# Softwaretest

Ein **Softwaretest** ist ein Test, der im Rahmen der Softwareentwicklung durchgeführt wird. Er bewertet die Funktionalität der Software gemäß ihrer Anforderungen und misst ihre Qualität. Die aus dem Softwaretest gewonnenen Erkenntnisse werden zur Behebung und Vermeidung von Softwarefehlern herangezogen.

Von diesem, *einzelne* Testdurchführungen bezeichnenden Begriff ist die gleich lautende Bezeichnung 'Test' (auch 'Testen') zu unterscheiden, unter der die *Gesamtheit* der Maßnahmen zur Überprüfung der Softwarequalität (inkl. Planung, Vorbereitung, Steuerung, Durchführung, Dokumentation usw.; siehe auch Definitionen) verstanden wird.

Den Nachweis, dass keine Fehler (mehr) vorhanden sind, kann das Softwaretesten nicht erbringen, es kann lediglich feststellen, dass bestimmte Testfälle erfolgreich waren. E. W. Dijkstra hierzu: *Program testing can be used to show the presence of bugs, but never show their absence!* (Programmtesten kann angewandt werden, um die Existenz von Fehlern zu zeigen, aber niemals deren Abwesenheit.) Der Grund ist, dass alle Programmfunktionen und auch alle möglichen Werte in den Eingabedaten in allen ihren Kombination getestet werden müssten – was (außer bei sehr einfachen Testobjekten) praktisch nicht möglich ist. Aus diesem Grund beschäftigen sich verschiedene Teststrategien und -konzepte mit der Frage, wie mit einer möglichst geringen Anzahl von Testfällen eine große Testabdeckung zu erreichen ist.

Pol, Koomen, Spillner [1] zu Testen: Tests sind nicht die einzige Maßnahme im Qualitätsmanagement der Softwareentwicklung, aber oft die letztmögliche. Je später Fehler entdeckt werden, desto aufwändiger ist ihre Behebung, woraus sich der Umkehrschluss ableitet: Qualität muss (im ganzen Projektverlauf) implementiert und kann nicht 'eingetestet' werden. Beim Testen in der Softwareentwicklung wird i. d. R. eine mehr oder minder große Fehleranzahl als 'normal' unterstellt oder akzeptiert. Hier herrscht ein erheblicher Unterschied zur Industrie: Dort werden im Prozessabschnitt 'Qualitätskontrolle' oft nur noch in Extremsituationen Fehler erwartet.

## Definition

Es gibt unterschiedliche Definitionen für den Softwaretest:

Nach ANSI/IEEE Std. 610.12-1990 ist Test *„the process of operating a system or component under specified conditions, observing or recording the results and making an evaluation of some aspects of the system or component.“*

Eine andere Definition liefert Denert[2] , wonach der *„Test […] der überprüfbare und jederzeit wiederholbare Nachweis der Korrektheit eines Softwarebausteines relativ zu vorher festgelegten Anforderungen“* ist.

Eine weitergehende Definitionen verwenden Pol, Koomen und Spillner in [1] : *Unter Testen versteht man den Prozess des Planens, der Vorbereitung und der Messung, mit dem Ziel, die Eigenschaften eines IT-Systems festzustellen und den Unterschied zwischen dem tatsächlichen und dem erforderlichen Zustand aufzuzeigen.* Bemerkenswert hierbei: Als Messgröße gilt 'der erforderliche Zustand', nicht nur die (möglicherweise fehlerhafte) Spezifikation.

'Testen' ist ein wesentlicher Teil im Qualitätsmanagement von Projekten der Softwareentwicklung.

## Ziele

*Globales Ziel* des Softwaretestens ist das *Messen der Qualität des Softwaresystems*. Dabei werden definierte Anforderungen überprüft und dabei ggf. vorhandene Fehler aufgedeckt. ISTQB: *Der Wirkung von Fehlern (im produktiven Betrieb) wird damit vorgebeugt*.

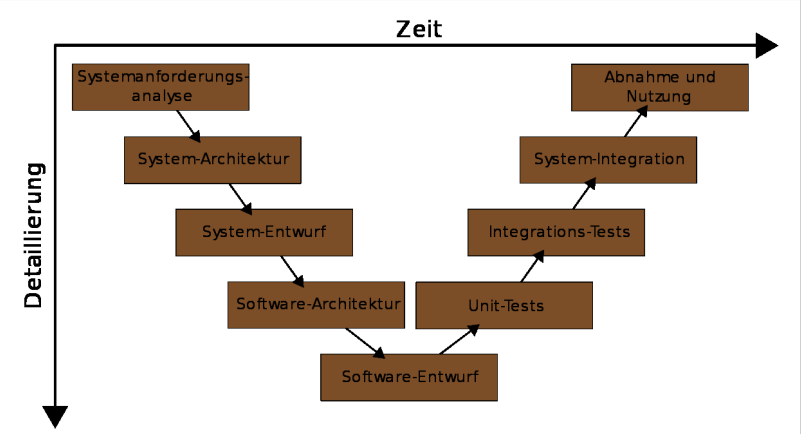
Ein Rahmen für diese Anforderungen können die *Qualitätsparameter gem. ISO/IEC 9126* sein, denen jeweils konkrete Detailanforderungen z. B. zur Funktionalität, Bedienbarkeit, Sicherheit usw. zugeordnet werden können. Im Besonderen ist auch die Erfüllung gesetzlicher und / oder vertraglicher Vorgaben nachzuweisen.

Die Testergebnisse (die über verschiedene Testverfahren gewonnen werden) tragen zur Beurteilung der realen Qualität der Software bei – als Voraussetzung für deren Freigabe zum operativen Betrieb. *Das Testen soll Vertrauen in die Qualität der Software schaffen* [1] .

