**CHECKLISTE RISIKOFAKTOREN JE QUALITÄTSMERKMAL**

# Checkliste Risikofaktoren je Qualitätsmerkmal

Im Prinzip könnte man für jedes einzelne Qualitätsmerkmal eine eigene Risikoanalyse durchführen. Das kann jedoch sehr zeitintensiv werden. Die folgende Checkliste zeigt die spezifischen Gründe dafür, ein Qualitätsmerkmal in Betracht zu nehmen. Die Beschreibung beschränkt sich auf die Fehlerwahrscheinlichkeit und den Schaden, die zwei wesentlichen Faktoren des Risikos. Der dritte Faktor, die Nutzungshäufigkeit, ist nicht spezifisch für ein Qualitätsmerkmal.

Durch die Schäden betroffen kann das Unternehmen selbst sein, aber auch die Kunden/Käufer von Software oder von Softwareunterstützten Dienstleistungen. Kunden erleiden Schäden durch die fehlerhafte Software zum Beispiel im Internet, am Geldautomaten, durch fehlerhafte Information des Dienstleisters (wie etwa falsche Rechnungen), Hardwarestörungen, usw. Falls die Kunden dem Hersteller der Software die Schuld hierfür zuweisen, dann entsteht indirekter Schaden. Falls das Unternehmen den für den Kunden entstanden Schaden übernimmt, dann erleidet das Unternehmen einen direkten Schaden.

Im Folgenden ist das Schadenpotenzial zum einem allgemein und zum anderen spezifisch für jedes Qualitätsmerkmal aufgelistet.

Darüber hinaus sind Faktoren aufgelistet, die die Fehlerwahrscheinlichkeit erhöhen.

NB: Cem Kaner erwähnt in seinem Buch Testing Computer Software 400 Möglichkeiten von technischen Fehlerarten. [Kaner, 1999]

## Allgemeine Faktoren

### Schaden

* Imageschaden der Marke und/oder Firma.
* Verschlechterung der Wettbewerbsposition.
* Umsatzverlust.
* Schadensersatzforderungen.
* Kosten für Korrekturmaßnahmen, sowie für zusätzliche Büroarbeit.
* Leerlauf, Mitarbeiter können ihre Tätigkeit mit dem System nicht durchführen.
* Kunden können nicht unterstützt werden.
* Helpdesk-Unterstützung.
* Strafgebühren, falls Anforderungen nicht erfüllt werden.
* Der Geschäftsprozess ist weniger Effizient oder sogar ungeeignet.
* Es entsteht ein negatives Bild über die IT.
* Wartezeiten für Dritte, weil das Information liefernde Unternehmen zu langsam ist.

### Erhöhte Fehlerwahrscheinlichkeit

* Anforderungen fehlen.
* Qualitätsmerkmal wird im Design nicht beachtet.
* Qualität des Designs ist mangelhaft.
* Designvorgaben werden nicht befolgt.
* Komplexität ist hoch, mit den Untermerkmalen: Große Codeblöcke, viele verwendete Variablen, Komplexe Logik, viele kleine Module, die miteinander gekoppelt sind, Verwendung und Überlappung mehrerer Computersprachen. Der Grad der gefühlten Komplexität hängt oft von den Fähigkeiten der Entwickler ab.
* Oft geänderte Software ist fehleranfällig.
* Neue Technologie(n) (Programmiersprachen, Entwicklertools, Softwarepaket für Netzwerk und Datenbankenabwicklung, etc.) erfordern einen Lernprozess bei den Entwicklern. Manchmal ist die Technologie noch nicht ganz ausgereift und verursacht Fehlerwirkungen.
* Die Größe des Entwicklerteams: Je größer das Team, desto größer ist das Risiko von Missverständen.
* Die Fähigkeiten des Entwicklerteams: der Grad an Erfahrung bestimmt zu einem großen Teil den Anteil an gefundenen Fehlern.
* Wissenserhalt im Entwicklerteam. Besonders wenn nicht dokumentiert wird, sind Personalwechsel bei den Entwicklern eine Quelle für Fehler.
* Zeitdruck.
* Verteilung des Teams auf mehrere Standorte vermindert gute Kommunikation und erhöht die Fehlerrate.
* Eingeschränktes Wissen über existierende produktive Systeme.
* Unausgereifte IT-Organisation.
* Eine hohe Anzahl von Schnittstellen zwischen den Systemen.
* Steuerung basiert vor Allem auf Zeit und Geld, weniger auf Qualität.
* Testaktivitäten erhalten kaum Beachtung (oder es wird wenig und / oder unstrukturiert getestet).
* Siehe auch TMap Kapitel 9 „Produktrisikoanalyse“, Abschnitt 9.1 „Einführung“, darin die Beschreibung der Faktoren Fehlerwahrscheinlichkeit und Schaden.

