**Qualitätssicherungskonzept**

**für die Softwareentwicklung**

**Teil 1 - Grundlagen des Softwaretestens**

**Autor: Marion Grabenweger**

**12. November 2012**

**Version 0.3**

**Inhaltsverzeichnis**

[**1 Einleitung 1-2**](#_Toc340584858)

[**1.1 Motivation 1-2**](#_Toc340584859)

[**1.2 Vision - OFFEN 1-2**](#_Toc340584860)

[**2 Grundlagen des Softwaretestens 2-3**](#_Toc340584861)

[**2.1 Begriffe 2-3**](#_Toc340584862)

[**2.2 Fehlerbegriff 2-3**](#_Toc340584863)

[**2.3 Testbegriffe 2-3**](#_Toc340584864)

[**3 Softwarequalität 3-7**](#_Toc340584865)

[**4 Organisation von Testteams 4-9**](#_Toc340584866)

[**4.1 Psychologie des Testens 4-9**](#_Toc340584867)

[**4.2 Organisationsformen im Test: 4-9**](#_Toc340584868)

[**4.3 Rollen 4-10**](#_Toc340584869)

[**4.4 Soziale Kompetenz 4-11**](#_Toc340584870)

[**5 Testwerkzeuge & Automatisierung 5-12**](#_Toc340584871)

[**5.1 Testwerkzeugkonzepte 5-12**](#_Toc340584872)

[**5.2 Testwerkzeugkategorien 5-13**](#_Toc340584873)

[**6 Normen und Standards 6-15**](#_Toc340584874)

# Einleitung

Das Qualitätssicherungskonzept soll die Grundsätze der Qualitätssicherungsmaßnahmen der Softwareentwicklung regeln.

Dieses Dokument gliedert sich in 4 Teile:

Teil 1 „Grundlagen des Softwaretestens“ beschreibt die Grundlagen des Softwaretestens und inkludiert Begriffsbestimmungen, um ein gemeinsames „Wording“ zu ermöglichen.

Im 2. Teil wird ein generischer Testprozess beschrieben. Durch die Gliederung in Teilprozesse soll sichergestellt werden, dass das Testvorgehen für alle Projekte anwendbar ist, wobei Teile des Prozesses von anderen Bereichen übernommen werden können.

Der 3. Teil beschäftigt sich mit der Dokumentation, die während der jeweiligen Testteilprozesse erstellt werden kann und liefert Templates und Beispiele.

Im 4. Teil werden die Testverfahren und Testtechniken erklärt, die zum Einsatz kommen können um eine systematische Testfallableitung zu gewährleisten.

Alle 4 Teile haben zum Ziel einen systematischen und strukturierten Testprozess einzuführen um die Qualität der zu liefernden Softwareprodukte zu steigern.

##  Motivation

„*Wenn irgendein Teil falsch eingebaut werden kann, so wird sich immer jemand finden, der das auch tut*“ - Murphy’s Gesetz

Lange Zeit wurden Softwaretests ausschließlich dafür eingesetzt Fehler aufzudecken. Am Ende der Entwicklungsphase wurden mehr oder weniger erfahrungsbasierte Tests durchgeführt, um bereits bekannte Fehler nicht neuerlich auszuliefern. Die Tester waren entweder Mitarbeiter aus dem Fachbereich, oder die Entwickler, die ihre eigene Software getestet haben.

Mittlerweile wurden international anerkannte Softwaretest-Standards geschaffen, sowie die Berufsgruppe Softwaretester etabliert. Der zertifizierte Softwaretester bringt eine Menge an Wissen über Rolle und Verantwortung bei der Erfüllung von Test-Aufgaben mit und kann die Methoden und Techniken des Softwaretestens einsetzen. Alle Schritte und Aktivitäten des Softaretestprozesses von der Planung, über die Spezifikation, der Durchführung der Tests bis hin zur Testendebewertung sollten daher von einem zertifizierten Softwaretester durchgeführt werden.

##  Vision - OFFEN

Nichtziel: QS der Prozessqualität

# Grundlagen des Softwaretestens

## Begriffe

Gerade im Bereich des Softwaretestens werden verwirrend viele Begriffe benutzt, was die Kommunikation erheblich erschwert. Als Basis für einen gemeinsamen Wortschatz, werden in diesem Handbuch die Begriffe aus dem ISTQB/GTB Standardglossar der Testbegriffe verwendet.

