Diplomarbeit – Software Testing

# Softwarre Testing – Introduction

Softwaretests sind ein wichtiger Bestandteil bei der Entwicklung einer Software. Sie dient dazu um die Qualität einer Software zu sichern und um Fehlerwirkungen vorzubeugen.   
Qualität ist definiert durch das Produkt, inwiefern dieses die Anforderungen erfüllt. Ein Standard für Qualitätsmanagement ist zum Beispiel das ISO 9000 oder das ISO 9126 speziell für Software.

Die ISTQB ( International Software Testing Qualification Board ) ist die führende gemeinnützige Zertifizierungsstelle für professionelle Software Tester in Europa. Ihr Ziel ist es, Softwaretests zu standardisieren.

# Allgemeine Informationen; Standards;

# Wieso Software Testen?

Eine gewisse Fehlerquote ist bei jedem Projekt zu erwarten, da Fehler nicht vollständig vermeidbar sind. Je nach Art des Fehlers, kann dieser jedoch das Produkt unbrauchbar machen, Verlust einbringen oder den Ruf einer Firma oder der Entwickler schaden.

Um dies zu vermeiden wird eine neu entwickelte Software getestet um Fehlerwirkungen zu finden. Dies ist eine wichtige Aufgabe und wird leider oft unterschätzt.

#Wieso wird getestet Wieso hab ich mich dafür entschieden?

# Allgemeine Begriffe

Im Bereich der Software Tests gibt es bestimmte Grundbegriffe. Die genauen Definitionen sind sehr wichtig für das Verständnis von Softwaretests und die Literatur die sich damit befasst.

|  |  |
| --- | --- |
| Fehlerwirkung | Eine Fehlerwirkung ist die Abweichung einer Komponente/eines Systems von der erwarteten Leistung oder Lieferung oder eine Abweichung des Ist-Zustand/verhalten vom Soll-Zustand/verhalten.  p 251 Glossar |
| Fehlerzustand | Defekte in einer Komponente oder einem System, der eine geforderte Funktion des Produkts beeinträchtigen kann, z.B. eine inkorrekte Anweisung oder Datendefinition. Ein Fehlerzustand, der zur Laufzeit angetroffen wird, kann eine Fehlerwirkung einer Komponente oder eines Systems verursachen. P.251 |
| Sollverhalten | Eine vorhergesagter Ausgabewert oder Verhalten eines Systems aufgrund der Spezifikation oder anderen Quelen. |
| Istverhalten | Das tatsächliche Verhalten/Ergebnis eines Systems bei einem Test unter spezifizierten Bedingungen. |
| Debugging | Tätigkeit die getan wird, um einen Fehlerzustand zu lokalisieren. |

# Testziel & Testumfang

Allgemein ist das Ziel von Softwaretests die Qualitätssicherung, unter anderem primär die Aufdeckung von Fehlerwirkungen im System. In der Praxis entwickelt ein System beim Finden von Fehlerwirkungen eine gewisse Testresistenz. Je mehr Fehlerwirkungen gefunden werden, desto schwieriger wird es, weitere zu finden. Der Testaufwand wird daher relativ zu der Dauer des Testens, immer größer. Grundsätzlich gilt, dass das komplette Testen und das Abdecken aller möglichen Eingabewerte für ein Programm, de facto unmöglich ist.

Dies lässt sich an einer einfachen Rechnung darstellen:  
  
Als Beispiel wird die Additionsfunktion eines Taschenrechners genommen, welcher bis zu 2 verschiedene ganzzahlige Werte addieren kann. Diese Zahlen werden im Programm, geschrieben in Java, als Integer dargestellt.

Ein Integer hat 2\*2^(31) mögliche Kombinationen.  
2\*2^(31) = 4294967296   
2^4294967296 sind die Anzahl der Möglichen Eingaben.

Werden automatisierte Tests genutzt und wird davon ausgegangen dass ein Test 0.002 Sekunden braucht, ergibt sich folgende Rechnung:  
2^4294967296 \* 0.002  
  
In Jahren umgerechnet, würde ein vollständiger Test mehr als 1 Milliarde Jahre dauern. Theoretisch aber nicht praktisch möglich.  
  