*Individuelle Testziele:* Da das Softwaretesten aus zahlreichen Einzelmaßnahmen besteht, die i. d. R. über mehrere Teststufen hinweg und an vielen Testobjekten ausgeführt werden, ergeben sich individuelle Testziele für jeden einzelnen Testfall und für jede Teststufe – wie z. B. Rechenfunktion X in Programm Y getestet, Schnittstellentest erfolgreich, Wiederinbetriebnahme getestet, Lasttest erfolgreich, Programm XYZ getestet usw.

## Teststufen

Die Einordnung der Teststufen (auch Testzyklen genannt) folgt dem Entwicklungsstand des Systems. Sie orientieren sich in der Regel an den Entwicklungsstufen von Projekten nach dem V-Modell. Dabei wird in jeder Teststufe (rechte Seite im 'V') gegen die Systementwürfe und Spezifikationen der zugehörigen Entwicklungsstufe (linke Seite) getestet.



Stufen des V-Modells

In der Realität werden diese Ausprägungen, abhängig von der Größe und Komplexität des Software-Produkts, weiter untergliedert. So könnten beispielsweise die Tests für die Entwicklung von sicherheitsrelevanten Systemen in der Transportsicherungstechnik folgendermaßen untergliedert sein: Komponententest auf dem Entwicklungsrechner, Komponententest auf der Ziel-Hardware, Produkt-Integrationstests, Produkttest, Produkt-Validierungstests, System-Integrationstest, Systemtest, System-Validierungstests, Feldtests und Akzeptanztest.

### Komponententest

Der Modultest, auch *Komponententest* oder *Unittest* genannt, ist ein Test auf der Ebene der einzelnen Module der Software. Testgegenstand ist die Funktionalität innerhalb einzelner abgrenzbarer Teile der Software (Module, Programme oder Unterprogramme, Units oder Klassen). Testziel dieser häufig durch den Softwareentwickler selbst durchgeführten Tests ist der Nachweis der technischen Lauffähigkeit und korrekter fachlicher (Teil-) Ergebnisse.

### Integrationstest

Der Integrationstest bzw. Interaktionstests testet die Zusammenarbeit voneinander abhängiger Komponenten. Der Testschwerpunkt liegt auf den Schnittstellen der beteiligten Komponenten und soll korrekte Ergebnisse über komplette Abläufe hinweg nachweisen.

### Systemtest

Der **Systemtest** ist die Teststufe, bei der das gesamte System gegen die gesamten Anforderungen (funktionale und nicht funktionale Anforderungen) getestet wird. Gewöhnlich findet der Test auf einer Testumgebung statt und wird mit Testdaten durchgeführt. Die Testumgebung soll die Produktivumgebung des Kunden simulieren. In der Regel wird der Systemtest durch die realisierende Organisation durchgeführt.

### Abnahmetest

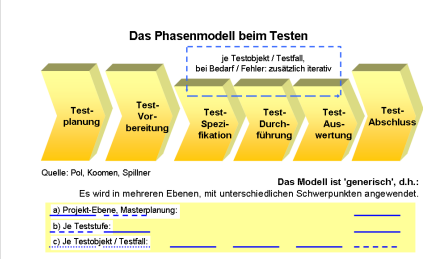
Ein **Abnahmetest**, **Verfahrenstest**, **Akzeptanztest** oder auch **User Acceptance Test** (UAT) ist ein Test der gelieferten Software durch den Kunden bzw. Auftraggeber. Oft sind Akzeptanztests Voraussetzung für die Rechnungsstellung. Dieser Test kann bereits auf der Produktivumgebung mit Kopien aus Echtdaten durchgeführt werden.

Die vorgenannten Teststufen sind in der Praxis oft nicht scharf voneinander abgegrenzt, sondern können, abhängig von der Projektsituation, fließend oder über zusätzliche Zwischenstufen verlaufen. So könnte zum Beispiel die Abnahme des Systems auf der Grundlage von Testergebnissen (Reviews, Testprotokolle) von Systemtests erfolgen.

Besonders für System- und Abnahmetests wird das Blackbox-Verfahren angewendet, d. h. der Test orientiert sich nicht am Code der Software, sondern nur am Verhalten der Software bei spezifizierten Handlungen (Eingaben des Benutzers, Grenzwerte bei der Datenerfassung, etc.).

## Testprozess / Testphasen

Pol, Koomen und Spillner [1] empfehlen ein Vorgehen nach dem in der Grafik dargestellten *Phasenmodell*. Sie nennen dieses Vorgehen *Testprozess*, die Einzelschritte *Testphasen* und unterscheiden dabei: Testplanung, Testvorbereitung, Testspezifikation, Testdurchführung, Testauswertung, Testabschluss



Der generische Testprozess und seine Testphasen

Dieses Vorgehen ist *generisch*, d.h. es wird - jeweils nach Erfordernis - für unterschiedliche Ebenen angewendet, nämlich für das Gesamtprojekt, für jede Teststufe und letztlich auch je Testobjekt und Testfall. Die in diesen Ebenen i. d. R. anfallende Arbeitsintensität ist in der Grafik durch Punkte (= gering), Striche (= mittel) und durchgehende Linien (= Schwerpunkt) dargestellt.

Testaktivitäten werden (nach Pol, Koomen und Spillner [1] ) rollenspezifisch zu sog. *Testfunktionen* zusammengefasst: Testen, Testmanagement, Methodische Unterstützung, Technische Unterstützung, Funktionale Unterstützung, Verwaltung, Koordination und Beratung, Anwendungsintegrator, TAKT-Architekt und TAKT-Ingenieur (bei Einsatz von Testautomatisierung; TAKT = Testen, Automatisierung, Kenntnisse, Tools). Diese

Funktionen (Rollen) haben Schwerpunkte in bestimmten Testphasen; sie können im Projekt selbst eingerichtet sein oder über spezialisierte Organisationseinheiten einbezogen werden.