## Fehlerbegriff

Eine Situation kann nur dann als fehlerhaft bezeichnet werden, wenn zuvor festgelegt wurde, wie die erwartete, korrekte, Situation aussehen soll. Daraus ergeben sich nachfolgende Begriffsbestimmungen:

* **Fehler:** Ein Fehler ist die Nichterfüllung einer festgelegten Anforderung, einer Abweichung zwischen dem IST- und dem SOLL-Verhalten laut Spezifikation oder Anforderung. [ISO 8402]
* **Mangel:** Ein Mangel liegt vor, wenn eine gestellte Anforderung oder berechtigte Erwartung unzureichend erfüllt wird.
* **Fehlhandlung:** Eine Fehlhandlung ist eine unwissentlich, versehentlich oder absichtlich ausgeführte Handlung oder Unterlassung, die unter Umständen dazu führt, dass eine geforderte Funktion eines Produktes beeinträchtigt wird [DIN 66271]. Die Fehlhandlung einer Person ist somit die Ursache für einen Fehlerzustand. Beispielsweise die fehlerhafte Programmierung des Entwicklers.
* **Fehlerzustand:** Auch Defekt oder innerer Fehler, bezeichnet einen fehlerhaften Programmteil, Anweisung oder Datendefinition, aber auch ein inkorrektes Dokument, das die Ursache für eine Fehlerwirkung sein kann.
* **Fehlerwirkung:** Auch Ausfall oder äußerer Fehler, ist die Wirkung eines Fehlerzustandes, die bei der Ausführung des Testobjektes nach außen für den Anwender sichtbar auftritt.
* **Fehlermaskierung:** Ein Umstand, bei dem ein Fehlerzustand die Aufdeckung eines anderen verhindert. Ein vorhandener Fehlerzustand wird durch einen oder mehrere andere Defekte kompensiert, so dass der fehlerhafte Zustand selbst gar keine Fehlerwirkung verursacht.

## Testbegriffe

* **Softwaretest:** Unter Testen versteht man den Prozess, der aus allen Aktivitäten des Lebenszyklus besteht (sowohl statisch als auch dynamisch), die sich mit der Planung, Vorbereitung und Bewertung eines Softwareprodukts und dazugehöriger Arbeitsergebnisse befassen.

 Ziel des Prozesses ist sicherzustellen,

* dass diese allen festgelegten **Anforderungen** genügen
* dass sie ihren **Zweck** erfüllen
* und etwaige **Fehlerzustände** zu finden.

 Es gibt unterschiedliche Begriffe für verschiedene Arten von Softwaretests. Die Tests können unterschiedlich bezeichnet werden. Und zwar nach:

* **Testziel**: Die Benennung einer Testart aufgrund des Zwecks des Testes (z.B.: Performancetest)
* **Testmethode**: Der Test wird nach der Methode benannt, die zur Spezifikation oder Durchführung der Tests eingesetzt wird. (z.B.: Anwendungsfallbasierte Tests)
* **Testobjekt:** Benennung nach Art des Testobjektes, das getestet wird (z.B.: GUI-Test, Datenbanktest)
* **Teststufe:** Der Test wird nach der Entwicklungsstufe des Vorgehensmodells benannt (z.B.: Komponententest, Integrationstest)
* **Testperson:** Der Test wird nach dem durchführenden Personenkreis bezeichnet (z.B.: Entwicklertest, Trägertest)
* **Testumfang:** Tests werden nach dem Umfang unterschieden
* **Testziel:** Testen verfolgt mehrere Ziele.
* Ausführung des Programms mit dem Ziel, Fehlerwirkungen nach zu weisen
* Ausführung des Programms mit dem Ziel, die Qualität zu bestimmen
* Ausführung des Programms mit dem Ziel, Vertrauen in das Programm zu erhöhen
* Analysieren des Programms oder der Dokumente, um Fehlerwirkungen vorzubeugen
* **Testbasis:**

Die Dokumentation aus der Testfälle abgeleitet werden können. In der Praxis ist die Testbasis oft vage oder widersprüchlich. Es kann die Abdeckung wichtiger Bereiche fehlen, unvollständig bzw. einfach nicht vorhanden sein. Selbst wenn die Testbasis gut spezifiziert vorliegt, können komplexere Interaktionen die Definition der erwarteten Testergebnisse erschweren.

