Beispiele für Qualitätsanforderungen an Software

Dr. Gernot Starke [gernot.starke@innoq.com](mailto:gernot.starke@innoq.com)

v0.6

# Über Qualität

Die Qualität eines Produktes oder Systems ist ganz allgemein als Menge von Eigenschaften oder Merkmalen definiert.

In der Praxis haben sich einige Kategorien (Oberbegriffe) für häufig auftretende *Qualitätsanforderungen* (synonym: Qualitätsziele) etabliert, im wesentlichen durch das DIN/ISO 9126 Begriffsmodell geprägt.

## Die Kategorien

* [Änderbarkeit](#aenderbarkeit)
* [Benutzbarkeit](#benutzbarkeit)
* [Effizienz](#effizienz)
* [Zuverlässigkeit](#zuverlaessigkeit)
* [Betreibbarkeit](#betreibbarkeit)
* [Sonstiges (u.a. Funktionalität)](#sonstige)

## Einordnung kann mehrdeutig sein

Manche Qualitätsziele oder -anforderungen gehören zu mehreren "Oberbegriffen" oder Kategorien - die Zuordnung der Beispiele zu Kategorien oder Kpiteln fiel mir deswegen manchmal schwer.

Falls Sie Ihre spezifischen Qualitätsanforderungen in der Gliederung nicht finden, so schauen Sie bitte im Anhang: [Q-Merkmale](#qmerkmale) - dort finden Sie eine umfangreiche Tabelle der Q-Merkmale mit ihren Synonymen und Oberbegriffen.

## Quellen

Die Ideen zu den hier aufgeführten Szenarien und Qualitätsanforderungen stammen aus unterschiedlichen Quellen und Projekten. Ich habe alle überarbeitet und weitgehend neutralisiert. Die Erklärungen von Qualitätsmerkmalen stammen teilweise aus DIN/ISO 9126 bzw. dem FURPS-Modell+.

[bass09] Len Bass et.al.: Software Architecture in Practice. Addision-Wesley, 2. Auflage 2009.

[clements01] Paul Clements et.al.: Evaluating Software Architectures. Addision-Wesley, 2001.

[robertson12] S. und J. Robertson: Mastering the Requirements Process - Getting Requirements Right. Addision-Wesley, 3. Auflage 2012.

[rupp09] Chris Rupp et.al: Requirements Engineering und -management. Hanser-Verlag, 5. Auflage 2009.

[eeles05] Peter Eeles: [Capturing Architectural Requirements](http://www.ibm.com/developerworks/rational/library/4706.html). Erläutert das FURPS+ Modell zur Erfassung von Qualitätsanforderungen an Software.

## Mitwirkende

* Christoph Iserlohn (innoQ), neue Szenarien
* Oliver Tigges (innoQ), neue Szenarien
* Alex Heusingfeld (innoQ), Infrastruktur, AsciiDoc
* Roland Krummenacher & div. N.N.: Reviews, Korrekturen

Ideen und Anregungen zu einigen Szenarien auch von Firebrand-Softwarearchitectures.

## Lizenz

Copyright 2012-2013 the original author (Gernot Starke) or authors and contributors.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

# Änderbarkeit

## Was bedeutet Änderbarkeit?

Änderbarkeit: Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist.

Änderungen sind dabei Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen & der funktionalen Spezifikationen.

Zur Änderbarkeit gehören gemäß DIN/ISO 9126 folgende Teilmerkmale:

**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:**

Aufwand, der zur Prüfung der geänderten Software notwendig ist

Umgangssprachlich heißt Änderbarkeit oftmals Flexibilität, Anpassbarkeit, Wartbarkeit oder Erweiterbarkeit.

## Szenarien für Änderbarkeit

### Szenario: Neuer Konnektor

Wenn die Datenimport-Komponente Daten aus einer zusätzlichen Datenquelle einlesen soll, so muss der dafür notwendige Konnektor innerhalb von drei Personenmonaten entwickelt, integriert und in Betrieb genommen werden können.

|  |  |
| --- | --- |
| Geschäftsziele | Flexibilität, Reaktionsfähigkeit, Entgegenkommen |
| Reaktion | Der neue Konnektor ist programmiert und in Betrieb. |
| Zielwert | 3 Personenmonate |

### Szenario: Jährliche Aktualisierung

Das Datenformat des jährlichen Buchungsreports wird jedes Jahr am 31. Dezember aktualisiert, um den gesetzlichen und steuerlichen Änderungen zu genügen. Das System muss das neue Format erzeugen können.

|  |  |
| --- | --- |
| Geschäftsziele | Betriebsbereitschaft |
| Reaktion | Die Buchungs- und Reportingkomponente ist geändert und liefert für den Buchungsreport das neue Format. |
| Zielwert | Die Änderung muss in weniger als 80 Personenstunden umgesetzt werden können. |

### Szenario: Plugins entwickeln

Ein unabhängiger Entwickler kann PlugIns für das System erstellen.

|  |  |
| --- | --- |
| Geschäftsziele | Ermögliche der "Community", die Basisfunktionalität des Systems zu erweitern |
| Auslöser | Unabhängiger Entwickler (eine "third party") möchte eigene Erweiterung ("PlugIn") für das System erstellen |
| Reaktion | Unabhängige Entwickler sind in der Lage, auf Basis vordefinierter Erweiterungspunkte das System zu erweitern. Dafür sind nebem dem System ausschliesslich Open Source Komponenten notwendig. |
| Zielwert | Einfache PlugIns sollen innerhalb von 8 Personenstunden umsetzbar sein. |

### Szenario: Wechsel des Algorithmus

Wechsel des Algorithmus zur Routenberechnung.

|  |  |
| --- | --- |
| Geschäftsziele | Setze Änderungen oder Verbesserungen in der Routenoptimierung möglichst schnell und einfach um |
| Auslöser | Entwickler möchte einen neuen / anderen Algorithmus zur Berechnung passender Routen entwickeln und in das Gesamtsystem integrieren. |
| Reaktion | Es gibt keinerlei Seiteneffekte auf andere Bausteine des Systems. |
| Zielwert | Es lediglich ein einziger Softwarebaustein (Modul, Komponente, Package) betroffen, der eigentliche Austausch des alten gegen den neuen Algorithmus geschieht innerhalb von 4h (inklusive eventuelle Anpassungen der Build-Umgebung). |

### Szenario: Neues Feld in Stammdaten

Ein Entwickler möchte ein neues Feld in die XY-Stammdaten aufnehmen.

|  |  |
| --- | --- |
| Geschäftsziele | Fähigkeit zur Änderung / Anpassung des Domänenmodells (fachliche Datenstruktur) inklusive Oberfläche und Datenbank. |
| Auslöser | Die Fach-, Business- oder Marketingabteilung benötigt in den Stammdaten ein neues Attribut vom Typ "String" (maximale Länge 80 Zeichen) |
| Reaktion | Ein Entwickler identifziert die zugehörigen Programmteile, nimmt die notwendigen Erweiterungen vor, testet die Erweiterungen in Integrations- und Systemtestumgebung. |
| Zielwert | Alle notwendigen Änderungen und zugehörigen Tests sind innerhalb von 40h (Aufwand) abgeschlossen. |

### Szenario: Unterstütze ANSI SQL-92 kompatibe Datenbanksysteme.

Biete das System Kunden mit verschiedenen DB-Systeme an.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Flexibilität, biete das System Kunden mit unterschiedlichen Datenbanksystemen an. |
| Auslöser | Ein Kunde möchte das System mit einer (für uns neuen) (ANSI-SQL-92 kompatiblen) Datenbank verwenden. |
| Reaktion | Entwickler und Tester nehmen das System mit der Datenbank in Betrieb und führen sämtliche Systemtests durch, insbesondere die SQL-92 Kompatibilitätstests. |
| Zielwert | Sofern die Datenbank die Kompatibilitätstests erfolgreich durchläuft, sind auch alle übrigen Systemtests erfolgreich. |