## Verbindungsfähigkeit

### Schaden

In der eigenen Organisation:

* Erhöhte Wartungskosten aufgrund von zusätzlichen Brücken zwischen den Applikationen.
* Erhöhte Wahrscheinlichkeit von fehlerhaften oder inkonsistenten Daten.
* Probleme beim Bewahren der Synchronität zwischen den Systemen und Daten.
* Redundante Daten (durch schlechte Verbindungen werden viele Daten in “privaten” Systemen geführt).

Bei anderen Organisationen:

* Daten werden zu spät empfangen; gestörter Geschäftsprozess.

### Erhöhte Fehlerwahrscheinlichkeit

* Entwicklung ohne technischer und funktioneller Architektur oder nicht konform zur Architektur.
* Es gibt keine Datenkontrakte oder -Vereinbarungen (Protokolle für Schnittstellen fehlen).
* Keine Dateneigentumsrechte.
* Viele verschiedene (technische) Plattformen.

## Kontinuität

### Schaden

In der eigenen Organisation:

* Umsatzverlust.
* Wiederaufbaukosten.
* Zusätzliche Ausweicheinrichtungen müssen eingerichtet werden (davor oder danach).
* Zusätzliche Systemmanagementkosten / höherer Personalbedarf.

Bei anderen Organisationen:

* Forderungen.
* Nichterfüllung.

### Erhöhte Fehlerwahrscheinlichkeit

* Keine Beachtung für Speichermanagement (Memory Leak).
* Verschiedene Softwareversionen auf den Desktops (dll Versionen).
* Keine Transparenz über die Einschränkungen (Potentielle Engpässe) des Systems.
* Kein Sicherungs- und Wiederherstellungsverfahren (Backup/Restore).
* Größtenteils ’fette’ Serverarchitektur (Überbrückung von Ausfällen durch dezentrale Prozesse kaum möglich).
* Software läuft 24h / 7 Tage die Woche.
* Viele Wartung (bzw. Wartungsreleases) von Nöten.
* Keine Sicherheitsmaßnahmen gegen Denial-of-Service Attacken auf das System.

## Kontrollierbarkeit der Daten

### Schaden

In der eigenen Organisation:

* Keine Konformität zu den Vorschriften an die Buchhaltung.
* Fehler mit finanzieller Haftung möglich.
* Verlängerte Dauer der Wiederherstellung.
* Dateninkonsistenz möglich (möglicherweise ohne es zu bemerken).

Bei anderen Organisationen:

* Empfangen inkorrekter Information.

### Erhöhte Fehlerwahrscheinlichkeit

* Keine Kommunikation mit dem Finanzwesen (z.B. Buchhaltung, System Controller, EDP Auditoren).
* Keine Verwendung von Kontrollsummen.
* Keine Daten-Änderungsverfolgung und -Markierung (Zeitstempel in den Datensätzen).

## Effektivität

### Schaden

In der eigenen Organisation:

* Geringere Produktivität, sowohl Online (tagsüber) als auch Batch (Nachtlauf nicht rechtzeitig beendet).
* Zusätzliche Infrastruktur benötigt.