* **Testorakel:** Informationsquelle zur Ermittlung des Sollverhaltens eines Testobjektes. Mögliche Testorakel sind:
* Anforderungsspezifikationen
* Altsystem
* Benutzerdokumentation
* Programmvariante im Back-to-back Test
* das Testobjekt selbst (beim explorativen Test)
* **Testbedingung:** Eine Einheit oder ein Ereignis, z.B. eine Funktion, eine Transaktion, ein Qualitätsmerkmal oder ein strukturelles Element einer Komponente oder eines Systems, die bzw. das durch einen oder mehrere Testfälle verifiziert werden kann.
* **Testobjekt:** Die Komponente, integriertes Teilsystem oder System, welches gegen die Anforderungen in der Testbasis getestet wird. Beispielsweise
* Codebasierte Testobjekte:
	+ eine ausführbare Komponente
	+ ein Teilsystem
	+ ein komplettes System
* Dokumentenverbundene Testobjekte:
	+ ein Plan (z.B.: Projektplan, Testplan, Konfigurationsmanagementplan)
	+ eine Anforderungsspezifikation
	+ Architekturentwurf
	+ Sourcecode
	+ ein Handbuch(z.B.: Benutzerhandbuch, Installationshandbuch)
	+ Testspezifikationen und Testprozedur
* **Testelement:** Das einzelne Element, das getestet wird. Gewöhnlich existieren einTestobjekt und viele Testelemente.
* **Testkriterien:** Die Kriterien, die ein System oder eine Komponente für das Bestehen eines Tests erfüllen muss.
* **Testfall:** Ein Testfall ist eine Menge von Eingaben, Ausführungsbedingungen und einem Pass/Fail-Kriterium
* **abstrakter Testfall:** Abstrakte Testfälle sind logische Testbeschreibungen (Teststeps) der Eingaben und erwarteten Ergebnisse. Die zu verwendenden konkreten Daten sind noch nicht definiert.
* **konkreter Testfall:** In konkreten Testfällen sind die zu verwendenden Daten definiert. Da sich viele Testfälle auf Tabellen mit konkreten Daten beziehen, bezeichnet man die Testfälle erst dann als konkret, wenn die Anweisungen mit diesen Informationen verknüpft sind.

 Wichtige Bestandteile der Beschreibung eines Testfalls sind:

* **Test-Id:** Eindeutige Bezeichnung, lfd. Nummer
* **Anforderungsreferenz/Quelle:** die Benennung des Testobjekts und der Spezifikationen, auf die sich der Testfall bezieht
* **Risiko/Priorität:** Beispielsweise „hoch“, „mittel“, „gering“
* **Konfiguration:** Hinweise zur Konfiguration der Testumgebung
* **Testverfahren & Testtiefe:** welche Test(entwurfs)verfahren, in welcher Testumgebung
* **Vorbedingung:** Vorbedingungen, die vor der Testausführung hergestellt werden müssen
* **Testschritte:** die Handlungen, die zur Durchführung des Testfalls notwendig sind, Anweisungen wie Eingaben na das Testobjekt zu übergeben sind
* **Testdatenliste:** die Eingabedaten für die Durchführung des Tests bzw. Verweis auf die Testdaten
* **Sollreaktionen:** die erwarteten Ergebnisse und/oder Reaktionen des Testobjekts auf die Eingaben
* **Nachbedingung:** die erwarteten Nachbedingungen, die als Ergebnis der Durchführung des Testfalls erzielt werden

Jeder Testfall muss reproduzierbar, nachprüfbar und bis zu den Anforderungen rückverfolgbar sein.

* **Positivtests:** Das Verhalten des Testobjekts in Folge gültiger Vorbedingungen und Eingaben wird überprüft.
* **Negativtests** (auch **Robustheitstests**)**:** Das Verhalten des Testobjekts in Folge ungültiger Vorbedingungen und /oder Eingaben wird überprüft.
* **Teststatus:** Testfälle nehmen im Verlauf ihrer Bearbeitung unterschiedliche Statusausprägungen an:
* im Entwurf
* zum Testen freigegeben
* im Test
* Test ok
* Test fehlerhaft
* **Testdaten:** Eingabe- und Zustandswerte für ein Testobjekt und die Sollwerte oder IST-Ergebnisse nach Ausführung des betreffenden Testfalls.
* **Testsuite:** Die Zusammenstellung mehrerer Testfälle für den Test einer Komponente oder eines Systems, bei der Nachbedingungen des einen Tests als Vorbedingungen des folgenden Tests genutzt werden können. Testsuiten können auch dazu genützt werden Daten anzulegen, zu manipulieren und wieder zu löschen um die Datenbank wieder in ihre Ausgangssituation zurück zu setzen.
* **Testszenario:** Die Testfälle oder Testsuiten werden zu ablauffähigen Testszenarien aneinander gereiht. Automatisierte Testszenarien werden auch als Testskript bezeichnet.
* **Testumgebung:** Gesamtheit aller Hardware- und Softwarekomponenten, die notwendig sind, um Testfälle durchzuführen.
* **Fehlernachtest und Regressionstest:**

 Erneuter Test eines bereits getesteten Programms bzw. einer Teilfunktionalität nach deren Modifikation mit dem Ziel, nachzuweisen, dass durch die vorgenommenen Änderungen keine Fehler eingebaut oder (bisher maskierte Fehler) freigelegt wurden.