Bei *anderen Autoren oder Instituten* finden sich zum Teil andere Gruppierungen und andere Bezeichnungen, die aber inhaltlich nahezu identisch sind. Z. B. wird bei ISTQB der "fundamentale Testprozess" mit folgenden Hauptaktivitäten definiert:
Testplanung und Steuerung, Testanalyse und Testdesign, Testrealisierung und Testdurchführung, Testauswertung und Bericht, Abschluss der Testaktivitäten

## Testplanung

Ergebnis dieser i. d. R. parallel zur Softwareentwicklung stattfindenden Phase ist i. W. der *Testplan*. Er wird für jedes Projekt erarbeitet und soll den gesamten Testprozess definieren. In TMap wird dazu ausgeführt: *Sowohl die zunehmende Bedeutung von IT-Systemen für Betriebsprozesse als auch die hohen Kosten des Testens rechtfertigen einen optimal verwaltbaren und strukturierten Testprozess.* Der Plan kann und soll je Teststufe aktualisiert und konkretisiert werden, sodass die Einzeltests im Umfang zweckmäßig und effizient ausgeführt werden können.

*Inhalte* im Testplan sollten z. B. folgende Aspekte sein: Teststrategie (Testumfang, Testabdeckung, Risikoabschätzung); Testziele und Kriterien für Testbeginn, Testende und Testabbruch - für alle Teststufen; Vorgehensweise (Testarten); Hilfsmittel und Werkzeuge; Dokumentation (Festlegen der Art, Struktur, Detaillierungsgrad); Testumgebung (Beschreibung); Testdaten (allgemeine Festlegungen); Testorganisation (Termine, Rollen), alle Ressourcen, Ausbildungsbedarf; Testmetriken; Problemmanagement

## Testvorbereitung

Aufbauend auf der Testplanung werden die dort festgelegten Sachverhalte zur operativen Nutzung vorbereitet und zur Verfügung gestellt.

*Beispiele* für einzelne Aufgaben (global und je Teststufe): Bereitstellen der Dokumente der Testbasis; Verfügbar machen (z.B. Customizing) von Werkzeugen für das Testfall- und Fehlermanagement; Aufbauen der Testumgebung(en) (Systeme, Daten); Übernehmen der Testobjekte als Grundlage für Testfälle aus der Entwicklungsumgebung; Benutzer und Benutzerrechte anlegen; ...
Beispiele für Vorbereitungen (für Einzeltests): Transfer / Bereitstellung von Testdaten bzw. Eingabedaten in die Testumgebung(en).

## Testspezifikation

Hier werden alle Festlegungen und Vorbereitungen getroffen, die erforderlich sind, um einen bestimmten Testfall ausführen zu können.

*Beispiele* für einzelne Aktivitäten: Testfallfindung und Testfalloptimierung; Beschreiben je Testfall (was genau ist zu testen); Vorbedingungen (incl. Festlegen von Abhängigkeiten zu anderen Testfällen); Festlegen und Erstellen der Eingabedaten; Festlegungen zum Testablauf und zur Testreihenfolge; Festlegen Soll-Ergebnis; Bedingung(en) für 'Test erfüllt'; ...

### Testdurchführung

Bei dynamischen Tests wird dazu das zu testende Programm ausgeführt, in statischen Tests ist es Gegenstand analytischer Prüfungen.

*Beispiele* für einzelne Aktivitäten: Auswählen der zu testenden Testfälle; Starten des Prüflings - manuell oder automatisch; Bereitstellen der Testdaten und des Ist-Ergebnisses zur Auswertung; Umgebungsinformationen für den Testlauf archivieren, ...

Weitere Anmerkung: Ein Testobjekt sollte nicht vom Entwickler selbst, sondern von anderen, wenn möglich unabhängigen, Personen getestet werden.

### Testauswertung

Die Ergebnisse aus durchgeführten Tests (je Testfall) werden überprüft. Dabei wird das Ist-Ergebnis mit dem Soll-Ergebnis verglichen und anschließend eine Entscheidung über das Testergebnis (ok oder Fehler) herbeigeführt.

- Bei Fehler: Klassifizierung (z. B. nach Fehlerursache, Fehlerschwere etc.), angemessene Fehlerbeschreibung und -Erläuterung, Überleitung ins Fehlermanagement; Testfall bleibt offen
- Bei OK: Testfall gilt als erledigt
- Für alle Tests: Dokumentation, Historisieren / Archivieren von Unterlagen

### Testabschluss

Abschluss-Aktivitäten finden auf allen Testebenen statt: Testfall, Testobjekt, Teststufe, Projekt. Der Status zum Abschluss von Teststufen wird (z. B. mit Hilfe von Teststatistiken) dokumentiert und kommuniziert, Entscheidungen sind herbeizuführen und Unterlagen zu archivieren. Grundsätzlich ist dabei zu unterscheiden nach:

- Regel-Abschluss = Ziele erreicht, nächste Schritte einleiten
- Alternativ möglich: Teststufe ggf. vorzeitig beenden oder unterbrechen (aus diversen, zu dokumentierenden Gründen); in Zusammenarbeit mit dem Projektmanagement

## Klassifikation für Testarten

In kaum einer Disziplin der Softwareentwicklung hat sich, der Komplexität der Aufgabe 'Testen' entsprechend, eine derart große Vielfalt an Begriffen für Verfahrensansätze gebildet wie beim Softwaretest. Dies beginnt bereits bei den *Typ-Ausprägungen* für Testvarianten, die mit Begriffen wie Teststufe, Testzyklus, Testphase, Testart, Testtyp, Testmethode, Testverfahren ... bezeichnet werden.