Um trotzdem viele Fehlerwirkungen zu finden, gibt es verschiedene Testmethoden und Teststrategien die im späteren genauer erläutert werden. Den tatsächlichen Testaufwand abzuschätzen ist jedoch sehr schwer. Auf die Frage ob sich ein Aufwand überhaupt auszahlt, kommt die Gegenfrage: „Im Vergleich zu was?“ (DeMarco 93)

Zumeist hängt der Testaufwand von dem Risiko ab. Das Risiko definiert sich durch die Höhe des Schadensfalles bei Eintritt und die Eintrittswahrscheinlichkeit. Ist das Risiko hoch, so ist ein höherer Testaufwand gerechtfertigt. Aus einem Artikel der Frankfurter Zeitung vom 17.01.2012 mit dem Titel „IT-Systemausfälle kosten viele Millionen“, geht hervor, dass der Ausfall von einer Stunde in der Börse 7,8 Mio. Dollar kostet. Bei anderen Systemen wie einem KIS ( Krankenhausinformationssystem ), kann ein Fehlverhalten sogar Menschenleben kosten.  
  
#Was ist das Ziel von Testing?   
Testobjekt?  
Testumfang?

# Softwarequalität

In einem Projekt werden Tests ausgeführt um Fehlverhalten zu finden und somit die Softwarequalität zu verbessern. Die Softwarequalität wird jedoch durch mehrere Faktoren definiert. ISO 9261 wird dabei als Standard angesehen, darin werden folgende Merkmale für die Softwarequalität definiert:

* Funktionalität
* Zuverlässigkeit
* Benutzbarkeit
* Effizienz
* Wartbarkeit
* Übertragbarkeit

## Funktionalität

Die Funktionalität definiert inwieweit die Software die geforderte Funktion erfüllt. Diese Funktionen werden oft durch ein spezifisches Ein-/Ausgabeverhalten und/oder Auswirkung auf das System beschrieben. Das Merkmal der Funktionalität kann in folgende weitere Teilmerkmale unterschieden werden:   
Richtigkeit, Interoperabilität, Sicherheit und Konformität

## Zuverlässigkeit

Offizielle Definition nach ISO/IEC 25010:2011:

„degree to which a system, product or component performs specified functions under specified conditions for a specified period of time“

ISO/IEC 25010:2011, Online in Internet: URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en>, 17.03.2016.

Zuverlässigkeit beschreibt inwiefern das System ein bestimmtes Leistungsniveau über einen bestimmten Zeitraum aufrechterhalten kann. Sie lässt sich in folgende Teilmerkmale aufteilen:  
Fehlertoleranz und Wiederherstellbarkeit.

## Benutzbarkeit

Bei Systemen die interaktiv sind und von vielen Kunden benutzt wird, spielt die Benutzbarkeit eine große Rolle. Sie beschreibt den Aufwand für die Nutzung der Software.

## Effizienz

Das Verhältnis der Leitung zu den verbrauchten Ressourcen beschreibt den Grad der Effizienz. Bei guter Leistung mit wenig Ressourcenverbrauch ist eine gute Effizienz gegeben und umgekehrt.

## Wartbarkeit

Wartbarkeit wird in der Fachliteratur auch als Änderbarkeit bezeichnet. Sie beschreibt wie gut das System modifizierbar und änderbar ist.

## Übertragbarkeit

Die Übertragbarkeit beschreibt wie gut das System sich in einer anderen Umgebung intrigieren lässt.

# Psychologie des Testens

S35

Bei der Entwicklung einer Software wird ein neues Produkt erschaffen. Softwareentwicklung wird generell als etwas Konstruktives angesehen, während Softwaretests als etwas eher Destruktives angesehen werden. Tester werden oft ungern gesehen, da diese auf unangenehme Fehler hinweisen.

Um Probleme vorzubeugen ist es wichtig, dass Probleme und Fehlerwirkungen sachlich und neutral dem Entwickler übermittelt werden. Die Fehlerwirkungen sollten als Verbesserungsvorschläge in Bezug auf die Software und nicht als Kritik an den Entwickler dargestellt werden. Entwickler und Tester sollten zwar unabhängig aber keine Feinde sein.