* + 1. Testaufwand

Vollständiges Testen ist nicht möglich, da alle Einflussgrößen, in allen möglichen Kombinationen, höchstens bei trivialen Funktionen prüfbar sind. Testen kann zwar die Anwesenheit von Fehlern zeigen, nicht aber deren Abwesenheit. Deshalb müssen die Testaktivitäten strategisch geplant werden, um die Testintensität angemessen zu wählen. Bei zu geringer Testintensität entstehen überproportional hohe Fehlerfolgekosten. Bei zu hoher Testintensität sind die Testkosten im Verhältnis zur erwartbaren Reduzierung von Fehlerfolgekosten zu hoch. Die Höhe des Testaufwands soll so bemessen sein, dass die Tests ausreichend Informationen liefern um über die Freigabe der Software entscheiden zu können.

* + 1. Testüberdeckung

Der Überdeckungsgrad stellt die Anzahl der ausgeführten Testfälle im Verhältnis zur Gesamtzahl der Testfälle dar, die nach einem bestimmten Verfahren ermittelt werden. Der Überdeckungsgrad ist ein Kriterium zur Beendigung des Tests, unterschiedlich je nach Testmethode, meist durch Werkzeug zu ermitteln.

**Beispiele**:

* Zu jeder Äquivalenzklasse wird mindestens ein Testfall definiert.
* Zur Erreichung jedes Zustandes des Testobjektes auf jede mögliche Weise wird mindestens ein Testfall definiert.
* Zur Überprüfung jeder bekannten Anforderung an das Testobjekt wird mindestens ein Testfall definiert.
	+ 1. Qualitätssicherungsmaßnahmen (QS)

Es wird zwischen 3 unterschiedlichen QS-Maßnahmen unterschieden

* **Planerische Maßnahmen** 🡺 alle organisatorischen Maßnahmen zur Ein- und Durchführung der QS im Unternehmen und im Softwareprojekt.
* **Konstruktive Maßnahmen** 🡺 Alle präventiven Maßnahmen zur Vermeidung von Fehlern und Mängeln, sowie alle korrigierenden Maßnahmen.
* **Analytische Maßnahmen 🡺** Alle prüfenden Maßnahmen zur Erkennung, Lokalisierung und Bewertung von Fehlern und Mängeln.

Die **analytischen QS-Maßnahmen** können weiter unterteilt werden in:

* **Statische Prüfung/Test** (Formale Verfahren und strukturierte Verfahren)
* **Dynamische Prüfung/Test** (Blackbox- und Whiteboxtests, Dynamische Analyse)

****

Abb. 2‑1 Analytische QS-Maßnahmen

# Softwarequalität

Unter Softwarequalität versteht man die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.



Abb. 3‑1 Qualitätsmerkmale nach ISO 9126

In der **ISO-Norm 9126** werden **Qualitätsmerkmale** für Softwareanwendungen oder - systeme festgelegt. Diese Faktoren sind beim Testen zu berücksichtigen, um die Gesamtqualität eines Softwareproduktes beurteilen zu können. Die einzelnen Qualitätsmerkmale dienen dazu, um geeignete Testmethoden auszuwählen und die Testfälle zu priorisieren. Es muss daher festgelegt werden, welche Qualitätsmerkmale mit welcher Intensität getestet werden sollen.

Es werden nachfolgende Qualitätsmerkmale und Teilmerkmale aufgeführt:

* **Funktionalität:** Fähigkeit der Software, beim Einsatz unter spezifizierten Bedingungen Funktionen zu liefern, die festgelkegte und vorausgesetzten Erfordernisse erfüllen.
	+ **Richtigkeit:** Fähigkeit der Software, die richtigen oder vereinbarten Ergebnisse oder Wirkungen mit dem benötigten Grad an Genauigkeit zu liefern.
	+ **Angemessenheit:** Fähigkeit der Software, für spezifizierte Aufgaben und Anwenderziele einen geeigneten Satz Funktionen zu liefern.
	+ **Interoperabilität:** Fähigkeit der Software, korrekt mit bestimmten anderen Systemen zusammen zu arbeiten.
	+ **Sicherheit:** Fähigkeit der Software, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern.
	+ **Ordnungsmäßigkeit:** Merkmale von Software, die bewirken, dass die Software anwendungsspezifische Normen oder Vereinbarungen oder gesetzliche Bestimmungen und ähnliche Vorschriften erfüllt.
* **Zuverlässigkeit:** Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum oder über eine festgelegte Anzahl von Transaktionen zu bewahren.
	+ **Reife:** Fähigkeit der Software Fehlerwirkungen auf Grundlage von Fehlerzuständen zu vermeiden.
	+ **Fehlertoleranz:** Fähigkeit der Software, ein spezifiziertes Leistungsniveau auch bei Fehlfunktionen oder trotz Fehleingaben aufrecht zu erhalten.
	+ **Wiederherstellbarkeit:** Fähigkeit, bei einem Versagen das spezifizierte Leistungsniveau des Systems wiederherzustellen und die direkt betroffenen Daten wieder zu gewinnen.
* **Benutzbarkeit:** Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe.
	+ **Verständlichkeit:** Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen.
	+ **Erlernbarkeit:** Aufwand für den Benutzer, die Anwendung zu erlernen.
	+ **Bedienbarkeit:** Aufwand für den Benutzer, die Anwendung zu bedienen.
	+ **Attraktivität:** Anziehungskraft der Anwendung gegenüber dem Benutzer.
* **Effizienz:** Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen.
	+ **Zeitverhalten:** Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung.
	+ **Verbrauchsverhalten:** Anzahl und Dauer der benötigten Betriebsmittel bei der Erfüllung der Funktionen. Ressourcenverbrauch, wie CPU-Zeit, Festplattenzugriffe usw.
* **Wartbarkeit:** Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen oder der funktionalen Spezifikationen einschließen.
	+ **Analysierbarkeit:** Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen.
	+ **Modifizierbarkeit:** Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder Anpassung an Umgebungsänderungen.
	+ **Stabilität:** Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen von Änderungen.
	+ **Testbarkeit:** Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen von Änderungen.
* **Übertragbarkeit:** Eignung der Software, von der Umgebung in eine andere übertragen werden zu können. Umgebung kann organisatorische Umgebung, Hardware- oder Software-Umgebung sein.
	+ **Anpassbarkeit:** Fähigkeit der Software, diese an verschiedene Umgebungen anzupassen.
	+ **Installierbarkeit:** Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist.
	+ **Koexistenz:** Fähigkeit der Software neben einer anderen mit ähnlichen oder gleichen Funktionen zu arbeiten.
	+ **Austauschbarkeit:** Möglichkeit, diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden, sowie der dafür notwendige Aufwand.
* **Konformität:** Grad, in dem die Software Normen oder Vereinbarungen erfüllt.