### Szenario: Verständlichkeit von generiertem Code

Generierter Code für Testfälle ist leicht verständlich und manuell modifizierbar.

|  |  |
| --- | --- |
| Qualitätsziele | Verständlichkeit, Testbarkeit, Modifizierbarkeit |
| Geschäftsziel(e) | Tester können den automatisch generierten Code für Testfälle leicht verstehen und eigenen Bedürfnissen anpassen. |
| Auslöser | Ein Tester beschreibt einen Testfall in einer (semi-formalen) Sprache. |
| Reaktion | Testcode (in C, Java oder C#) für bestimmte Teile des Systems wird durch den XY-Generator automatisch auf Basis dieser Testfallbeschreibung erzeugt. |
| Zielwert | Ein Tester kann den generierten (C- oder Java) Quellcode für einen Testfall im Mittel in weniger als 30 Minuten verstehen und einfache Änderungen daran vornehmen. |
| Anmerkung | Im hier genannten Szenario ändern Tester generierten Code. Je nach Art des Codegenerators könnte dieser geänderte Code bei Neugenerierung überschrieben werden. Entsprechende Anforderungen müssten in anderen Szenarien gestellt werden. |

### Szenario: Schnelle Lokalisierung von Fehlern

Entwickler kann die Ursache fehlgeschlagener Testfälle in kurzer Zeit lokalisieren.

|  |  |
| --- | --- |
| Geschäftsziele | Kurze Fehlerbehebungszeiten, gute Änderbarkeit, geringes Risiko bei Änderungen |
| Auslöser | Entwickler, Tester oder das Build-System führen automatische Testfälle (Unit- oder Integrationstests) aus. |
| Reaktion | Ein Testfall schlägt fehl. Aus den Fehler- und/oder Log-Nachrichten kann ein Entwickler die Ursache/den Auslöser des Fehlers in kurzer Zeit lokalisieren. |
| Zielwert | Entwickler findet die Ursache durchschnittlich in weniger als 10 Minuten. |

### Szenario: Neue Funktion

* Der Auftraggeber fordert eine neue Funktion im System.
* Entwickler identifizieren kurzer Zeit die für diese Erweiterung passende Stelle innerhalb des Quellcodes.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Eine neue Funktion soll in kurzer Zeit implementiert werden können. |
| Auslöser | Änderungsanforderung |
| Reaktion | Entwickler suchen innerhalb des Quellcodes die geeignete Stelle für die Erweiterung. |
| Zielwert | Entwickler finden die geeignete Stelle in weniger als zwei Stunden. |

### Szenario: Kurze Implementierungszeit für neue Funktion

Eine neue Funktion mittlerer Komplexität soll in kurzer Zeit implementiert werden können.

|  |  |
| --- | --- |
| Geschäftsziele | Eine neue Funktion mittlerer Komplexität soll in kurzer Zeit implementiert werden können. |
| Auslöser | Änderungsanforderung |

Reaktion und Zielwert: Entwickler implementieren diese Funktion inklusive automatisierter Testfälle (Unit-Tests) innerhalb von weniger als 5 PT.

### Szenario: Erweiterte Logmeldungen

Das bisherige Format der Logmeldungen genügt für den Betreiber nicht mehr. Sämtliche Logmeldungen müssen um zusätzliche Informationen ergänzt werden.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Flexible Anpassung an geänderte/erweiterte Anforderungen beim Logging. |
| Auslöser | Betreiber benötigt zusätzliche Informationen in Logmeldungen (beispielsweise IP-Adresse des ausführenden Webservers, Session-ID oÄ). |
| Reaktion | Entwickler müssen die betroffenen Stellen im Quellcode innerhalb von als 40h anpassen können. Anmerkung: Für dieses Szenarion ist das Verhältnis aus Zielwert und Umfang des betroffenen Quellcodes (LoC) relevant: |

### Szenario: Report in kurzer Zeit implementieren

Ein Entwickler möchte einen Report über alle Buchungen eines Tages implementieren

|  |  |
| --- | --- |
| Geschäftsziel(e) | Eine neue Funktion (geringer Komplexität) soll in kurzer Zeit umsetz- und testbar sein |
| Auslöser | Auftraggeber benötigt die Ergebnisse des Buchungsreports |
| Reaktion | * Entwickler implementiert diese Funktion innerhalb von 3 Arbeitstagen. * Tester testet diese Funktion gegen die Spezifikation innerhalb von 2 Arbeitstagen. |

### Szenario: Schnittstelle um Authentifizierung erweitern

Entwickler erweitert die externe XY-Schnittstelle um Authentifizierung.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Die XY-Schnittstelle benötigt ab sofort eine sichere Authentifizerung. |
| Auslöser | Regularien oder Kundenanforderungen erfordern eine sichere Authentifizierung über OAuth 2.0 der XY-Schnittstelle |
| Reaktion und Zielwert | Entwickler erweitern die Schnittstelle um OAuth 2.0 innerhalb von 5 PT Entwicklungszeit. |

### Szenario: Kommerzielle durch Open-Source Datenbank ersetzen

Das verwendete Datenbanksystem muss von einem kommerziellen durch ein Open-Source System ersetzt werden können.

|  |  |
| --- | --- |
| Geschäftsziele | Bei Bedarf und in bestimmten Einsatzszenarien des Systems müssen Lizenzkosten reduziert werden. |
| Auslöser | Betreiber/Kunde des Systems möchte die LIzenzkosten des verwendeten Datenbanksystems reduzieren. |
| Reaktion | Entwickler können die standardmässig verwendete Oracle (™) Datenbank durch eine quelloffene (etwa: MySQL oder PostgreS ersetzen. |
| Zielwert | * Der Wechsel der Datenbank ist mit weniger als 40 PT Aufwand durchführbar. * Alle funktionalen Anforderungen werden danach erfüllt, nachgewiesen durch Integrations-, System- und Abnahmetests. * Die Laufzeiten der wichtigsten 15 Anwendungsfälle verschlechtert sich gegenüber der kommerziellen Datenbank um höchstens 15%. * Die Laufzeiten aller übrigen Anwendungsfälle verschlechtert sich gegenüber der kommerziellen Dankbank um höchstens 25%. |

### Szenario: Geschäftsprozess erweitern

Der XY-Geschäftsprozess kann zur Laufzeit um zusätzlicher Verarbeitungsschritte ergänzt werden

|  |  |
| --- | --- |
| Geschäftsziele | Dynamische Erweiterung des Geschäftsprozesses XY trägt zur Zufriedenheit der Anwender bei. |
| Auslöser | Zur Anpassung an Marktbedürfnisse erweitert ein Entwickler oder Architekt den Geschäftsprozess XY um einen zusätzlichen Schritt, während die aktiven Prozessinstanzen von XY vom System bearbeitet werden. |
| Kontext | Mehr als 20 Benutzer haben unvollständige Projekte (Daten) auf Basis des aktuellen XY-Prozesses gespeichert. |

Reaktion / Zielwert: Das System aktualisiert selbständig und ohne Datenverlust die vorhandenen Daten der unvollständigen Projekte (automatische Migration der Benutzerspezifischen Daten auf die neue Version von XY).

Die Änderung des XY-Prozesses dauert nicht länger als 80 Personenstunden (Aufwand).

### Szenario:

Der einzelne Verarbeitungsschritt AB innerhalb des Anwendungsfalls XY wird von der Regulierungsbehörde für ungültig erklärt und im System entfernt. Die vom System bearbeiteten Daten sind nicht betroffen.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Die Änderung am Anwendungsfall XY kann mit geringen Kosten und ohne negative Auswirkungen durchgeführt werden. |
| Auslöser | Der Gesetzgeber, vertreten durch die Regulierungsbehörde, untersagt die Verwendung des Verarbeitungsschrittes AB. |
| Reaktion | Ein Entwickler oder Architekt entfernt im System den Verarbeitungsschritt AB (durch löschen der entsprechenden Aufrufe oder durch Neukonfiguration der Prozessabläufe). |
| Zielwert | Die Änderung erfordert höchstens 24 Zeitstunden mit höchstens 48 Personenstunden Aufwand. Nach dieser Zeit ist das System wieder völlig funktionsfähig. |

Diese Änderung hat keine Auswirkung auf die im System vorhandenen Daten der Anwender/Kunden bezüglich des XY-Anwendungsfalles. Eine (automatische) Migration einiger Daten ist zulässig, darf allerdings die 24 Zeitstunden-Grenze nicht überschreiten.