Die Unabhängigkeit zwischen Entwickler und Tester sind insofern wichtig, da Entwickler alle Use Cases und alle möglichen Sonderfälle schon während der Entwicklung berücksichtigen. Ein testen der schon sowieso berücksichtigten Fälle wäre weniger effizient. Ein unabhängiger Tester kann jedoch andere Testszenarien berücksichtigen und anhand der Spezifikation die Sicht des Endkunden nachvollziehen. Natürlich kann ein Entwickler als Tester auch Vorteile bringen, vor allem bei geringen finanziellen und zeitlichen Ressourcen, eine allgemein gültige Aussage was besser ist, kann nicht getroffen werden.

Im Idealfall sollte der Entwickler sachlich mit dem Tester zusammenarbeiten, um so die Qualität des Produktes zu verbessern.

#ProblemeBeiFehlerFIndung

# 7 Prinzipien des SoftwareTestens

Aus dem Buch „Software-TEST IT PROFESSION@LlY“ von Mag. Anja Kribernegg geht hervor, dass sich in den letzten Jahren Grundsätze zum Testen gebildet haben, welche als Leitlinien angesehen werden. Diese sind wie folgt:

**Grundsatz 1: Durch Tests werden Fehlerzustände gefunden**Durch das Testen sollen möglichst viele Fehlerwirkungen gefunden werden und damit die Fehlerzustände nachgewiesen werden. Absolute Fehlerfreiheit kann jedoch nicht durch das Testen nachgewiesen werden, sie verringert nur die Anwesenheit unentdeckter Fehlerzustände.

**Grundsatz 2: Keine Software kann vollständig getestet werden** Das vollständige Testen und ausführen alle möglichen Testfälle, ist nur in der Theorie aber nicht in der Praxis möglich. Tests werden mit Stichproben durchgeführt. Je vielfältiger die Stichproben, desto höher der Testaufwand. Der Testaufwand entspricht relativ dem Risiko und somit der Priorität.

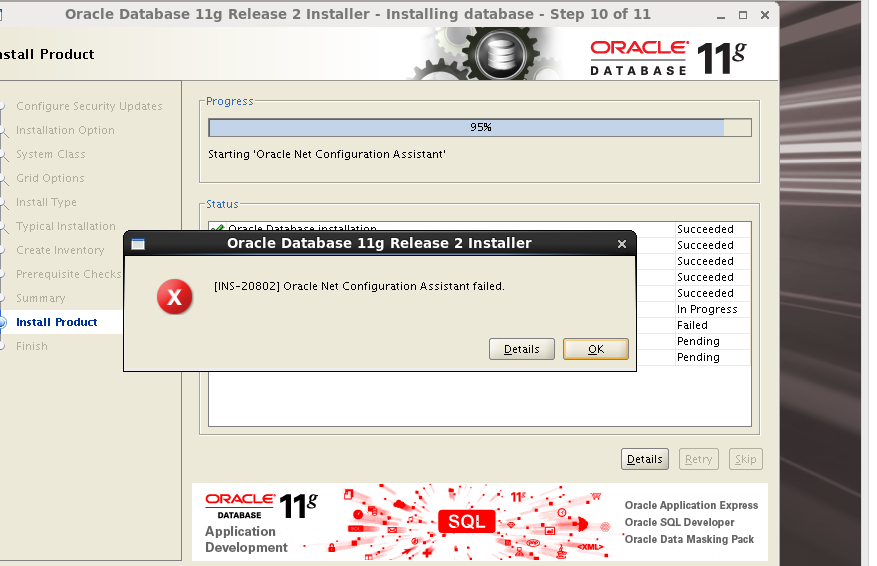
**Grundsatz 3: Je früher desto besser**Das Testen einer Software sollte so früh wie möglich begonnen werden. Am vorteilhaftesten ist das Testen während der Entwicklung. Auf fehlerhaften Komponenten aufbauen, kann zu noch mehr Fehlern führen, außerdem ist Testen an sich eine sehr zeitintensive Tätigkeit.

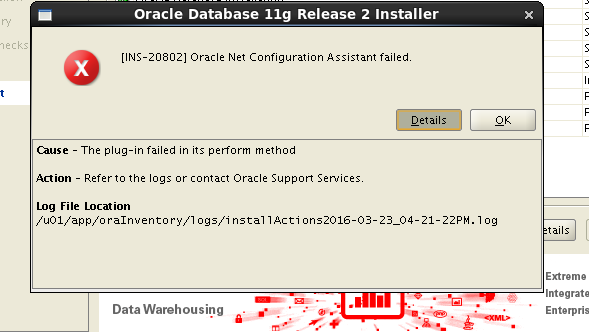
**Grundsatz 4: Fehlerdichte beachten**Erfahrungsgemäß häufen sich die auftretenden Fehlerwirkungen in bestimmten Bereichen eines Programmes.