# Organisation von Testteams

Wegen des Problems der „Blindheit gegenüber eigenen Fehlhandlungen“ ist es wesentlich effizienter, Test und Entwicklung personell zu trennen und möglichst unabhängig zu organisieren.

## Psychologie des Testens

* Testen kann als eine „destruktive“ Tätigkeit verstanden werden - **Testen ist aber konstruktiv für das Management von Produktrisiken**
* Ein Tester ist erfolgreich, wenn er Fehler findet!
* Bedingt durch diese Zielsetzung kann kein Entwickler sein eigener Tester sein, da er nach den gleichen Voraussetzungen und Ideen testet, nach denen er die Software entwickelt.
* Zusätzlich besteht noch eine emotionale Bindung: Der Entwickler testet, um die Funktionsfähigkeit der Software nachzuweisen.

🡺 Bedingt durch die Natur des Testens sollen Tester und Entwickler voneinander unabhängig sein. 🡺 **Unabhängiges Testen:**

* Unabhängige Tester finden zusätzliche und andere Fehler als die Entwickler, denn sie arbeiten objektiver
* und können (implizite) Annahmen, die in der Spezifikation oder bei der Programmierung getroffen wurden, kritisch hinterfragen.

## Organisationsformen im Test:

1. Entwickler verantwortlich für Test 🡺 Entwickler testet eigenen Code
2. Team verantwortlich für Test 🡺 Entwickler testen Code gegenseitig
3. Einzeltester im Entwicklungsteam 🡺 ein Tester testet Code mehrerer Entwickler
4. Testconsultants (intern oder extern) 🡺 Ein oder mehrere Tester testen Code mehrerer Entwickler
5. Institution für Tests: Testlabor (intern oder extern) 🡺 mehrere Tester testen Code mehrerer Entwickler mehrerer Projekte
6. Tester aus der Fachabteilung, Anwendergruppe, IT-Abteilung,…🡺 Tester testen gegen ihre spezifische Sicht

****

Abb. 7‑1 Testorganisation

## Rollen

Zur Durchführung der Testarbeiten sollten Spezialisten zur Verfügung stehen, deren Know-how alle Aufgabenbereiche im Testprozess abdeckt. Folgende **Rollen** sind auszufüllen und im Idealfall mit speziell qualifizierten Mitarbeitern zu besetzen:

* + 1. Testmanager

**Typische Aufgaben:**

* Formulierung und Abstimmung der Testpolitik (QS-Handbuch)
* Erstellen und Abstimmen des Testkonzepts
* Vertreten der Testinteressen gegenüber dem Projektmanagement
* Beschaffung der Testressourcen
* Auswahl und Einführung von Testwerkzeugen
* Entscheidung über Art und Umfang der Testumgebung und der Testautomatisierung
* Einführung/Optimierung unterstützender Prozesse zur Rückverfolgbarkeit von Änderungen und zur Sicherstellung der Reproduzierbarkeit der Tests (u.a. Fehlermanagement, Konfigurationsmanagement)
* Einführung, Anwendung und Auswertung der im Testkonzept definierten Metriken
* Regelmäßiges Anpassen der Testpläne auf Basis gewonnener Testergebnisse und des Testfortschritts
* Erstellen und Kommunizieren von Testberichten