Bei anderen Organisationen:

* Unzufriedene Benutzer bzw. Kunden.

### Erhöhte Fehlerwahrscheinlichkeit

* Einsatz neuer Technologie oder stark geänderter Technologie.
* Verarbeitung von großen Datenmengen gemessen an den Standards der Infrastrukturkomponenten.
* Ungeeignete, nicht erweiterbare Infrastruktur.
* Keine Transparenz über den aktuellen und künftigen Gebrauch.

## Effizienz

### Schaden

In der eigenen Organisation:

* Vermehrte Verwendung von knapper und teurer Computerbetriebsmittel: CPU, Netzwerk, Datenbankkanälen (IO-Zeit).

Bei anderen Organisationen:

* Könnte zu unzureichender Leistungsfähigkeit führen.

### Erhöhte Fehlerwahrscheinlichkeit

* Entwickler haben mit den Komponenten der technischen Architektur wenig Erfahrung.

## Flexibilität

### Schaden

In der eigenen Organisation:

* Wartungskosten steigen, infolge der Implentierung und dem Testen vieler Änderungen.
* Nutzer ist vom System stark abhängig.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* Anpassungen werden im Design nicht vorhergesehen.

## Funktionalität

### Schaden

In der eigenen Organisation:

* Geschäftsprozess wird nicht unterstützt.
* Zusätzliche manuelle Aktivitäten erforderlich.
* Zusätzliche Zwischenlösungen (Workarounds) benötigt.
* Zusätzliche Ressourcen erforderlich.
* Fehlerhafte Kalkulationen.

Bei anderen Organisationen:

* Die gleichen Auswirkungen wie bei der Organisation falls die Daten kommuniziert wurden.

### Erhöhte Fehlerwahrscheinlichkeit

* Siehe allgemeine Liste.

## (Eignung der) Infrastruktur

### Schaden

In der eigenen Organisation:

* Einbindung der Software in der Produktionsumgebung erfordert mehr Aufwand.
* Upgrades von Infrastrukturkomponenten verursachen mehr Wartungsarbeit.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* (Erwartete) Zunahme der Nutzung des Systems (mehr Anwender, mehr Tätigkeit).

## Wartbarkeit

Dieses Qualitätsmerkmal ist nur dann relevant, wenn sich die Organisation für das Total Cost of Ownership (TCO) interessiert (= bewusstes Balancieren zwischen Kosten der Entwicklung / Qualität der Software / Wartungskosten).

### Schaden

In der eigenen Organisation:

* Unstrukturierte Software leidet unter hohen Wartungskosten.
* Unnötig lange Reparaturzeiten.
* Notwendige Anpassungen an neue Verordnungen oder gesetzliche Vorgaben können nicht rechtzeitig implementiert werden.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* Keine Standards.
* Nichtbefolgung von Standards.
* Z.B. C++ ist relativ unfreundlich gegenüber einer Wartung [Les Hatton 19xx]).
* Viele Altlast-Systeme.
* Verantwortlichkeit der Wartung liegt bei einer anderen Abteilung als der Entwicklung.
* Arbeiten unter hohem Zeitdruck (Struktur und Wartung kosten Geld und sind Zeitintensiv).
* Schlechte Systemdokumentation.

## Verwaltungsfähigkeit

### Schaden

In der eigenen Organisation:

* Zusätzlicher Systemmanagementaufwand.
* Wartezeit falls das Systemmanagement nicht ausreichend schnell auf Schwierigkeiten reagieren kann.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* Mangelnde Einbindung der Systemmanager in das Projekt.
* Keine Anforderungen oder Akzeptanzkriterien von Seiten der Systemmanager.
* Eingeschränkte Einsicht in die Benutzung des Systems von Seiten der Systemmanager.
* Kein oder nur minimales Konfigurationsmanagement.
* Kein Überblick über betriebliche Kosten.