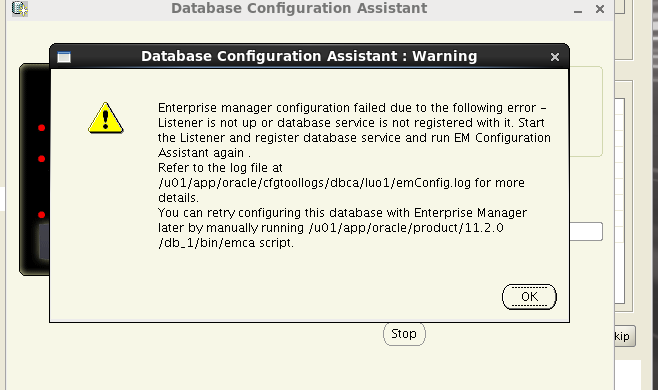
**Grundsatz 5: Testresistenz**Mit zunehmender Testdauer und Testaufwand steigt auch die Testresistenz. Je mehr Testfälle ausgeführt wurden, desto weniger neue Fehlerwirkungen werden gefunden.

**Grundsatz 6: Das Umfeld ist entscheidend**Je nach Umfeld, kann das Vorgehen anders sein. Generell wird eine Eismaschine anders getestet als ein Krankenhausinformationssystem (KIS). Sowohl der Aufwand als auch die verwendete Teststrategien weichen sehr stark voneinander ab.

**Grundsatz 7: Fehlerfreiheit bedeutet nicht Kundenzufriedenheit**Fehlerfreiheit kann nie garantiert werden und sie bedeutet auch nicht, dass das System dann auch den Erwartungen und Vorstellungen des Kunden entspricht.







# Testen im softwarezyklus

Die richtige Integration der Testaktivitäten in das laufende Projekt ist sehr wichtig. Das Testen während der Entwicklung hilft dem Team frühzeitig Fehler zu erkennen und Folgefehler zu vermeiden, es muss jedoch eine richtige Planung zuvor erfolgen die je nach Vorgehensweise des Projektes angepasst wird.

## Modelle

Im Bereich des Projektmanagements gibt es verschiedene Modelle die eine gewisse Vorgehensweise zur Entwicklung eines Produktes beschreiben. Ein paar der bekannteren Modelle:

* Wasserfallmodell
* Extreme Programming
* Kanban
* Scrum
* V – Modell

Für Tester ist das V-Modell nach Barry W. Boehm ein sehr interessantes und wichtiges Modell. Es setzt die Wichtigkeit von den einzelnen Teststufen mit denen der Entwicklung gleich. Boehm fokussiert mit dem V Modell 4 wesentliche Ziele an: Completeness, consistency, feasibility und testability.  
Das Modell besteht grundsätzlich aus zwei Zweigen die zusammen ein V bilden. Der linke Zweig beschreibt die einzelnen Schritte zur Erstellung einer Software, vom Design bis hin zur Programmierung. Die Schritte werden von grob immer genauer, während auf dem rechten Zweig parallel die Verifizierung und Validierung durchgeführt wird.

Das Testen endet aber nicht zwingend, nachdem eine Software auf dem Markt kommt. Bestehende Systeme werden oft Jahrzehnte lang genutzt und benötigen immer wieder Updates. Diese Updates können wiederum Fehlerwirkungen hervorrufen, die davon nicht da waren. Funktionalitäten die bei vorherigen Tests keine Fehlerwirkungen hervorgerufen haben, können nach den Updates ihr Verhalten ändern. Aus diesem Grund gibt es Regressionstest. Regressionstest sind Tests die wieder ausgeführt werden, um die Fehlerfreiheit in den schon durchgeführten Testfällen, nach einem Update zu gewährleisten. Im Zusammenhang mit Regressionstest, spielen automatische Tests (siehe Glossar) eine große Rolle.

