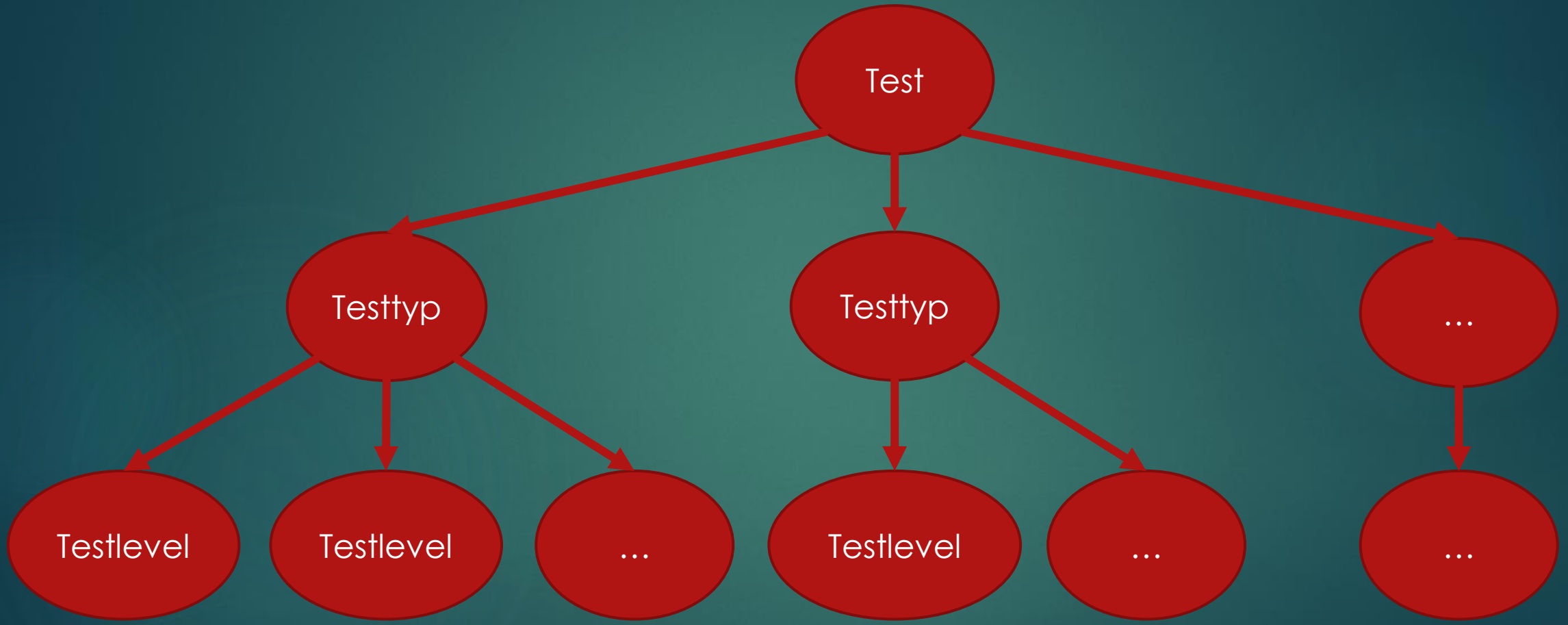
Software-Testing

- EINE EINFÜHRUNG -

Einordnung von Tests



Testtypen

- Functional Tests
- Nonfunctional Tests
- White-Box Tests
- Change-Related Tests (Regression)

Jeder der Testtypen kann auf allen vier Testlevels durchgeführt werden!

Für Beispiele bitte unter https://www.programsbuzz.com/article/test-types-and-test-levels in die Testtypen einlesen!

Testlevels

	Unit Tests	Integration Tests	System Tests	Acceptance Tests
Testobjekt	Individuelle Komponente (z.B. Funktion oder Klasse)	Schnittstellen/Interaktion zwischen den Komponenten	Gesamte System	Auszuliefernde Software
Testfokus	Komponenten	Schnittstellen	Funktionale und nicht funktionale Anforderungen	Funktionalität aus Sicht den End-Users
Fehler bei	Fehler in Komponente	Fehler bei Komponentenschnittstelle	Funktionale oder nichtfunktionale Anforderungen nicht eingehalten	Funktionalität aus Sicht des End-Users nicht gegeben
Durchzuführen von	Entwickler	Entwickler & Tester	Entwickler & Tester	Auftraggeber & End-User