**Qualifikation:**

* Certified Tester Advanced Level – Test Manager
* Experte für Testplanung und Teststeuerung
* Know-how/Erfahrung in den Bereichen
* Softwaretest
* Qualitätsmanagement
* Projektmanagement
* Personalführung
	+ 1. Testanalyst (Testdesigner)

**Typische Aufgaben:**

* Review von Requirements
* Review von Spezifikationen und Systemmodellen
* Testfallanalyse
* Erstellung von Testspezifikationen
* Ermittlung und Aufbereitung von Testdaten

**Qualifikation:**

* Certified Tester Advanced Level – Test Analyst
* Experte für Testmethoden und Testspezifikationen
* Know-how/Erfahrung in den Bereichen
* Softwaretest
* Softwareengineering
* (formale) Spezifikationsmethoden
	+ 1. Technical Testanalyst

**Typische Aufgaben:**

* Automatisiert spezifizierte Tests unter Nutzung der im Projekt vorhandenen Toolumgebung

**Qualifikation:**

* Certified Tester Advanced Level – Technical Test Analyst
* Experte für Testautomatisierung
* Testgrundlagenwissen
* Programmiererfahrung
* Sehr gute Kenntnisse der eingesetzten Testwerkzeuge und Skriptsprachen
	+ 1. Tester

**Typische Aufgaben:**

* Review von Testplänen und Testfällen
* Anwendung von Testtools und Testüberwachungstools
* Ausführung und Protokollierung von Tests mit Auswertung und Dokumentation der Resultate und festgestellter Abweichungen

**Qualifikation:**

* Certified Tester Foundation Level
* IT-Grundlagenwissen
* Testgrundlagenwissen:
* Bedienung der eingesetzten Testwerkzeuge
* Verständnis des Testobjekts
	+ 1. Testadministrator (optional)

**Typische Aufgaben:**

* + - Aufsetzen und Betreuung der Testumgebung

**Qualifikation:**

* + - Experte für Installation und Betrieb der Testumgebung (Systemadministrator-Know-how)
		1. Keyuser (EDV-Koordinatoren/Fachbereich)

**Typische Aufgaben des Keyusers:**

* Der Keyuser hat die Aufgabe, die Einführung der Software in der eigenen Abteilung zu unterstützen.
* Er ist Ansprechpartner für die Kollegen in der eigenen Abteilung und dem Projektleiter.
* Er vertritt die fachlichen Interessen des Fachbereiches.

**Vorteile Fachbereichsmitarbeiter in den Test zu integrieren:**

* fachliche Knowhowträger
* idealer Trainer für Anwenderschulungen
* späterer Ansprechpartner in der Anwendung
* Motivator für die Veränderung, die sich durch die Software ergibt

## Soziale Kompetenz

Neben seinen technischen und testspezifischen Fähigkeiten benötigt ein **Softwaretester**, um erfolgreich zu sein, auch **soziale Kompetenz**:

* Disziplin und strukturiertes Vorgehen
* Teamfähigkeit
* Gutes Kommunikationsvermögen
* Durchsetzungskraft
* diplomatisches Geschick
* Bereitschaft, scheinbare Tatsachen zu hinterfragen
* Exaktheit und Kreativität
* Fähigkeit, sich schnell in komplexe Anwendungsgebiete und Applikationen einzuarbeiten
* Sachlicher und lösungsorientierter Kommunikationsstil
* Professioneller Pessimismus

# Testwerkzeuge & Automatisierung

Ein Testwerkzeug ist eine Art von Software, die bei Testaufgaben unterstützt. Es kann ein Programm, ein Skript, eine Datenbank oder sogar ein Worksheet-Dokument sein. Der Zweck des Testwerkzeugs besteht darin, die Effizienz des Tests zu erhöhen und den Informationsverlust möglichst gering zu halten.

## Testwerkzeugkonzepte

Testwerkzeuge können nach unterstützten Testaktivitäten oder folgenden Kriterien klassifiziert werden:

* **Teststufe:**

Wo wird getestet?

* Komponententest
* Integrationstest
* Systemtest
* Abnahmetest
* **Fehlern:**

Welche Fehler werden gesucht, gefunden, berücksichtigt…

* Laufzeitfehler
* Funktionale Fehler
* Nichtfunktionale Fehler
* Anomalien, Richtlinienverletzungen …
* **Testverfahren:**

Welche werden unterstützt?

* Äquivqlenzklassenanalyse
* Zustandsbasiertes Testen
* Datenflussbasiertes Testen
* …
* **Zielsetzung:**

Worum geht es?

* Messung
* Treiber
* Protokollierung
* Vergleiche …
* **Fachlichem Inhalt:**

Welche Domäne wird adressiert?