## Komponententest

Meistens entwickelt jeder Entwickler einen Teil der Software, eine Komponente davon, die dann später mit den Komponenten anderer Entwickler zusammengeführt werden.

Der Komponententest dient dazu, diese Komponenten im einzelnem zu testen. Diese Komponenten, auch Module oder Units genannt, können wenn sie fehlerhaft sind, später bei der Integration, Fehlermaskierungen hervorurrufen.

In zahlreichen Programmiersprache wie zum Beispiel C#, gibt es eigene Frameworks wie NUnit die sich mit automatisierten Unit Tests auseinandersetzen.

## Integrationstest

Integrationstests befassen sich mit der Zusammenarbeit der einzelnen Komponenten untereinander. Ziel des Integrationstestes ist es, Fehlerwirkungen zwischen den Komponenten zu finden.

Auch wenn die einzelnen Komponenten fehlerfrei funktionieren, können bei der Integration der einzelnen Teile in das System, Fehler entstehen. Komponente A könnte Komponente B irrtümlich falsche Datentypen übergeben, oder richtige Datentypen im falschen Einheitsmaß.

Ein gutes Beispiel für einen schweren Fehler der beim Integrationstest übersehen wurde, ist der Mars Climate Orbiter (MCO) bei dem eine Sonde abstürzte. Der Grund war eine Berechnung im falschen Einheitensystem. Im offiziellen Bericht hieß es:

„Findings of the failure review board indicate that a navigation error resulted from some spacecraft commands being sent in English units instead of being converted to metric. This caused the spacecraft to miss its intended 140 - 150 km altitude above Mars during orbit insertion, instead entering the martian atmosphere at about 57 km. “

## Systemtest

Der Systemtest befasst sich mit dem System als Ganzes. Während der Integrationstest die Interaktion zwischen den einzelnen Komponenten aus technischer Sicht testet, wird im Systemtest die Software als Einheit gegen die Produktdefinition geprüft. Wie die Software aufgebaut ist, wird dabei nicht berücksichtig.

## AbnahmeTest

Der Abnahmetest ähnelt dem Systemtest, jedoch dient der Abnahmetest nicht zur Qualitätssicherung, sondern zum Nachweis, dass die geforderten Funktionen gesetzlich erfüllt worden sind. In den meisten Ländern ist die Abnahme eines Produktes gesetzlich geregelt.

So heißt es zum Beispiel im deutschen Bürgerliches Gesetzbuch (BGB) Paragraph 640:

„Der Besteller ist verpflichtet, das vertragsmäßig hergestellte Werk abzunehmen, sofern nicht nach der Beschaffenheit des Werkes die Abnahme ausgeschlossen ist. Wegen unwesentlicher Mängel kann die Abnahme nicht verweigert werden.”

(BGB01, Parag. 640, Abs. 1, Satz 1 und 2)

Die Abnahme wird zusammen mit dem Kunden und einem Abnahmeprotokoll durchgeführt. Erfüllt das Produkt vertraglich alle Anforderungen, wird dies mit einer Unterschrift bestätigt.

# Grundlegende Softwaretest

Es kann während der verschiedenen Teststufen, verschiedene Testarten unterschieden werden. Je nach Teststrategie und Test Ziel, werden verschiedene Testarten genutzt.

## Funktionale und nicht funktionale Tests

Die einfachste Unterscheidung ist die Unterscheidung zwischen funktionalen und nicht funktionalen Tests. Softwarequalität befasst sich nicht nur mit dem Bereich der Funktionalität. Diese wird mit funktionalen Tests geprüft, es gibt aber auch andere Merkmale der Qualität wie Benutzbarkeit und Zuverlässigkeit. Diese werden mit nicht funktionalen Tests geprüft.

## Statische Tests

### Diverse ReviewArten

### Diverse COmpiler /REfactor

## Dynamische Tests

### White box

### Black Box

### Äquivalenzklassen Bildung

### Zustandsbezogene Tests

### Anweisungstest

### Entscheidungstest

### Pfadtest

# Testmanagement

## Analyse

## Testkonzept

## Testpriorität

## Kriterien für Testart

## Testprotokoll

## Testauswertung

## Fehlermanagement

## Kriterien zum Test ENde

# Sofrwaretesting – Doungen

MTP