### Szenario: Erweiterung um automatisierte Testsuite

Erweitere das X-Subsystem um eine vollständig automatisierte Testsuite.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Verbessere die Änderbarkeit und Testbarkeit des Systems. |
| Auslöser | Kunde kündigt umfangreiche Änderungswünsche am X-Subsystem an. Die bessere Änderbarkeit und Testbarkeit reduzieren das Risiko dieser Änderungen. |
| Reaktion | Entwickler implementieren Unit- und Integrationstests für sämtliche Klassen sowie Schnittstellen des Subsystems X. |
| Zielwert | Gesamtaufwand der Änderungen liegt unter 200 Personentagen. |

Szenario: Neuer Typ von Client Erweitere das XY-System um einen mobilen Client (Android, iOS), ohne die Performance der übrigen GUI-Teile zu beeinträchtigen.

|  |  |
| --- | --- |
| Geschäftsziele | Einführung neuer Zugangswege verbreitert die mögliche Kundenbasis und steigert die Attraktivität des Systems. |
| Auslöser | Kunden und Verbraucher erwarten (wie selbstverständlich) native mobile Clients als Bestandteil des Produktportfolios. |
| Reaktion | Entwicklungsteam entwirft und implementiert mobile Clients für das XY-System sowie die dazu benötigte (Daten-)Schnittstelle. |
| Zielwert | - Die bisherigen (Browser- und Rich-)Clients werden in ihrer Performance in keiner Weise beeinträchtigt. - Bei 100 parallelen Browser-Benutzern und 100 gleichzeitig angemeldeten mobilen Clients dürfen maximal 3% der Datenzugriffe maximal 20% mehr Zeit beanspruchen als vor der Einführung der mobilen Clients. |
| Anmerkung | Dieses Szenario beschreibt mehrere Qualitätsmerkmale: Attraktivität, Effizienz/Performance, Verfügbarkeit, Robustheit und Änderbarkeit. |

### Szenario: Unabhängige Erweiterung eines Subsystems

Erweiterungen oder Änderungen eines Subsystems sollen unabhängig von allen anderen Subsystemen möglich sein.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Leichte Änderbarkeit, schnelle Reaktion auf Fehler. |
| Auslöser | Entwicklungsteam ändern innerhalb eines Subsystems - die Aussenschnittstellen dieses Subsystems bleiben dabei identisch! |
| Zielwert | Kein anderes Subsystem muss geändert werden. Für sämtliche anderen Subsysteme gilt:   * Der Quellcode bleibt identisch * Compile-, Build- und Testprozesse bleiben identisch * Deployment-, Installation und Konfiguration bleiben identisch |
| Anmerkung | Diese Anforderung bedeutet, dass sämtliche Subsysteme nur über ihre öffentlichen ("offiziellen") Schnittstellen kommunizieren - und kein Subsystem Interna eines anderen verwendet oder ausnutzt. Herausfordernd insbesondere bei Kopplung über Daten oder Datenstrukturen. |

# Benutzbarkeit

## Was bedeutet Benutzbarkeit?

Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzer-gruppe. Hierunter fällt auch der Bereich Softwareergonomie.

Zu Benutzbarkeit gehören nach DIN/ISO 9126 folgende Teilmerkmale:

|  |  |
| --- | --- |
| **Verständlichkeit** | Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen. |
| **Erlernbarkeit** | Aufwand für den Benutzer, die Anwendung zu erlernen (z.B. Bedienung, Ein-, Ausgabe) |
| **Bedienbarkeit** | Aufwand für den Benutzer, die Anwendung zu bedienen. |

## Szenarien für Benutzbarkeit

### Szenario: Einfache Benutzbarkeit von Testwerkzeug

Das interaktive Testwerkzeug muss einfach benutzbar sein. Tester (für den Benutzer-Akzeptanztest) müssen innerhalb von zwei Stunden die Bedienung erlernen können.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Produktive und schnelle Akzeptanztests, schnelle Durchläufe |
| Auslöser | Ein BA-Tester testet mit dem Testwerkzeug ein neues Release der Software. |
| Reaktion | Der Tester kann das Testwerkzeug vollständig bedienen. |
| Zielwert | Benötigt dafür weniger als zwei Stunden Einarbeitung. |

### Szenario: Konsistente Tastaturkürzel

Endanwender / Endbenutzer können identische Tastaturkürzel ("keyboard shortcuts") in allen, unabhängig voneinander entwickelten, Modulen des Systems verwenden.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Konsistente Benutzerführung ("User experience") im gesamten Produkt |
| Auslöser | Benutzer möchte Tastatur zur Navigation innerhalb des Systems verwenden |
| Reaktion | Benutzer kann die gleichen Funktionen wie bei der Benutzung der Maus ausführen. |
| Zielwert | Die Tastaturkürzel sind einheitlich vergeben. Bei Verwendung der Tastatur können sämtliche Funktionen schneller oder mindestens genauso schnell wie bei Nutzung der Maus verwendet werden. |

### Szenario: Schnelle Erfassbarkeit von Informationen

Benutzer soll auch Informationen, die nicht im Zusammenhang mit der aktuellen Bildschirmmaske stehen, schnell erfassen können.

|  |  |
| --- | --- |
| Qualitätsziele | Benutzbarkeit, Effizienz |
| Geschäftsziel(e) | Einfache Benutzbarkeit, einfache Navigierbarkeit innerhalb des Systems |
| Auslöser | Der Benutzer arbeitet mit dem System. Während der Bearbeitung einer Bildschirmmaske möchte er zu einem gänzlich anderen Thema Informationen erfassen oder bearbeiten. |
| Reaktion | Der Benutzer kann einfach zu dem gewünschten Thema navigieren und nach dessen Bearbeitung einfach zum aktuellen Thema zurückkehren. |
| Zielwert | Navigation zum gewünschten Thema erfolgt in weniger als 10 Sekunden, die Rückkehr zum aktuellen Thema erfolgt mit nur einem Knopfdruck / Mausklick. |

### Szenario: Hinweis auf Fehleingaben

Benutzer werden grundsätzlich auf inkonsistente oder fehlerhafte Eingaben hingewiesen.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Konsistenz bei Benutzereingaben |
| Auslöser | Benutzer gibt Daten ein. |
| Reaktion | Das System nimmt korrekte Eingabedaten an, weist inkonsistente oder fehlerhafte Eingabedaten zurück. |
| Zielwert | Im Falle inkonsistenter oder fehlerhafter Eingabedaten gibt das System eine passende Meldung, die den Fehler oder die Inkonsistenz eindeutig und einfach aufzeigt. |

### Szenario: Benutzerfreundliches Verhalten bei Backend-Prozess

Falls ein Benutzer die pdf-Generierung des XY-Reports unterbricht, hält das System diese Generierung an und übergibt die Kontrolle innerhalb von 15 Sekunden wieder an die Benutzeroberfläche.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Verbessere die Benutzbarkeit der pdf-Generierung |
| Auslöser | Benutzer möchte die pdf-Generierung des XY-Reports unterbrechen (etwa aufgrund vorheriger Fehleingaben oder sonstiger Gründe) und klickt den "Abbrechen"-Button |
| Reaktion | Das System unterbricht die Generierung, speichert den bisherigen Generierungszustand (für eventuelle Fortsetzungen) und übergibt die Kontrolle an die Benutzeroberfläche. |
| Zielwert | Benutzer erhält Kontrolle über das UI innerhalb von 15 Sekunden (d.h. In spätestens 15 Sekunden haben alle beteiligten Generierungsprozesse den Abbrechen-Befehl erfolgreich quittiert). |
| Bemerkung | Die Generierung dieses Reports läuft in mehreren parallelen Threads (oder Prozessen), eventuell sogar auf mehreren unterschiedlichen Prozessoren oder (virtuellen) Maschinen. Die Unterbrechung muss synchron und konsistent über alle diese Ausführungsinstanzen erfolgen. |