* Netzwerke, Protokolle
* Transaktionen
* Expertensysteme …
* **Aufgabenbereich im Test:**

Wo im Test unterstützt das Tool:

* Dateneingabe
* Umgebung
* Konfiguration
* …
* **Einsatzweise:**

Wie wird das Werkzeug angewandt?

* Off the shelf (Standardsoftware)
* Framework (noch anzupassen)
* Plug-in (z.B. Eclipse)
* Eigenentwicklung des Werkzeugs
* **User:**

Wer setzt das Werkzeug ein?

* Testmanager
* Testanalyst
* Technical Testanalyst
* …

## Testwerkzeugkategorien

* **Testmanagementwerkzeuge**

Testmanagementwerkzeuge ermöglichen:

* Rückverfolgbarkeit von Testartefakten
* Erfassung der Testumgebungsdaten
* Testorganisation:
	+ Wo werden eventuell zur gleichen Zeit auf verschiedenen Umgebungen an verschiedenen Orten gleiche Testsuiten durchgeführt?
* Erfassung von Metriken:
	+ Testbedingungen
	+ Testfälle
	+ Ausführungsdauer (je nach Einheit)
	+ Anzahl der Testfälle, Testartefakte, Umgebungen …
	+ Verhältnis bestandener/nicht bestandener Tests
	+ Anzahl offener Testfälle (differenziert nach Ursache)
	+ Tendenzen/Trends
	+ Beziehungen zwischen Testartefakten
* Zentrale Verwaltungsaufgaben:
	+ Organisation und Verwaltung von Testartefakten, Testdatenbank/Repository, Treiber …
	+ Testbedingungen und Testumgebungen (und Neuinitialisierung)
	+ Metriken über die Testartefakte (im Rahmen der Testdokumentation), um den Testfortschritt zu dokumentieren
* **Testausführungswerkzeuge (Capture-Replay, Testroboter)**

Testausführungswerkzeuge ermöglichen:

* Ausführung von Tests (idealerweise) ohne manuelle Eingriffe
* Kontrolle der Ergebnisse
* Reduzierung von Kosten und Aufwand
* Ausführung einer erhöhten Testfallmenge
* Wiederholbarkeit der Tests
* Automatisierte Durchführung von Testfällen, dabei
	+ Wiedergabe der Aktionen
	+ Eingabe der Testdaten
	+ Vergleich der Ist- mit der Sollreaktion
	+ Protokollierung der Testdurchführung
* **Schlüsselwort-getriebene Fehleranalysewerkzeuge**

Schlüsselwort-getriebene Testautomatisierung ermöglicht:

* Definition von Schlüsselwörtern und darauf aufbauenden Testfällen in der Sprache der Fachexperten
* Dadurch sind Testfälle durch Fachexperten erstellbar und /oder wartbar
* Arbeitsteilung je nach Qualifikation/Schwerpunkt
* Wartbare Testautomatisierung auf Basis dieser Spezifikationen
* Unabhängigkeit zwischen logischen Testfällen, Testdaten und der Implementierung der Testfälle
* **Fehlereinpflanzungs- & Fehleranalysewerkzeuge**

Fehlereinpflanzungswerkzeuge werden eingesetzt

* um die Qualität der Modultests zu bewerten
* manipulieren Code, dass Fehler oder Ausfälle entstehen
* erstellen systematisch eine begrenzte Anzahl, oder bestimmte Arten von Fehlern in fehlerfreien Codeabschnitten
* dienen zur Überprüfung der Fehlertoleranz der Software
* schaffen Vertrauen in die Testfälle und in die Testvorgehensweise, wenn die eingepflanzten Fehler durch die Testfälle gefunden werden
* **Simulations- & Emulations-Werkzeuge**

Simulations- und Emulationswerkzeuge unterstützen Tests, deren Durchführung in der Einsatzumgebung oder mit dem Zielsystem nicht möglich, zu aufwändig oder zu gefährlich ist.

* häufig individuell erstellte Werkzeuge
* werden in der Regel über Schnittstellen vom Testobjekt angesprochen
* Können teils Fehlersituationen nachbilden
* ersetzen häufig technische Systeme, um einen frühzeitigen Test zu ermöglichen (Simulation von Sensoren und Aktoren)

Emulatoren sind eine Spezielle Art von Simulatoren:

* Emulatoren bilden Hardware durch Software nach
* Komplexere Tests werden möglich, da HW-Verhalten gezielt beeinflussbar ist (z.B. defektes Register)
* Teils sind zeitabhängige Aspekte gezielt modellierbar, wenn auch die Uhr/der Takt der Hardware im Emulator steuerbar ist
* **Statische und dynamische Analysewerkzeuge**

Statische Codeanalysatoren liefern Hinweise auf Fehlerquellen wie z.B.:

* nicht initialisierte Variablen
* falsche Speicherzugriffe

aber auch

* Maßzahlen zu verschiedenen Charakteristika des Programmcodes
* Komplexe und damit fehleranfällige bzw. risikoreiche Codeabschnitte
* Informationen über die Einhaltung von Programmierrichlinien

Dynamische Analysatoren ermitteln während der Programmausführung zusätzliche Informationen:

* Belegung, Verwendung und Freigabe von Speicher

Überdeckunganalysatoren liefern

* Maßzahlen der strukturellen Testabdeckung auf Codeebene
* **Web-Werkzeuge**

Web-Werkzeuge dienen zur Analyse von Websites.