Die Bezeichnung *konkreter Testarten* leitet sich meistens aus ihren individuellen Zielen und Charaktermerkmalen ab - wodurch sich eine Vielzahl an Bezeichnungen ergibt. Dieser Vieldimensionalität entsprechend können für einen konkreten Test die Bezeichnungen mehrerer Testarten zutreffen. Beispiel: *Entwicklertest, dynamischer Test, Blackbox-Test, Fehlertest, Integrationstest, Äquivalenzklassentest, Batchtest, Regressionstest*. Durchaus im Sinn effizienter Testprozesse ist es, mehrere Testfälle mit nur einem konkreten Test abzudecken, z. B. eine technische Datenschnittstelle, die Prüfung korrekter Wertebereiche und eine Rechenformel.

Die nachfolgend beschriebenen Testart-Bezeichnungen sind Beispiele aus der Literatur. Im praktischen Einsatz werden aber viele dieser Bezeichnungen nicht verwendet, sondern (zum Beispiel) einfach als 'Funktionstest' bezeichnet und nicht als Fehlertest, Batchtest, High-Level-Test etc. Die Testeffizienz wird hierdurch nicht beeinträchtigt - wenn die Tests ansonsten angemessen geplant und ausgeführt werden. Die nachfolgenden Aufzählungen können auch eine Vorstellung davon vermitteln, was beim Testen, vor allem in der Testplanung berücksichtigt werden sollte oder könnte.

Ein Mittel zum Verständnis dieser Begriffsvielfalt ist die nachfolgend angewendete *Klassifikation* - bei der Testarten nach unterschiedlichen Kriterien gegliedert werden.

## Klassifikation nach der Prüftechnik

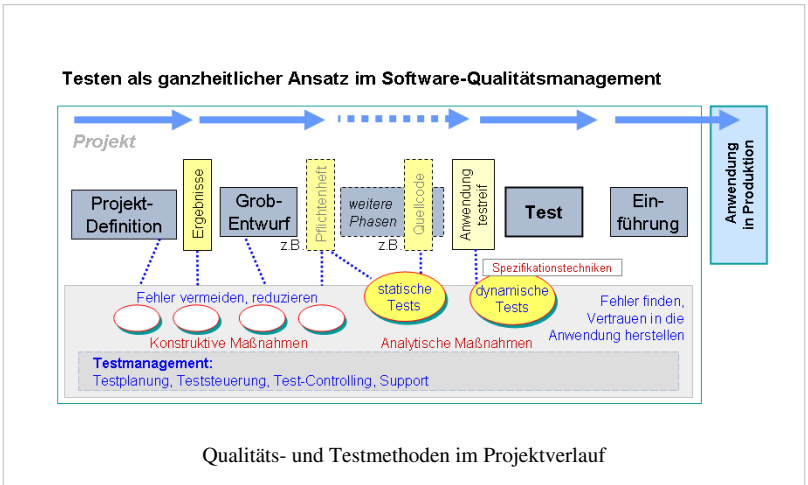
Analytische Maßnahmen: Softwaretests werden oft als analytische Maßnahmen, die erst nach Erstellung des Prüfgegenstandes durchgeführt werden können, definiert. Liggesmeyer [3] klassifiziert diese Testmethoden folgendermaßen (verkürzt und z. T. kommentiert):

**Statischer Test** (Test ohne Programmausführung)

- Review
- Statische Code-Analyse, auch Formale Verifikation

**Dynamischer Test** (Test während Programmausführung)

- Strukturorientierter Test
  - Kontrollflussorientiert (Maß für die Überdeckung des Kontrollflusses)
    - Anweisungs-, Zweig-, Bedingungs- und Pfadüberdeckungstests
  - Datenflussorientiert (Maß für die Überdeckung des Datenflusses)
    - Defs-/Uses Kriterien, Required k-Tupels-Test, Datenkontext-Überdeckung
- Funktionsorientierter Test (Test gegen eine Spezifikation)
  - Funktionale Äquivalenzklassenbildung, Zustandsbasierter Test, Ursache-Wirkung-Analyse z. B. mittels Ursache-Wirkungs-Diagramm, Syntaxtest, Transaktionsflussbasierter Test, Test auf Basis von Entscheidungstabellen
  - Positivtest (versucht die Anforderungen zu verifizieren) und Negativtest (prüft die Robustheit einer Anwendung)
- Diversifizierender Test (Vergleich der Testergebnisse mehrerer Versionen)
  - Regressionstest, Back-To-Back-Test, Mutationen-Test
- *Sonstige* (nicht eindeutig zuzuordnen, bzw. Mischformen)
  - Bereichstest bzw. Domain Testing (Verallgemeinerung der Äquivalenzklassenbildung), Error guessing, Grenzwertanalyse, Zusicherungstechniken




Qualitäts- und Testmethoden im Projektverlauf

Konstruktive Maßnahmen: Diesen analytischen Maßnahmen, bei denen Testobjekte "geprüft" werden, gehen die sog. konstruktiven Maßnahmen voraus, die bereits im Verlauf der Software-Erstellung zur Qualitätssicherung betrieben werden. Beispiele: Anforderungsmanagement, Prototyping, Review von Pflichtenheften.

Spezifikationstechniken: Weiterhin sind von den Prüftechniken die Spezifikationstechniken zu unterscheiden: Sie bezeichnen keine Testarten, mit denen Testobjekte aktiv geprüft werden, sondern nur die Verfahren, nach denen die Tests vorbereitet und spezifiziert werden.
Beispiele Äquivalenzklassentest und Überdeckungstest: Testfälle werden nach diesen Verfahren identifiziert und spezifiziert, konkret überprüft jedoch (z. B.) im Integrationstest, Batchtest, Sicherheitstest etc.

### Klassifikation nach dem Testkriterium

Die Einordnung erfolgt hier je nachdem welche inhaltlichen Aspekte getestet werden sollen.

Funktionale Tests bzw. Funktionstests

: überprüfen ein System in Bezug auf funktionale Anforderungsmerkmale wie Korrektheit und Vollständigkeit.