### Szenario: Aussagekräftige Fehlermeldungen ohne Absturz

Falls eine Fehlersituation auftritt, wird dies dem Benutzer in aussagekräftigen Meldungen angezeigt. Das System stürzt bei Ausnahmesituationen (Speicherüberlauf, Hardwarefehler) nicht ab, sondern fährt höchstens kontrolliert heruntre.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Verbessere die Benutzbarkeit (und gefühlte Zuverlässigkeit). Ermögliche Benutzern, zu Fehlern führende Kombinationen von Eingabedaten zu korrigieren, ohne dass das System abstürzt. |
| Auslöser | Ein Fehler / Ausnahmesituation in der Infrastruktur tritt auf (Speicherüberlauf, Out-of-Memory, Hardwarefehler). |
| Reaktion | Das System erkennt den Fehler, meldet (soweit möglich) an den Benutzer und fährt kontrolliert herunter. |
| Zielwert | Fehlererkennung erfolgt innrehalb von 15 Sekunden, Meldung an Benutzer (sofern noch möglich) innerhalb von 1 Sekunde, herunterfahren innerhalb von 15 Sekunden. |

### Szenario: Status lang laufender Prozesse erkennbar

Das System zeigt den Fortschritt der lange laufenden XY-Konvertierungsprozesse in der grafischen Oberfläche dem Benutzer an.

|  |  |
| --- | --- |
| Anmerkung | Diese XY-Konvertierung dauert 1-18 Stunden. |
| Geschäftsziel(e) | Fortschrittsüberwachung, Benutzerfreundlichkeit |
| Auslöser | Benutzer möchte über Fortschritt der lang laufenden Berechnungen/Prozesse informiert werden. |
| Reaktion | Das System zeigt die Anzahl der bisher verarbeiteten Datensätze, das verarbeitete Datenvolumen in Megabyte sowie den geschätzten verbleibenden Restaufwand an der GUI an. |
| Zielwert | Die Aktualisierung dieser Informationen erfolgt mindestens alle 60 Sekunden - höchstens alle 5 Sekunden. |

### Szenario: Einhaltung der Windows-8 User Experience Guidelines

Das System soll den Auflagen und Vorschlägen der Microsoft Windows User Experience Guidelines für Windows-8 entsprechen.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Einheitliches, dem Windows-8 Look-and-Feel entsprechendes Aussehen und Verhalten. |
| Auslöser | Die Benutzeroberfläche und interaktiven Komponenten des Systems sollen neu gestaltet und implementiert werden. |
| Zielwert | Ein fachkundiger Auditor testiert die Übereinstimmung mit o.g. Guidelines ohne Einschränkung. |

### Szenario: Benutzerinteraktionen an grafischer Oberfläche

Sämtliche Benutzerinteraktion und -meldungen im System sind als GUI implementiert.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Erhöhe die Akzeptanz und Produktivität der Arbeit mit dem System. |
| Auslöser | Alle Anforderungen an das System. |
| Zielwert | Das System erfordert keine Benutzerinteraktion außerhalb der grafischen Oberfläche. |

# Effizienz

## Was bedeutet Effizienz?

Nach DIN/ISO 9126: Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen.

In der Praxis oftmals vereinfacht als *Performance*, Verarbeitungsgeschwindigkeit, Antwortzeit, Skalierbarkeit, Durchsatz, Speicherbedarf oder Mengengerüst bezeichnet.

Nach DIN/ISO 9126 gehören zu Effizienz folgende Teilmerkmale:

* *Zeitverhalten*: Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung.
* *Verbrauchsverhalten*: Anzahl, Menge und Dauer der benötigten Betriebsmittel für die Erfüllung der Funktionen.

Anmerkung: Die Effizienz vorhandener Software können Sie "am lebenden Objekt" objektiv messen. Insofern ist die Prüfung, ob Effizienzanforderungen an Software erreicht werden, verhältnismäßig einfach möglich.

## Szenarien für Effizienz

### Szenario: Schnelle Erzeugung von Testdaten

Schnelle Erzeugung großer Mengen an Testdaten für das XY-System.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Effektives Testen, Test mit großen Datenmengen. |
| Auslöser | Ein Tester benötigt für den Test des XY-Systems Testdaten. |
| Reaktion | Der Testdaten-Generator erzeugt 1 Gigabyte an passenden, fachlich korrekten Testdaten. |
| Zielwert | Laufzeit weniger als eine Stunde. |

### Szenario: Diagnose hat kaum Auswirkungen auf Ausführungszeit

Das Diagnose-Subsystem beeinflusst die Ausführungszeit von Funktionen und Transaktionen des Systems nur in geringem Umfang .

|  |  |
| --- | --- |
| Geschäftsziel(e) | Genaues Reporting über Laufzeiten und genaue Fehlerdiagnose ohne Beeinträchtigung von Laufzeiten. |
| Auslöser | Benutzer, Tester oder Administrator ruft eine Diagnosefunktion auf. |
| Reaktion | Das System arbeitet ohne Einschränkung weiter. |
| Zielwert | Alle Funktionen und Transaktionen des Systems laufen funktional korrekt. Laufzeiten sind gegenüber abgeschaltetem Diagnose-Subsystem höchstens 5% höher. |

### Szenario: Generierung von Reports in weniger als 4h

Generierung aller für den Monatsabschluss erforderlichen Reports und Listen innerhalb von 4h Laufzeit.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Performanter und pünktlicher Monatsabschluss |
| Auslöser | Die Controlling- oder Finanz-Abteilung startet nach Buchungsschluss den Monatsabschluss. |
| Reaktion | Das System generiert alle notwendigen Reports und Listen. |
| Zielwert | Die Generierung endet spätestens nach 4h Laufzeit, erste (einfache) Reports stehen dem Controlling nach 30 Min Laufzeit zur Verfügung. |

### Szenario: Integrationstests in weniger als 15 Minuten

Sämtliche Integrationstests des Subsystems XY können innerhalb von 15 Minuten automatisiert ausgeführt werden.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Risikoarme Änderungen und Erweiterungen. |
| Auslöser | Entwickler führt eine Änderung am Quellcode im Subsystem XY durch und startet anschliessend die automatisierte Testsuite (der Initegrations- und Unittsts) dieses Subsystems. |
| Reaktion | Das Testframework führt sämtliche Testfälle aus und berichtet die Testergebnisse an den Benutzer. |
| Zielwert | Sämtliche Testfälle sind in weniger als 15 Minuten komplett ausgeführt. |
| Bemerkung | Last-, Performance- oder Stresstests können unabhängig von den Integrationstests auch länger laufen. Diese sind nicht Bestandteil dieses Szenarios. |

### Szenario: Daten innerhalb von 3 Sekunden

In 90% aller Fälle erhalten Benutzer die XY-Daten innerhalb von 3 Sekunden.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Paralleles Arbeiten mehrerer Benutzer ist mit akzeptabler Laufzeit möglich |
| Auslöser | 10 echt parallel arbeitende Benutzer fordern vom System die XY-Daten an. |
| Reaktion | Das System zeigt bei allen anfordernden Benutzern die korrekten Daten an. |
| Zielwert | Bei mindestens 9 von 10 dieser Benutzer dauert diese Anfrage 3 Sekunden oder weniger (in 90% der Anfragen nach den XY-Daten antwortet das System in 3 Sekunden oder schneller). |