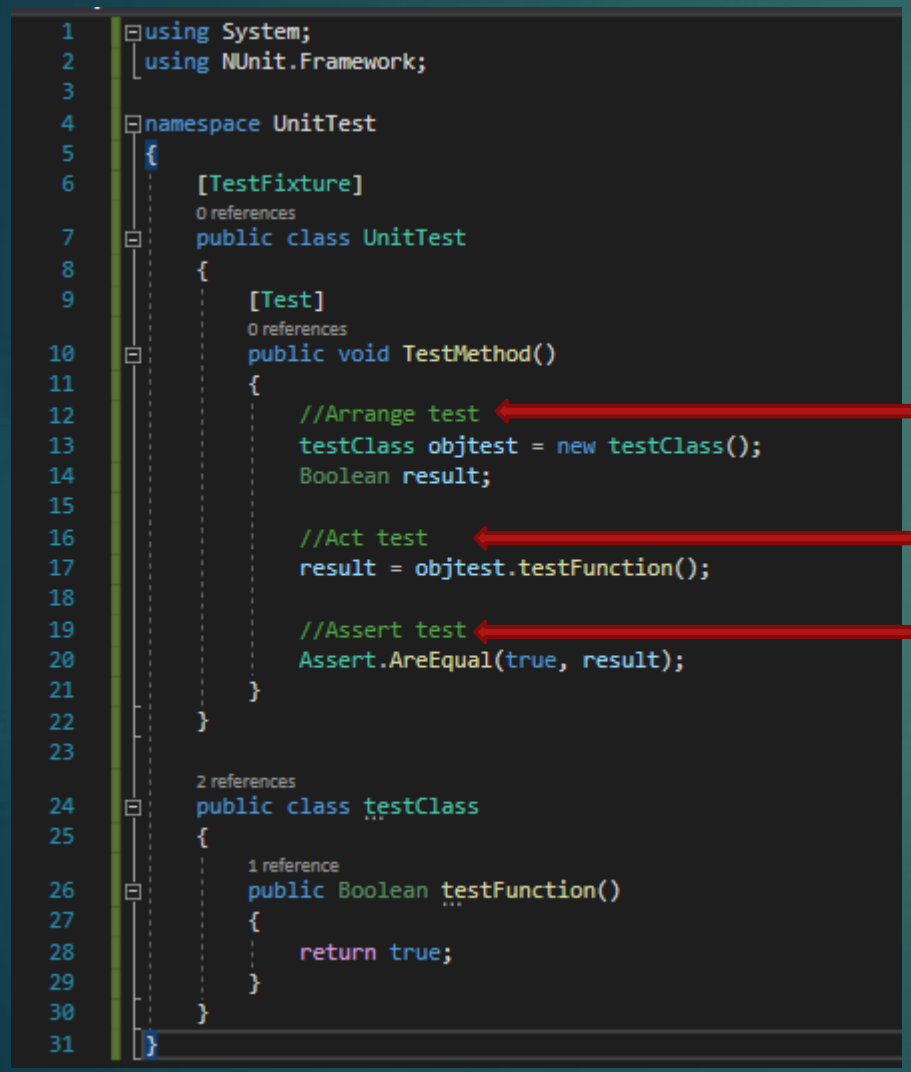
Testtechnik

- Use-Case Testing:
  - Testtechnik zur Identifizierung von Testfällen
  - Meist für Testfälle auf den Levels System Test und Acceptance Tests
  - Für jede Ablaufvariante im Use-Case ein Testfall (jede Möglichkeit sollte abgedeckt werden!)
  - Vorgehen bei der Erstellung:
    1. Analyse des Use-Cases und der unterschiedlichen Ablaufvarianten
    2. Festlegung der Testfälle: Ist Testfall für jede der Varianten nötig/sinnvoll? Werden mehr oder weniger Testfälle gebraucht?
    3. Festlegung und Herausarbeiten der Testdaten für jeden Testfall
  - Informationen und Bespiel unter: https://artoftesting.com/use-case-testing

Teststrategie

- Legt fest: Welche Teile des Systems mit welcher Intensität unter Anwendung welcher Testtechniken unter Nutzung welcher Test-Infrastruktur und in welcher Reihenfolge (siehe Testlevels) zu testen sind.
- Beispiele:
  - top-down: Haupt- vor Detailfunktionen testen, untergeordnete Routinen werden beim Test zunächst ignoriert oder simuliert
  - bottom-up: Detailfunktionen zuerst testen, übergeordnete Funktionen oder Aufrufe werden mittels „Testdriver“ simuliert
  - hardest first: Situationsbedingt das Wichtigste zuerst
  - big-bang: Alles auf einmal

Framework: NUnit



```
using System;
using NUnit.Framework;

namespace UnitTest
{
    [TestFixture]
    0 references
    public class UnitTest
    {
        [Test]
        0 references
        public void TestMethod()
        {
            //Arrange test
            testClass objtest = new testClass();
            Boolean result;

            //Act test
            result = objtest.testFunction();

            //Assert test
            Assert.AreEqual(true, result);
        }
    }

    2 references
    public class testClass
    {
        1 reference
        public Boolean testFunction()
        {
            return true;
        }
    }
}
```

AAA-Prinzip

Arrange: Vorbereitung für Testausführung (hier z.B. Objekterzeugung)

Act: Ausführung der zu testenden Funktion/Komponente

Assert: Überprüfung des Testergebnisses