Nicht-funktionale Tests

: überprüfen die nicht funktionalen Anforderungen, wie z. B. die Sicherheit, die Gebrauchstauglichkeit oder die Zuverlässigkeit eines Systems. Dabei steht nicht die Funktion der Software (Was tut die Software?) im Vordergrund, sondern ihre Funktionsweise (Wie arbeitet die Software?).

Schnittstellentests

: testen die Funktionalität bei der Zusammenarbeit voneinander unabhängiger Komponenten (für jede der beteiligten Komponente, anhand der Spezifikation, beispielsweise mit Hilfe von Mock-Objekten)

Fehlertests

: testen, ob die Verarbeitung von Fehlersituationen korrekt, d.h. wie definiert erfolgt.

Datenkonsistenztests

: testen die Auswirkung der getesteten Funktion auf die Korrektheit von Datenbeständen (Testbezeichnungen: Datenzyklustest, Wertebereichstest, Semantiktest, CRUD-Test)

Wiederinbetriebnahmetests

: testen ob ein System nach einem Abschalten oder Zusammenbruch (z. B. ausgelöst durch Stresstest) wieder in Betrieb genommen werden kann.

Interoperabilitätstests

: testen die Funktionalität bei der Zusammenarbeit voneinander unabhängiger Komponenten unter Einsatz mehrerer Komponenten.

Installationstests

: testen die Softwareinstallationsroutinen, ggfs. in verschiedenen Systemumgebungen (z. B. mit verschiedener Hardware oder unterschiedlichen Betriebssystemversionen)

Oberflächentests

: testen die Benutzerschnittstelle des Systems (z. B. Verständlichkeit, Anordnung von Informationen, Hilfefunktionen); für Dialogprogramme auch *GUI-Test oder UI-Test* genannt.

Stresstests

: sind Tests, die das Verhalten eines Systems unter Ausnahmesituationen prüfen und analysieren.

Crashtests

: sind Stresstests, die versuchen, das System zum Absturz zu bringen.

Lasttests

: sind Tests, die das Systemverhalten unter besonders hohen Speicher-, CPU-, o. ä. -Anforderungen analysieren. Besondere Arten von Last-Tests können Multi-User-Tests (viele Anwender greifen auf ein System zu, simuliert oder real) und Stresstests (dabei wird das System an die Grenzen der Leistungsfähigkeit geführt) sein.

Performance Tests

: sind Tests, die ein korrektes Systemverhalten bei bestimmten Speicher- und CPU-Anforderungen sicherstellen sollen.

Rechnernetz-Tests

testen das Systemverhalten in Rechnernetzen (z. B. Verzögerungen der Datenübertragung, Verhalten bei Problemen in der Datenübertragung).

Sicherheitstests

testen ein System gegen potentielle Sicherheitslücken.

## Weitere Klassifikationen für Testarten

**Aus den Qualitätsmerkmalen gem. ISO/IEC 9126** (die für die meisten Testanforderungen den Rahmen bilden können) leitet sich eine große Anzahl von Testarten ab. Auf Grund ihrer Vielfalt werden nachfolgend nur wenige Beispiele genannt: *Sicherheitstest, Funktionstest, Wiederanlauftest, GUI-Test, Fehlertest, Installationstest, Lasttest.*

**Zum Testen ausgewählte methodischen Ansätze** spiegeln sich ebenfalls in Testart-Bezeichnungen wieder. Das sind z. B.:

Spezielle Teststrategien: *SMART-Testing, Risk based testing, Data driven Testing, exploratives Testen, top-down / bottom-up, hardest first, big-bang.*

Besondere Methoden sind für *Entscheidungstabellentests, Use-Case- oder anwendungsfallbasierte Tests, Zustandsübergangs- / zustandsbezogene Tests, Äquivalenzklassentests und Pair-wise-Tests* die Grundlage für Testartbezeichnungen.

Testart-Bezeichnungen leiten sich u. a. auch vom **Zeitpunkt der Testdurchführung** ab:

Die bedeutendsten und meist auch im allgemeinen Sprachgebrauch benutzten Testartbezeichnungen sind die Teststufen, die mit *Modultest, Integrationstest, Systemtest, Abnahmetest* bezeichnet werden.

Als Testarten für frühe Tests sind auch bekannt: *Alpha-Test* (erste Entwicklertests), *Beta-Test* (oder Feldtest; spätere Benutzer testen eine Pre-Version der Software)

*Produktionstests* werden auf den Systemen der Produktionsumgebung, evtl. sogar erst im produktiven Betrieb der Software (nur für unkritische Funktionen geeignet) durchgeführt. Möglicher Grund: Nur die Produktionsumgebung verfügt über bestimmte, zum Testen erforderliche Komponenten.

Auch Test-Wiederholungen gehören zum Aspekt 'Testzeitpunkt': Diese Tests werden *Regressionstest oder Retest* etc. genannt.

Indirekt mit Zeitbezug sind zu nennen: *Entwicklertest (vor Anwendertest ...), statisches Testen (vor dynamischem Testen).*

Auch **nach der Testintensität** werden einige Testarten speziell benannt:

Unterschiedliche Grade der Testabdeckung (Test-Coverage- bzw. Code-Coverage) werden mit *Überdeckungstests* erreicht, die (auf Grund der geringen Größe der Testobjekte) besonders für *Komponententests* geeignet sind. Testartenbezeichnungen hierzu: Anweisungs- (C0-Test, C1-Test), Zweig-, Bedingungs- und Pfadüberdeckungstest.

Bewusst nach dem Zufallsprinzip werden *Smoketests* ausgeführt, frühe, oberflächliche Tests, bei denen lediglich ausprobiert werden soll, ob das Testobjekt 'überhaupt irgend etwas tut - ohne abzurauchen'. Beim *Error Guessing* provozieren (erfahrene) Tester bewusst Fehler.