### Szenario: Hohe Performanz bei 200 gleichzeitigen Benutzern

Bei 200 oder mehr gleichzeitig angemeldeten Benutzern verhält sich das System immer noch performant.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Fähigkeit des Systems, seine Funktionsfähigkeit, insbesondere an der GUI, auch bei mehreren parallelen Benutzern zu erhalten. |
| Auslöser | * 200 oder mehr Benutzer sind am System angemeldet. * 20 oder mehr Benutzer arbeiten gleichzeitig im XY-Dialog aus oder starten eine YZ-Berechnung. |
| Reaktion | Das System arbeitet normal und bedient sämtliche Benutzer. |
| Zielwert | Die Reaktionen des Systems im XY-Dialog erfolgen innerhalb von 2 Sekunden. Das System führt die YZ-Berechnung in weniger als 5 Sekunden durch. Bemerkung: Skalierbarkeit |

### Szenario: Umfangreiche Reports in weniger als 1 Sekunde

Das XY-System soll auch umfangreiche benutzerdefinierte Reports in weniger als 1 Sekunde speichern.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Reaktive Benutzeroberfläche steigert Bedienkomfort. |

Auslöser: Benutzer hat einen spezifischen Report konfiguriert und speichert diesen über die "speichern" Funktion ab.

Reaktion und Zielwert: Das System speichert die gesamte Reportdefinition (im xml-Format) in weniger als 1 Sekunde.

### Szenario: Konfigurationsoberfläche erscheint in weniger als 2 Sekunden

An Benutzerprofil angepasste grafische Konfigurationsoberfläche erscheint in weniger als 2 Sekunden.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Reaktive Benutzeroberfläche steigert Bedienkomfort. |
| Auslöser | Benutzer startet die Konfigurationsoberfläche für Reports. |
| Reaktion und Zielwert | Beim Start der Report-Konfigurationsoberfläche erscheint eine gemäß des Benutzeprofils aufbereitete grafische Oberfläche in weniger als 2 Sekunden. |

### Szenario: Nutzung von höchstens 1GB RAM

Die Anwendung soll als Sockel nur 1 GB RAM benötigen und pro gleichzeitig eingeloggten User nicht mehr als weitere 5 MB

|  |  |
| --- | --- |
| Geschäftsziel(e) | Die Anwendung soll zur Verfügung gestellte Systemressourcen effizient nutzen und Wachstum der Mitarbeiter ermöglichen. |
| Auslöser | Ein Benutzer meldet sich am System an. |
| Reaktion und Zielwert | Die Anwendung soll als Sockel nur 1 GB RAM benötigen und pro gleichzeitig eingeloggten User nicht mehr als weitere 5 MB. Nach dem Logout müssen die 5 MB wieder zur freien Verfügung stehen. |

# Zuverlässigkeit

## Was bedeutet Zuverlässigkeit?

Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren.

Zu Zuverlässigkeit gehören nach DIN/ISO 9126 folgende Teilmerkmale:

**Reife:**

Geringe Versagenshäufigkeit durch Fehlzustände.

**Fehlertoleranz:**

Fähigkeit, ein spezifiziertes Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren.

**Wiederherstellbarkeit:**

Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen.

## Szenarien für Zuverlässigkeit

### Szenario: Detaillierte Auskunft über Fehler

Wenn der Datenimport fehlschlägt, gibt das System detaillierte Auskunft über den/die aufgetretenen Fehler.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Fähigkeit, Fehler im Datenimport schnell zu identifzieren, lokalisieren und zu beheben. |
| Auslöser | Datenimport schlägt fehl. |
| Reaktion | System sammelt die für Fehlerdiagnose und -behebung relevanten Informationen (Art des Fehlers, betroffene Datensätze, Zeit, letzte erfolgreiche ausgeführte Aktion etc.) |
| Zielwert | Relevante Informationen werden in weniger als 30 Sekunden nach Auftreten des Fehlers ins Logfile geschrieben und per smtp-Mail an <[[x@y.com](mailto:x@y.com)](mailto:x@y.com)> geschickt. |

### Szenario: Messgenauigkeit für Zeitmessungen

Das Diagnose-Subsystem soll die gleiche Messgenauigkeit für Zeitmessungen besitzen, wie entsprechende externe Werkzeuge.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Externe Messwerkzeuge für die Ausführungszeiten von Transaktionen liefern identische Ergebnisse zum internen Diagnose-Subsystem. |
| Auslöser | Ein Benutzer stößt im System eine beliebige Transaktion an. Die interne Diagnose ist dabei auf "ein" konfiguriert. |
| Reaktion | Das interne Diagnose-Subsystem speichert die Anfangs- und Endzeit der Transaktion. |
| Zielwert | Die vom internen Diagnose-Subsystem gemessenen Zeiten stimmen im Bereich von 5% mit Werten überein, die externe Werkzeuge für diesselbe Transaktion ermittelt haben. |

### Szenario: Ausfallsicherung für Container

Das System besitzt eine Ausfallsicherung für den Servlet-Container.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Problemlose Behandlung genereller Fehler- und Ausfallsituationen |
| Auslöser | Der Servlet-Container stürzt aufgrund eines Softwareproblems ab- Hardware und Betriebssystem sind weiterhin verfügbar. |
| Reaktion und Zielwert | Das Monitoringsystem entdeckt den Ausfall innerhalb von 1 Sekunde, stellt innerhalb von 15 Sekunden einen Ersatz-Container bereit und ist nach spätestens 120 Sekunden wieder voll funktionsfähig. |
| Bemkerungen | Für ein konkretes System wäre hierbei zusätzlich zu spezifizieren, ob und in welchem Umfang die gerade aktiven Sessions des ausgefallenen Containers gesichert und auf das Ersatzsystem übertragen werden müssen. |

### Szenario: Keine Abstürze bei Speicherknappheit

Das System verarbeitet während der pdf-Generierung und Dateikonvertierung (im Speicher) Daten im Bereich bis zu mehreren Gigabyte. Sollte es zu Speicherknappheit oder -überlauf kommen, darf das System nicht abstürzen, sondern muss aussagekräftige Log-Meldungen schreiben, die Generierung kontrolliert beenden und die Benutzer darüber benachrichtigen.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Zuverlässigkeit des Systems auch bei umfangreichen und voluminösen Generierungs- und Konvertierungsaufgaben. |
| Auslöser | Das System generiert oder konvertiert Daten, eventuell verteilt auf mehrere Threads, Prozesse oder Knoten. Es tritt an mindestens einem dieser Threads, Prozesse oder Knoten ein Speicherüberlauf auf. |
| Reaktion | Das System beendet die entsprechenden Prozesse kontrolliert und speichert den Zwischenstand der Generierung/Konvertierung zur späteren Verwendung ab. Es erzeugt eine passende Logmeldung und informiert den jeweiligen Benutzer über die Situation. |
| Zielwert | Speicherüberlauf wird innerhalb von 15 Sekunden erkannt, alle beteiligten Prozesse innerhalb weiterer 15 Sekunden kontrolliert beendet. |
| Bemerkung | auch Benutzerbarkeit |

### Szenario: Funktionale Korrektheit auch bei Unterspannung

Das System verhält sich auch bei Unterspannungen der Hardware-Sensoren (bis zu 15% unterhalb der Nennspannung) in allen Belangen funktional korrekt.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Zuverlässigkeit |
| Auslöser | Die Spannung der Hardware-Sensoren (z.B. Sensor zur Messung der Papier-Transportgeschwindigkeit, der Durchflussgeschwindigkeitoä) sinkt höchstens 15% unterhalb der vorgeschriebenen Nennspannung. |
| Reaktion | Alle Systemfunktionen arbeiten korrekt weiter. |
| Bemerkung | Einige Sensor-Typen verhalten sich bei Unterspannung unkontrollierbar, manche arbeiten langsamer, andere ungenau, andere gar nicht mehr. Die Systemfunktionen müssen daher die Spannung der Sensoren überwachen und auf Spannungsprobleme entsprechend reagieren. |

### Szenario: Kein Datenverlust bei Spannungsverlust oder Unterspannung.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Robustheit gegenüber Schwankungen oder Ausfällen der elektrischen Versorgung. Das System verliert bei Spannungsverlust oder Unterspannung (der gesamten Netzversorgung) keine Daten. |
| Auslöser | Die Versorgungsspannung fällt aus oder schwankt um bis zu 25%. |
| Reaktion | Das System ist lange genug durch redundante Stromversorgung gesichert, um im Falle des Verlustes der regulären Stromversorgung noch sämtliche im Speicher befindlichen Daten konsistenz auf langfristigen Speichermedien sichern zu können. |
| Zielwert | Spannungsverlust oder Unterspannung wird innerhalb von 200msec erkannt. |

### Szenario: Das System bietet eine Ausfallsicherung des Servlet-Containers.

Geschäftsziel(e)

Das System soll Ausfälle der allgemeinen Infrastruktur (insbesondere Servlet-Container) problemlos und ohne Absturz behandeln.