* Sammlung von Informationen über Benutzeranzahl und -verhalten
* Graphen zur Baumstruktur der Webseite
* Überprüfen auf unterbrochene Verlinkungen einer Website
* Überwachen Websites auf Verfügbarkeit und warnen bei Nichtverfügbarkeit der überwachten Dienste
* Geschwindigkeit von Downloads, durchschnittliche Datenmengen …
* Überwachung von Service-Level (SLA)

# Normen und Standards

Normen und Standards definieren allgemein anerkannte Regeln der Technik und damit den fachlich-rechtlichen Rahmen, innerhalb dessen sich Projekte abspielen. Sie fördern die Einheitlichkeit von Produkten und Prozessen und sind die Minimalanforderungen für professionelles Arbeiten.

Nachfolgend eine Liste relevanter internationaler Standards und Normen:

* BS 7925-1:1998, *Software testing - Vocabulary*
* BS 7925-2:1998, *Software testing - Software component testing*
* IEC 60300-3-9:1995, *Risk Analysis of technological systems*
* IEEE Std 601.12–1990*, IEEE Standard Glossary of Software Engineering Terminology*
* IEEE Std 829*–*2008, *IEEE Standard for Software and System Test Documentation*
* IEEE Std 1008*–*1987, *IEEE Standard for Software Unit Testing*
* IEEE Std 1012*–*2012, *IEEE Standard for System and Software Verification and Validation*
* IEEE Std 1028*–*2008, *IEEE Standard for Software Reviews and Audits*
* ISO 15489-1:2001, *Information and documentation -- Records management -- Part 1: General*
* ISO 31000:2009, *Risk management -- Principles and guidelines*
* ISO 9001:2008, *Quality management systems -- Requirements*
* ISO/IEC 12207:2008, *Systems and software engineering – Software life cycle processes*
* ISO/IEC 16085:2006, *Systems and software Engineering – Life cycle Processes – Risk Management*
* ISO/IEC 25000:2005, *Software Engineering -- Software product Quality Requirements and Evaluation*

 *(SQuaRE) -- Guide to SQuaRE*

* ISO/IEC 25010-2011, *Systems and Software Engineering – Systems and software Quality Requirements and*

 *Evaluation (SQuaRE) – System and software quality models*

* ISO/IEC 25051:2006, *Software engineering — Software product Quality Requirements and Evaluation*

 *(SQuaRE) — Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing*

* ISO/IEC 90003:2004, *Software engineering — Guidelines for the application of ISO 9001:2000 to computer software*
* ISO/IEC/IEEE 24765:2010, *Systems and software engineering -- Vocabulary*
* ISO/IEC DIS 29119*, Software and Systems Engineering — Software Testing* (in Entwicklung)

**Quellen:**

* **ISTQB\_/GTB Standardglossar der Testbegriffe**

**Deutsch/Englisch**

**Herausgeber: German Testing Board e.V.**

**Dr. Matthias Hamburg, Dr. Uwe Hehn**

[**http://www.software-tester.ch/PDF-Files/CT\_Glossar\_DE\_EN\_V21.pdf**](http://www.software-tester.ch/PDF-Files/CT_Glossar_DE_EN_V21.pdf)

* **ISTQB Lehrplan**
* **Andreas Spillner/Tilo Linz
Basiswissen Softwaretest
Aus- und Weiterbildung zum Certified Tester
Foundation Level nach ISTQB-Standard**
* **Graham Bath / Judy McKay**

**Praxiswissen Softwaretest – Test Analyst und Technical Test Analyst**

**Aus- und Weiterbildung zum Certified Tester
Advanced Level nach ISTQB-Standard**

* **Andreas Spillner/Thomas Roßner/Mario Winter/Tilo Linz**

**Praxiswissen Softwaretest – Testmanagement**

**Aus- und Weiterbildung zum Certified Tester
Advanced Level nach ISTQB-Standard**

* **Wikipedia**

[**http://de.wikipedia.org**](http://de.wikipedia.org)

[**http://de.wikiversity.org/wiki/Kurs:Software-Test**](http://de.wikiversity.org/wiki/Kurs%3ASoftware-Test)