Mit *Vorbereitungstests* werden vorerst nur wichtige Hauptfunktionen getestet.

Testarten werden auch danach bezeichnet, welcher **Informationsstand über die zu testenden Komponenten** vorhanden ist (auf dessen Basis Testfälle spezifiziert werden könnten):

*Black-Box-Tests* werden ohne Kenntnisse über den inneren Aufbau des zu testenden Systems, sondern auf der Basis von Entwicklungsdokumenten entwickelt. In der Praxis werden Black-Box-Tests meist nicht von den Software-Entwicklern, sondern von fachlich orientierten Testern oder von speziellen Test-Abteilungen oder Test-Teams entwickelt. In diese Kategorie fallen auch Anforderungstests (Requirements Tests; auf der Grundlage spezieller Anforderungen); stochastisches Testen (statistische Informationen als Testgrundlage)

*White-Box-Tests*, auch strukturorientierte Tests genannt, werden auf Grund von Wissen über den inneren Aufbau der zu testenden Komponente entwickelt. Entwicklertests sind i. d. R. White-Box-Tests - wenn Entwickler und Tester dieselbe Person sind. Hierbei besteht die Gefahr, dass Missverständnisse beim Entwickler auch zu 'falschen' Testfällen führen, der Fehler also nicht erkannt wird.

*Grey-Box-Tests* werden von den gleichen Entwicklern entwickelt wie das zu testende System selbst, gemäß den Regeln der testgetriebenen Entwicklung, also vor der Implementierung des Systems, und damit (noch) ohne Kenntnisse über das zu testende System. Auch hier gilt: Missverständnisse zur Aufgabenstellung können zu falschen Testfällen führen.

**Aus der Art und dem Umfang des Testobjekts** ergeben sich die folgenden Testart-Bezeichnungen:

*Systemtest, Geschäftsprozesstest* (Integration von Programmteilen im Geschäftsprozess), *Schnittstellentest* (zwischen 2 Programmen), *Modultest*, für einzelne Codeteile *Debugging* (Testen des Programmcodes unter schrittweiser oder abschnittsweiser Kontrolle und ggf. Modifikation des Entwicklers) und *Mutationstest*.

*Batchtest* werden Tests von Stapelprogrammen, *Dialogtest* Tests für Dialogprogramme genannt.

Internet- oder Intranet-Funktionen werden mit *Web-Tests* (auch Browsertest genannt) durchgeführt.

*Hardwaretests* sind Tests, die auf konkrete, Hardwarekomponenten betreffende Last- und andere Kriterien ausgerichtet sind - wie Netzlast, Zugriffszeit, Parallelspeichertechniken etc. Auch *Produktionstests* gehören hierzu.

Testarten können auch danach benannt sein, **wer die Tests ausführt oder spezifiziert**:

*Entwicklertests* (Programmierertests), Benutzertests, Anwendertests, Benutzerakzeptanztests (User Accaptence Tests - UAT) *werden von der jeweiligen Testergruppe durchgeführt.*

*Abnahmetests* führen die fachlich für die Software verantwortlichen Stellen aus.

*Betriebstests, Installationstests, Wiederanlauftests, Notfalltests* nehmen u. a. auch Vertreter des Rechenzentrums vor - zur Sicherstellung der Einsatzfähigkeit nach definierten Vorgaben.

Von eher untergeordneter Bedeutung sind Testbegriffe, die sich an der **Art der Softwaremaßnahme** orientieren, aus der der Testbedarf resultiert:

In Wartungsprojekten werden *Wartungstests und Regressionstests* ausgeführt; dabei werden i. d. R. bereits vorhandenen Testfälle und Testdaten benutzt.

In Migrationsprojekten werden *Migrationstests* durchgeführt; die Datenüberführung und spezielle Migrationsfunktionen sind hierbei z. B. Testinhalte.

# Weitere Teilaspekte beim Testen

## Teststrategie

Pol, Koomen und Spillner beschreiben in [1] die Teststrategie als **umfassenden Ansatz:** *Eine Teststrategie ist notwendig, da ein vollständiger Test, d. h. ein Test, der alle Teile des Systems mit allen möglichen Eingabewerten unter allen Vorbedingungen überprüft, in der Praxis nicht durchführbar ist. Deswegen muss in der Test-Planung anhand einer Risikoabschätzung festgelegt werden, wie kritisch das Auftreten eines Fehlers in einem Systemteil einzuschätzen ist (z.B. nur finanzieller Verlust oder Gefahr für Menschenleben) und wie intensiv (unter Berücksichtigung der verfügbaren Ressourcen und des Budgets) ein Systemteil getestet werden muss oder kann.*

Demnach ist in der Teststrategie festzulegen, welche Teile des Systems mit welcher Intensität unter Anwendung welcher Testmethoden und -Techniken unter Nutzung welcher Test-Infrastruktur und in welcher Reihenfolge (siehe auch Teststufen) zu testen sind.

Sie wird vom Testmanagement im Rahmen der Testplanung erarbeitet, im Testplan dokumentiert und festgelegt und als Handlungsrahmen für das Testen (durch die Testteams) zu Grunde gelegt.

Nach einer anderen Interpretation wird "Teststrategie" als **methodischer Ansatz** verstanden, nach dem das Testen angelegt wird.