Auslöser

Der (für viele Systemfunktionen notwendige) Servlet-Container fällt aus.

Reaktion

Das System erkennt den Ausfall und transferiert alle noch zur Verfügung stehenden Daten/Sessions auf einen Ersatz-Servlet-Container.

Zielwert

* Entdecke den Fehler im Servlet-Container innerhalb von 1 Sekunde.
* Fährt den Hot/Cold-Standby Servlet-Container innerhalb von 30 Sekunden hoch.
* Nach 180 Sekunden hat das System die gesamte Funktionalität von vor dem Ausfall wieder hergestellt.

### Szenario: Stabilität auch bei Dauerbenutzung

Auch im Dauerbetrieb verhält sich das System gegenüber Online-Benutzern angemessen stabil und robust.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Stabilität, Benutzerkomfort |
| Auslöser | Endbenutzer verwenden das System für mindestens 8 Stunden ohne Neustart, Abmeldung oder sonstige Unterbrechung. Dabei sind ständig mindestens 20 parallele Benutzer am System angemeldet - höchstens 1000 parallele Benutzer. |
| Reaktion | Das System funktioniert für alle angemeldeten Benutzer korrekt. Für den oder die die lange angemeldeten Benutzer gilt dies ebenfalls. |
| Zielwert | In der gesamten Zeit tritt bei den Benutzern kein Absturz auf. Anmerkung: Es muss für die Benutzer zumindest so aussehen, als verhalte sich das System stabil. Serverseitige Probleme muss das System gegenüber den Benutzern kaschieren oder durch Standby-Systeme oder Failover kompensieren können. |

# Betreibbarkeit

## Was bedeutet Betreibbarkeit?

Betreibbarkeit wird in DIN/ISO 9126 nicht definiert.

In der Praxis gehören hierzu folgende Teilmerkmale:

**Analysierbarkeit:**

Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen.

**Installierbarkeit:**

Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist.

**Übertragbarkeit:**

Eignung der Software, von einer Umgebung in eine andere übertragen zu werden. Umgebung kann organisatorische Umgebung, Hardware- oder Software-Umgebung einschließen. Teilweise als "Portabilität" bezeichnet.

**Austauschbarkeit:**

Möglichkeit, diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden, sowie der dafür notwendige Aufwand.

**Koexistenz:**

Fähigkeit der Software, neben einer anderen mit ähnlichen oder gleichen Funktionen zu arbeiten

## Szenarien für Betreibbarkeit

### Szenario:

Das Werkzeug zur Messung der aktuellen Datenbank-Performance muss sowohl unter MySQL, Oracle und DB2 laufen.

|  |  |
| --- | --- |
| Geschäftsziel(e)  Reaktion | Das Werkzeug muss eine Verbindung zu allen genannten Datenbanken aufbauen können und die jeweilige DB-Struktur anzeigen können. |
| Zielwert | Initialisierung und Erkennung des verbundenen DB-Typs erfolgt in weniger als 30 Sekunden. |

### Szenario: Automatische Installation

Das XY-System lässt sich inklusive sämtlicher benötigten Softwarekomponenten vollständig automatisiert installieren.

|  |  |
| --- | --- |
| Geschäftsziel(e) | Kurze Releasezyklen, um schnelle Erweiterungen oder Fehlerbehebungen produktiv setzen zu können |
| Auslöser | Betreiber (oder Administrator) möchte neue Version von XY (innerhalb einer neuen virtuellen Maschine bzw. neuen Hardware) installieren |
| Reaktion | Ein automatischer Installer (Skript, Programm) installiert auf Basis einer Ubuntu-Linux Standardinstallation sowohl alle Teile des XY-Systems wie auch sämtliche benötigten zusätzlichen Software- komponenten (etwa: Datenbank, Middleware, Crypto-Module). Sämtliche benötigten Daten (etwa: Datenbank-Initialisierung, LDAP-Standardbenutzer) werden mit angelegt. |
| Zielwert | Die gesamte Installation dauert auf Basis eines bereits installierten Ubuntu-Linux höchstens 30 Minuten. |

### Szenario: Langlebigkeit

Die Software kann über 20 Jahre hinweg auf jeweils aktueller Hardware und Software (Betriebssysteme, DB-Systeme usw.) portiert werden, wozu nicht mehr als 5 Personentage pro Monat investiert werden müssen.

### Szenario: Installation ohne Internetzugang

Unabhängigkeit der Installation von Zufällen des Internets

Die Software lässt sich auf einem System installieren, ohne dass ein Internetzugang vorhanden ist.

# Sonstige Qualitätsanforderungen

Da hätten wir noch eine nahzu beliebige Menge weiterer Qualitätsanforderungen, beispielsweise Funktionalität. Oder diverse Synonyme bereits vorgestellter Q-Merkmale :-)

Einige Definitionen:

## Funktionalität:

Vorhandensein von Funktionen mit festgelegten Eigenschaften; diese Funktionen erfüllen die definierten Anforderungen. Zu Funktionalität gehören nach DIN/ISO 9126 noch folgende Teilmerkmale:

**Richtigkeit**

Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen, z.B. die benötigte Genauigkeit von berechneten Werten.

**Angemessenheit**

Eignung der Funktionen für spezifizierte Aufgaben, z.B. aufgaben-orientierte Zusammensetzung von Funktionen aus Teilfunktionen.

**Interoperabilität**

Fähigkeit, mit vorgegebenen Systemen zusammenzuwirken. Hierunter fällt auch die Einbettung in die Betriebsinfrastruktur.

**Ordnungsmäßigkeit**

Erfüllung von anwendungsspezifischen Normen, Vereinbarungen, gesetzlichen Bestimmungen und ähnlichen Vorschriften.

## Sonstige Szenarien

### Szenario: Verfügbarkeit für unterschiedliche Datenbanksysteme

Das System (Werkzeug) dient zur Performancemessung für Datenbanksysteme. Es muss für unterschiedliche Datenbanksysteme zur Verfügung stehen.

|  |  |
| --- | --- |
| Geschäftsziel | Portabilität, Effizienz, Betreibbarkeit. Vielseitiges Werkzeug, soll für unterschiedliche Datenbanksysteme zur Verfügung stehen |
| Auslöser | Das Werkzeug wird (per Dialog, Kommandozeile oder programmatisch) mit einem Datenbanksystem verbunden ("connected"): Oracle, DB2, Sybase, MySQL, PostgreSQL |
| Reaktion | Das Werkzeug stellt die Verbindung zum jeweiligen DBMS erfolgreich her - im Werkzeug ist die DB-Struktur sichtbar. |
| Zielwert | Der "connect" zur Datenbank erfolgt innerhalb von 30 Sekunden. |

### Szenario: Kernfunktion portabel auf iOS und Android

Die Kernfunktionen der Mac-OS Software können unter iOS und Android wiederverwendet werden.

|  |  |
| --- | --- |
| Qualitätsziele | Wiederverwendbarkeit, Austauschbarkeit |
| Geschäftsziel | Schnelle Time-to-Market, reduzierte Entwicklungskosten, Marktvergrößerung |
| Auslöser | Code ist für Mac-OS entwickelt |
| Reaktion | Der Code ist weitmöglich (soweit die Technologie das zulässt) sowohl unter iOS und Android wieder verwendbar. |
| Zielwert | Das Entwicklungsteam soll die Kernfunktionen wiederverwenden, ohne sie komplett neu implementieren beziehungsweise entwerfen zu müssen. |

Ausnahme hiervon sind direkte Aufrufe der jeweiligen Betriebssystemfunktionen.

