**Soll = Ist**

Fehlerwirkung – Fehler der nach außen sichtbar wird

Fehlerzustand – Ursache einer Fehlerwirkung

Fehlzustand – Menschliche Handlung die zum Fehlerzustand führt

Qualitätssicherung

|  |  |
| --- | --- |
| Konstruktiv  (vermeiden das Fehler entstehen) | Analytisch  (überprüfen auf Fehler nach Fertigstellung) |
| Test Driven Development | Unit Tests |
| Planung | Blackbox/Whitebox Test |
| Coding Guidelines | Prüfen (Statische Code Analyse) |
|  | Testen (Programm ausführen) |

Lastenheft vom Kunden (was er will)

Pflichtenheft vom Entwickler (was er tun wird)

Komponente: Ein (abgeschlossener) Teil mit einer Schnittstelle

4-Phasen Testmuster: Setup, Execute, Verify, Teardown

Funktionaler Test: Testet Funktionalität, betrachtet das nac haußen Sichtbare Verhalten (BlackBox)

Nicht-Funktionaler Test: Performance, Lasttest (BlackBox)

Strukturorientierter Test: Testen auf Basis der Struktur (WhiteBox)

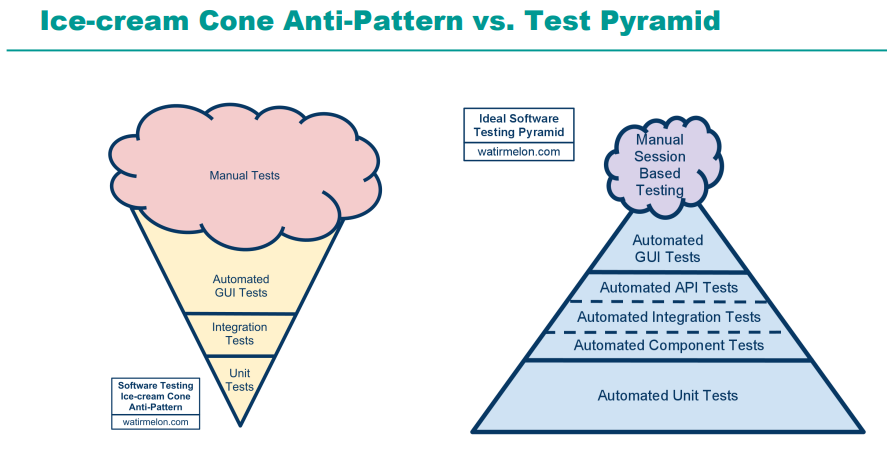
Test Principles:

* Fast (schnelle rückmeldung)
* Independent (tests hängen nicht voneinander ab)
* Repeatable
* Self-Validating (verify)
* Timely(kurz vor/nach/während der implementierung)

Selenium: UI Tests durch Browser automatisierung

Finding Elements By Id\Class-name

Page Object Model (POM): Design Pattern um Testcode besser wartbar zu machen



Mocks und Backdoor

Backdoor: Objekt Manipulieren um es in den Gewünschten Zustand zu bringen

Mocks: Objekt simulieren

Wenn Backdoor Manipulation nicht verfügbar oder zu Aufwendig ist

Whitebox Testing:

Kontrollflussbaum: Alles Statements und Verzweigungen einmal durchgehen

C0 Statement Abdeckung (Knoten) in Anzahl Abgedeckte Statemets z.B. 9

C0 Statement Überdeckung (Knoten) in Anzahl Abgedeckt Statemets z.B. 9 / Anzahl Statements (10) = 90%

C1 Zweig Abdeckung (Kanten) in Anzahl Abgedeckte Branches z.B. 9

C1 Zweig Überdeckung (Kanten) in Anzahl Abgedeckt Branches z.B. 9 / Anzahl Statements (10) = 90%



9 Statements/13 Branches