So benennt z. B. ISTQB Ausprägungen für Teststrategien wie folgt:

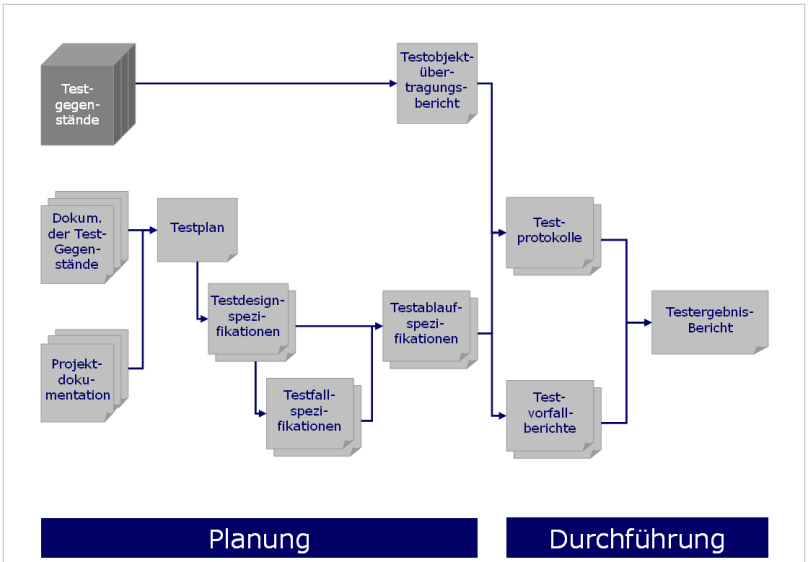
- top-down: Haupt- vor Detailfunktionen testen; untergeordnete Routinen werden beim Test zunächst ignoriert oder (mittels "Stubs") simuliert
- bottom-up: Detailfunktionen zuerst testen; übergeordnete Funktionen oder Aufrufe werden mittels "Testdriver" simuliert
- hardest first: Situationsbedingt das Wichtigste zuerst
- big-bang: Alles auf einmal

Weitere Prinzipien und Techniken für Teststrategien sind:

- Risk based testing: Testprinzip, nach dem die Testabdeckung an den Risiken ausgerichtet wird, die in den Testobjekten (für den Fall des Nichtfindens von Fehlern) eintreten können.
- Data driven Testing: Testtechnik, mit der über Einstellungen in den Testscripts die Datenkonstellationen gezielt geändert werden können, um damit mehrere Testfälle hintereinander effizient testen zu können
- Testgetriebene Entwicklung
- SMART: Testprinzip "Specific, Measurable, Achievable, Realistic, time-bound"
- Keyword driven testing
- framework based: Test-Automatisierung mittels Testwerkzeugen für bestimmte Entwicklungsumgebungen / Programmiersprachen

## Dokumentation

Zur Testplanung gehört auch die Vorbereitung der Dokumentation. Eine normierte Vorgehensweise dazu empfiehlt der Standard IEEE 829.[4] [5] . Laut diesem Standard gehören zu einer vollständigen Testdokumentation folgende Unterlagen:




Zusammenhang der Dokumente und Verfahrensschritte laut IEEE 829

Testplan

> Beschreibt Umfang, Vorgehensweise, Terminplan, Testgegenstände.

Testdesignspezifikation

> Beschreibt die im Testplan genannten Vorgehensweisen im Detail.

Testfallspezifikationen

> Beschreibt die Umgebungsbedingungen, Eingaben und Ausgaben eines jeden Testfalls.

Testablaufspezifikationen

> Beschreibt in Einzelschritten, wie jeder Testfall durchzuführen ist.

Testobjektübertragungsbericht

> Protokolliert, wann welche Testgegenstände an welche Tester übergeben wurden.

Testprotokoll

Listet chronologisch alle relevanten Vorgänge bei der Testdurchführung.

Testvorfallbericht

Listet alle Ereignisse, die eine weitere Untersuchung erforderlich machen.

Testergebnisbericht

Beschreibt und bewertet die Ergebnisse aller Tests.

### Testautomatisierung

Insbesondere bei Tests, die häufig wiederholt werden, ist deren Automatisierung angeraten. Dies ist vor allem bei Regressionstests und bei testgetriebener Entwicklung der Fall. Darüber hinaus kommt Testautomatisierung bei manuell nicht oder nur schwer durchführbaren Tests zum Einsatz (z.B. Lasttests).

- Durch Regressionstests wird nach Softwareänderungen meist im Zuge des System- oder Abnahmetests der fehlerfreie Erhalt der bisherigen Funktionalität überprüft.
- Bei der testgetriebenen Entwicklung werden die Tests im Zuge der Softwareentwicklung im Idealfall vor jeder Änderung ergänzt und nach jeder Änderung ausgeführt.

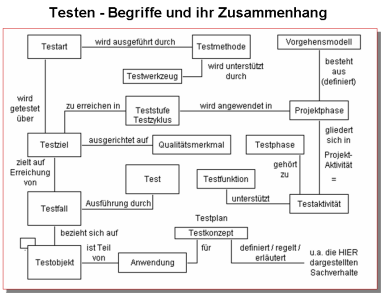
Bei nicht automatisierten Tests ist in beiden Fällen der Aufwand so groß, dass häufig auf die Tests verzichtet wird.

## Übersichten / Zusammenhänge

### Begriffe beim Testen

Die nebenstehende Grafik zeigt Begriffe, die im Kontext 'Testen' auftreten - und wie sie mit anderen Begriffen in Verbindung stehen.

Zur Erläuterung: Die Grafik ist optisch ein ER-Diagramm, sie zeigt aber keine Datenobjekte, sondern nur *Begriffe*. Der den Zusammenhang beschreibende Text ist aus Sicht des 'ersten' Begriffs formuliert und steht links von der Linie, die (aus seiner Sicht) zum zweiten Begriff führt.




Begriffe und ihr Zusammenhang

### Schnittstellen beim Testen

Die Grafik zeigt die wichtigsten Schnittstellen, die beim Testen auftreten. Zu den von Thaller[6] genannten 'Partnern' beim Testen wird nachfolgend beispielhaft angeführt, WAS jeweils kommuniziert / ausgetauscht wird.