### Szenario: 60% Testabdeckung für Unit-Tests.

|  |  |
| --- | --- |
| Qualitätsziele | Testbarkeit, Änderbarkeit |
| Geschäftsziel | Einfache Erweiter- und Änderbarkeit durch hohe Testabdeckung, schnelle Rückmeldung über mögliche Nebenwirkungen bei Codeänderungen. |
| Auslöser | Entwickler entwickelt oder ändert eine Funktion / Methode / Klasse. |
| Reaktion | Die betroffene Funktion / Methode / Klasse wird durch Unit-Tests überprüft. |
| Zielwert | Über 60% Pfadabdeckung wird durch die Unit-Tests erreicht. |

### Szenario: Einfaches Hinzufügen neuer Tests.

Tester sollen mit geringem Aufwand neue Tests zu bestehenden Testsuites hinzufügen können.

|  |  |
| --- | --- |
| Qualitätsziele | Testbarkeit, Änderbarkeit |
| Geschäftsziel | Fähigkeit, neue Tests einfach in bestehende Test-Suites zu integrieren. |
| Auslöser | Tester möchte neuen Test zu einer bestehenden Testsuite hinzufügen |
| Reaktion | Der Test wird zugefügt ohne den Code des Systems selbst (d.h. Den Produktivcode) zu modifizieren. |
| Zielwert | Es ist kein Re-Compile oder Neukonfiguration des Systems (genauer: des Produktivcodes) nötig. |
| Anmerkung | Testcode oder Testkonfigurationen dürfen jedoch verändert werden. |

### Szenario: Ähnliche Resultate bei stochastischen Testreihen

Werden zwei stochastische Testreihen ausgeführt, sind die Resultate zu 90% ähnlich.

|  |  |
| --- | --- |
| Qualitätsziele | Testbarkeit, Konsistenz, Nachvollziehbarkeit. Konsistente Testergebnisse auch bei stochastischen Tests oder Testreihen. |
| Auslöser | Testszenario oder Testreihe mit Anteil an zufällig bestimmten Testdaten wird ausgeführt |
| Reaktion | Ähnliche Testergebnisse. |
| Zielwert | Die Ergebnisse zweier Testreihen sind bei 90% aller Einzeltests ähnlich. |

### Szenario: Testausführung mit einelnem Befehl

Ein Tester möchte mehrere Testszenarien mit einem einzelnen Befehl durchführen können.

|  |  |
| --- | --- |
| Geschäftsziel | Effizienz, Testbarkeit |
| Auslöser | Testadministrator oder Tester möchte mehrere Tests oder Testszenarien durchführen |
| Reaktion | Aggregation oder Hintereinander-Ausführung mehrerer Tests. |
| Zielwert | Ausführung mehrerer Tests benötigt nur einen einzigen Befehl des Testers oder Testadministrators. |

### Szenario: X-Daten für externe Systeme zugänglich

Die X-Daten des Y-Systems sollen für ein externes Werkzeug zur Anforderungsanalyse und -management (etwa: Requisite-Pro, Enterprise-Architect oä) zugänglich sein.

|  |  |
| --- | --- |
| Qualitätsziel | Interoperabilität |
| Geschäftsziel(e) | Datenkompatibilität mit marktüblichen Werkzeugen zur Anforderungsanalyse zählt als Vorteil gegenüber Mitbewerbern. |
| Auslöser | Ein Endbenutzer möchte die X-Daten des Y-Systems in eines der unterstützten Anforderungstools überführen. |
| Kontext | 20 Benutzer haben X-Daten in Form einzelner Projekte im System erfasst. Jedes dieser Projekte enthält mindestens ein, höchstens 100 unterschiedliche Requirements. |
| Reaktion | Das Y-System exportiert die betreffenden X-Daten in das Anforderungstool (Requisite-Pro oder Enterprise-Architect). |
| Zielwert | * Beim Export der Daten treten keine Fehler auf. * Von den im Y-System enthaltenen Requirements werden mindestens 98% korrekt exportiert. * Sämtliche nicht exportierten Requirements werden den betroffenen Benutzern als Fehler gemeldet. |

# Anhang: Q-Merkmale

Draußen im Dschungel der Realität warten Dutzende verschiedener Qualitätsanforderungen oder Qualitätsziele auf ihre Erfüllung. Hier der Versuch, diese etwas präziser zu definieren und zu kategorisieren.