Wichtige Schnittstellen beim Testen

- Projektmanagement: Termin- und Aufwandsrahmen, Status je Testobjekt ('testready'), Dokumentationssysteme
- Linienmanagement (und Linienabteilung): Fachlicher Support, Testabnahme, fachliche Tester stellen
- Rechenzentrum: Testumgebung(en) und Testwerkzeuge bereitstellen und betreiben
- Datenbankadministrator: Testdatenbestände installieren, laden und verwalten
- Konfigurations Management: Testumgebung einrichten, Integrieren der neuen Software

- Entwicklung: Test-Basisdokumente, Prüflinge, Support zum Testen, Testergebnisse erörtern
- Problem- und CR-Management: Fehlermeldungen, Rückmeldung zum Retest, Fehlerstatistik
- Lenkungsausschuss: Entscheidungen zur Test(stufen)abnahme oder zum Testabbruch

## Einzelnachweise

[1] Pol, Koomen und Spillner: *Management und Optimierung des Testprozesses.* dpunkt.Verlag, ISBN 978-3-89864-951-3
[2] Ernst Denert: *Software-Engineering.* Springer, Berlin 1991, 1992. ISBN 3-540-53404-0
[3] Peter Liggesmeyer: *Software-Qualität - Testen, Analysieren und Verifizieren von Software.* Spektrum Akademischer Verlag, Heidelberg/Berlin 2002, S.34. ISBN 3-8274-1118-1
[4] IEEE Standard for Software Test Documentation (IEEE Std. 829-1998)
[5] IEEE Standard for Software and System Test Documentation (IEEE Std. 829-2008)
[6] Georg-Edwin Thaller: *Softwaretest*

## Literatur

- Andreas Spillner, Tilo Linz: *Basiswissen Softwaretest. Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard* dpunkt.verlag, Heidelberg 2010, 4. Aufl., ISBN 978-3-89864-642-0
- Harry Sneed, Manfred Baumgartner, Richard Seidl: *Der Systemtest - Anforderungsbasiertes Testen von Software-Systemen.* Carl Hanser, München/Wien 2006. ISBN 3-446-40793-6
- Georg Erwin Thaller: *Software-Test. Verifikation und Validation.* Heise, Hannover 2002. ISBN 3-88229-198-2

# Quelle(n) und Bearbeiter des/der Artikel(s)

**Softwaretest** *Quelle*: http://de.wikipedia.org/w/index.php?oldid=86318834 *Bearbeiter*: Abubiju, AchimR, Aka, Akropolit, Ammeling, Ano Nym, Avron, Bananen-Joe, BesondereUmstaende, Burmagroup, Bücherwurm16, Carol.Christiansen, ChristianHujer, Clemfix, Comc, Dachris, Darok, Defrenrokorit, Dlm, EinStein, Ellyce, Elwe, Entlinkt, Erkan Yilmaz, Fizfaz, Flavia67, Fleasoft, Fristu, Fruchtcocktail, GB, GGlasauer, Gadelor, Gaius L., Gereon K., Ghw, Gnu1742, Hadhuey, Hansruedi.Tremp, HenrikHolke, Himuralibima, Jpp, JustB, Karl-Friedrich Lenz, Kipparj, Kku, Kraymer, LKD, Laetterman, Levin, Liberatus, Lostintranslation, Lustiger seth, Löschfix, Ma-Lik, Mareike jacobshagen, Marsellus, Mcaspari01, Media lib, Michael Kunz, Mikue, Mussklprozz, NSz, Nutzer Eins, Pelz, PeterFrankfurt, QFS, Ramtam, Reallimk, ReneS, Reseka, Rfc, Rufus46, S.K., STBR, Sebastian.Dietrich, Silberchen, Sinn, SonniWP2, Sparti, Staro1, Stefan Knoepfel, Stefan.qn, StefanMacke, Stephan.Weissleder, Taschenrechner, Test-tools, Theoprakt, Thomaserl, Till.niermann, Traute Meyer, Tsor, Tuthmosis VII, Tönjes, Uncopy, VerwaisterArtikel, Video2005, Vsesb01, VÖRBY, Wiki-piet, Xflupp, Zaungast, Zinnmann, 216 anonyme Bearbeitungen

# Quelle(n), Lizenz(en) und Autor(en) des Bildes

**Datei:V-Modell.svg** *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:V-Modell.svg *Lizenz*: unbekannt *Bearbeiter*: User:EinStein

**Datei:Test Phasenmodell.png** *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:Test_Phasenmodell.png *Lizenz*: unbekannt *Bearbeiter*: VÖRBY. Original uploader was VÖRBY at de.wikipedia

**Datei:Test ganzheitlich.png** *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:Test_ganzheitlich.png *Lizenz*: unbekannt *Bearbeiter*: VÖRBY

**Datei:IEEE829_Uebersicht_Deutsch.png** *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:IEEE829_Uebersicht_Deutsch.png *Lizenz*: Creative Commons Attribution-Sharealike 3.0 *Bearbeiter*: User:Mussklprozz

**Datei:Testen BegriffeZusHang.png** *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:Testen_BegriffeZusHang.png *Lizenz*: unbekannt *Bearbeiter*: VÖRBY. Original uploader was VÖRBY at de.wikipedia

**Datei:Testen SchnittstellenThaller.png** *Quelle*: http://de.wikipedia.org/w/index.php?title=Datei:Testen_SchnittstellenThaller.png *Lizenz*: unbekannt *Bearbeiter*: VÖRBY. Original uploader was VÖRBY at de.wikipedia

# Lizenz

### 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- **O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.
You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

### 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

### 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

### 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

### 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

### 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

### 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
>
> Permission is granted to copy, distribute and/or modify this document
>
> under the terms of the GNU Free Documentation License, Version 1.2
>
> or any later version published by the Free Software Foundation;
>
> with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
>
> A copy of the license is included in the section entitled
>
> "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the
>
> Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.