|  |  |  |
| --- | --- | --- |
| Q-Merkmal | Bedeutung | Oberbegriffe |
| Absturzsicherheit |  | Zuverlässigkeit |
| Administrierbarkeit | Benötigter Aufwand zur Administration (Verwaltung) einer Software | Betreibbarkeit |
| Analysierbarkeit | Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen | Änderbarkeit |
| **Änderbarkeit** | Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen sind Korrekturen, Verbesserungen oder Anpassungen der Umgebung, Infrastruktur, Betriebsmittel, der Anforderungen, der internen Struktur, der Implementierung oder technischer Konzepte. |  |
| Angemessenheit | Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen, z.B. die benötigte Genauigkeit berechneter Ergebnisse. | Funktionalität |
| Anpassbarkeit | Fähigkeit der Software zur Anpassung an verschiedene Umgebungen oder Nutzungsszenarien. | Änderbarkeit |
| Antwortzeit | Zeit, bis die Software ein gewünschtes Ergebnis erzielt oder errechnet. | Effizienz |
| Auditierbarkeit |  | Prüfbarkeit |
| Ausfallsicherheit | Fähigkeit des Systems, auch bei Ausfall einiger Bestandteile (Hardware oder Software) ein definiertes Leistungsniveau erhalten zu können. Im Zuge von verteilten Systemen auch: Toleranz gegenüber partiellen Ausfällen einzelner Netzsegmente. | Zuverlässigkeit, Partitionstoleranz |
| Ausschaltzeit |  | Effizienz |
| Austauschbarkeit | Möglichkeit (und der dafür nötiger Aufwand), diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden. | Übertragbarkeit |
| Autonomie | Fähigkeit eines Systems, sein Leistungsniveau unabhängig von anderen Systemen zu erbringen. | **Betreibbarkeit** |
| Bedienbarkeit | Aufwand für den Benutzer, die Anwendung zu bedienen. | Benutzbarkeit |
| **Benutzbarkeit** | Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe. |  |
| Benutzerfreundlichkeit | Siehe Benutzbarkeit | Benutzbarkeit |
| Berechenbarkeit |  | Zuverlässigkeit |
| Betreibbarkeit | \* Fähigkeit zur Erfüllung vorgegebener Aufgaben im Rahmen allgemeiner Anforderungen an Wirtschaftlichkeit, Verfügbarkeit, Sicherheit etc. \* Möglichkeit das System mit wirtschaftlich und organisatorisch angemessenem Aufwand in seiner Ablaufumgebung zu betreiben | Administrierbarkeit |
| Datensicherheit |  | Sicherheit |
| Durchsatz |  | Effizienz |
| **Effizienz** | Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen | Performance |
| Einfachheit | - Einfache Bedienbarkeit - Einfache Änderbarkeit | Bedienbarkeit, Änderbarkeit, |
| Einheitlichkeit |  | Verständlichkeit, Bedienbarkeit, Änderbarkeit |
| Erlernbarkeit | Aufwand für den Benutzer, die Anwendung zu erlernen (z.B. Bedienung, Ein-, Ausgabe) | Bedienbarkeit |
| Erweiterbarkeit |  | Änderbarkeit |
| Fehlertoleranz | Fähigkeit, ein spezifiziertes Leistungsniveau bei Softwarefehlern oder Nichteinhaltung spezifizierter Schnittstellen zu bewahren | Zuverlässigkeit |
| Flexibilität | Anpassungsfähigkeit an wechselnde Umstände. Möglichkeit zur einfachen Einarbeitung von Änderungen, Erweiterungen oder Behebung von Fehlern. | Änderbarkeit |
| **Funktionalität** | Vorhandensein von Funktionen mit festgelegten Eigenschaften;diese Funktionen erfüllen die definierten Anforderungen |  |
| Gefahrlosigkeit |  | Zuverlässigkeit |
| Genauigkeit |  | Zuverlässigkeit, Funktionalität |
| Geschwindigkeit |  | Effizienz |
| Glaubwürdigkeit | Maß der Bereitschaft von Benutzern eines Systems, dessen Ergebnisse als gültig zu akzeptieren. | Zuverlässigkeit, Robustheit |
| Größe | Umfang der Software, etwa in Lines-of-Code oder in Byte | Effizienz |
| Gültigkeit | i.d.R. bezogen auf Daten | Zuverlässigkeit, Funktionalität |
| Installierbarkeit | Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist | **Übertragbarkeit**, Betreibbarkeit |
| Integrität |  |  |
| Interoperabilität | Fähigkeit, mit vorgegebenen Systemen zusammenzuwirken.Hierunter fällt auch die Einbettung in die Betriebsumgebung oder technische Infrastruktur. | Kompatibilität |
| Konfigurierbarkeit |  | Betreibbarkeit, Änderbarkeit |
| Konformität | Grad, zu dem die Software Normen oder Vereinbarungen erfüllt. Differenziert nach Merkmalen (etwa bezüglich Normen zur Sicherheit oder Zuverlässigkeit) |  |
| Konsistenz | Synonym: Integrität. \* Bezüglich Daten: + Maß, in dem Daten sowie deren Beziehungen deren Gültigkeitsregeln genügen. + Clienten einer Datenbank erhalten bei identischen Anfragen identische Ergebnisse. \* Bezüglich Verhalten: Maß, in dem sich ein System schlüssig und nachvollziehbar verhält.  Weitere Verfeinerung: Monotonic-Read-Consistency, Montonic-Write-Consistency, Read-Your-Writes-Consistency, Write-Follows-Read-Consistency. Siehe Erläuterungen zum CAP-Theorem. <http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed> | Integrität |
| Korrektheit | Eigenschaft eines Systems, seiner Spezifikation zu genügen. | Funktionalität, Zuverlässigkeit |
| Latenz | Synonym: Verzögerungszeit. Zeit vom Ende eines Ereignisses bis zum Beginn der Reaktion auf dieses Ereignis. | Effizienz |
| Laufzeiteffizienz | Sparsamkeit eines Systems (meist: eines Algorithmus) bezüglich der Resource "Rechenzeit" | Effizienz |
| Leistungsfähigkeit | Fähigkeit eines Systems, spezifizierte Dienste oder Leistungen zu erbringen. | Effizienz |
| Lokalisierbarkeit | Anpassungsfähigkeit an landes- oder sprachspezifische Anforderungen. | Bedienbarkeit, Änderbarkeit, Flexibilität |
| Modifizierbarkeit | Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder Anpassung an Umgebungsänderungen. | Änderbarkeit |
| Modularität | Zerlegung eines Systems in Einzelbausteine mit definierten Schnittstellen. | Änderbarkeit |
| Nachvollziehbarkeit |  |  |
| Nichtabstreitbarkeit |  | Sicherheit |
| Nichtangreifbarkeit |  | Sicherheit |
| Normgerechtigkeit | Siehe Konformität. | Konformität |
| Ordnungsmäßigkeit | Erfüllung von anwendungsspezifischen Normen, Vereinbarungen, gesetzlichen Bestimmungen und ähnlichen Vorschriften | Funktionalität |
| Partitionstoleranz | Das System arbeitet auch bei Ausfall einzelner Knoten, Netzsegmente oder sonstiger Systembestandteile weiter. Begriff wird insbesondere im Zusammenhang mit dem [CAP-Theorem](http://en.wikipedia.org/wiki/CAP_theorem) und verteilten Datenbanken ([NoSQL-DB](http://nosql-database.org/)) verwendet. | **Zuverlässigkeit**, Ausfallsicherheit |
| Performanz | Siehe Effizienz. | Effizienz |
| Personalisierbarkeit |  | Änderbarkeit, Betreibbarkeit |
| Portabilität | Grad der Plattformunabhängigkeit | Übertragbarkeit |
| Prüfbarkeit | Aufwand, der zur Prüfung der Software notwendig ist, insbesondere nach Änderungen | Zuverlässigkeit |
| Reaktionszeit |  | Effizienz |
| Reife | Geringe Versagenshäufigkeit durch Fehlzustände | Zuverlässigkeit |
| Richtigkeit | Eignung der Funktionen für spezifizierte Aufgaben. | Korrektheit |
| Robustheit |  | Zuverlässigkeit |
| **Sicherheit** | Fähigkeit, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern. |  |
| Skalierbarkeit | Fähigkeit eines Systems, unter Nutzung zusätzlicher Resourcen seine Kapazitäten zur Leistungserbringung zu steigern. | Effizienz |
| Stabilität | Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen, entweder aufgrund von Benutzung oder Änderungen | Zuverlässigkeit, Robustheit |
| Startup-Zeit | Zeit, die das System zum Start, d.h. bis zum Herstellen der (vollständigen oder teilweisen) Betriebsbereitschaft benötigt. | Effizienz |
| Strapazierfähigkeit |  | Zuverlässigkeit |
| Testbarkeit |  | Zuverlässigkeit |
| Überprüfbarkeit |  | Zuverlässigkeit |
| **Übertragbarkeit** | Wie leicht lässt sich die Software in eine andere (Hardware-, Software- oder organisatorische) Umgebung übertragen? |  |
| Überwachbarkeit | Fähigkeit, (Betriebs-)Ablauf, Störungsfreiheit, Auslastung, Ressourcennutzung oder sonstige Eigenschaften zur Laufzeit ohne funktionale Beeinträchtigung zu beobachten. | Betreibbarkeit |
| Unterstützbarkeit |  | Betreibbarkeit |
| Verbrauchsverhalten | Anzahl und Dauer der benötigten Betriebsmittel für die Erfüllung der Funktionen | Effizienz |
| Verfügbarkeit |  | Zuverlässigkeit, Robustheit |
| Verständlichkeit | 1.) Externe : Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen 2.) Interne: Aufwand, die interne Struktur, deren Konzepte und Implementierung zu verstehen | 1.) Benutzbarkeit 2.) Wartbarkeit, |
| Verteilbarkeit |  | Betreibbarkeit |
| Vertraulichkeit |  | Sicherheit |
| Vorhersagbarkeit |  | Zuverlässigkeit |
| Wartbarkeit | Welchen Aufwand erfordert es, vorgegebene Änderungen an der Software durchzuführen? | Änderbarkeit |
| Wiederherstellbarkeit | Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen. | Zuverlässigkeit |
| Wiederverwendbarkeit | Eigenschaft eines Systems oder Bausteins, auch ausserhalb des ursprünglich geplanten Einsatzzwecks oder -ortes verwendet zu werden. | Flexibilität, Portabilität |
| Zeitverhalten | Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung | Effizienz, Performance |
| Zugriffsschutz | Maßnahmen gegen unerwünschten Zugriff auf Resourcen oder Systemteile | Sicherheit |
| **Zuverlässigkeit** | Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren | Robustheit |