## Leistungsfähigkeit (Performance)

### Schaden

In der eigenen Organisation:

* Geringe Produktivität sowohl online (Wartezeiten) als auch im Batch (Nachtveränderung wird nicht rechtzeitig fertig).
* Zusätzliche Infrastruktur erforderlich.

Bei anderen Organisationen:

* Benutzer/Kunden verlassen die Anwendung.

### Erhöhte Fehlerwahrscheinlichkeit

* Neue Technologie oder größeres Release neuer Technologie.
* Verarbeitung großer Datenmengen; groß in Bezug auf den Standard für die infrastrukturellen Komponenten.
* Infrastruktur ist unpassend.
* Infrastruktur kann nicht vernünftig erweitert werden.
* Kein Einblick in die gegenwärtige und zukünftige Benutzung.

## Übertragbarkeit (Portierbarkeit)

### Schaden

In der eigenen Organisation:

* Zusätzliche Kosten für das Nachrüsten für andersartige Plattformen (Hardware oder Betriebssysteme).
* Keine Flexibilität für das Einspielen auf neuen Plattformversionen (im Bezug auf Softwarepakete).

Bei anderen Organisationen:

* Verbindung zum System ist nicht möglich.

### Erhöhte Fehlerwahrscheinlichkeit

* Viele aneinander gekoppelte Systeme operieren auf alten Plattformen (geringe Übertragbarkeit der Schnittstellen, womit die Systemflexibilität eingeschränkt wird).
* Ausgedehnte Kenntnisse über spezifische Plattformen (Festhalten an der aktuellen Situation, geringe Flexibilität oder Bereitschaft zur Betrachtung neuer Entwicklungen).
* Geringes Allgemeinwissen über Plattformen.

## Wiederverwendbarkeit

Wiederverwendbarkeit ist für solche Organisationen von Gewicht, die ihre Produkte/Komponenten wiederverwenden möchten.

### Schaden

In der eigenen Organisation:

* Erweiterung von Systemen wird teurer.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* Unstrukturiertes Anwendungsdesign.
* Schlechte Dokumentation.
* Insel-Organisation.
* Mangelhaftes Wissensmanagement.
* Komponenten von verschiedenen Herstellern.

## Sicherheit

### Schaden

In der eigenen Organisation:

* Verschwundene und/oder fehlerhafte Dateien.
* Aufgedeckte Daten: z.B. Verstoß gegen die Vertraulichkeit von Daten im Auftrag von Konkurrenten beim Produzieren neuer Kreditkarten.
* Verletzung gegen Vertraulichkeitsbestimmungen für schützenswerte Personendaten.
* Inkonsistente Datenbank(en) (Sicherheit bezieht sich auch auf den Zufluss von Daten ins System).

Bei anderen Organisationen:

* Andere mit Viren infizieren, z.B. Kunden.
* Verletzung von Datenschutzbestimmungen.

### Erhöhte Fehlerwahrscheinlichkeit

* System zur Außenwelt verbunden, z.B. via dem öffentlichen Netzwerk (Internet).
* Unkontrollierte Zugangsmöglichkeiten.
* Kein Gebrauch von Verschlüsselungen.
* Unterbesetzte Systemadministration, wodurch Sicherheitslücken schließende Patches nicht installiert werden können.
* Diebstahl von Benutzeridentifikation (Code und / oder Pass) und Passwörtern.
* Keine Anti-Virus Richtlinien.
* Keine Sicherheitspolitik und -Architektur.

## Angemessenheit

### Schaden

In der eigenen Organisation:

* (Besondere) operative Abläufe werden vom System nicht unterstützt (zum Beispiel: Korrektur nach einer Korrektur kann nicht durchgeführt werden). Dies verursacht arbeitsintensive Notlösungen (Workarounds) oder inkorrekte Daten im System.
* Rechtzeitiger Zugriff auf Daten kann nicht verwirklicht werden (finde den richtigen Hans Schmitz in Köln).
* Im ungünstigsten Fall kann das System nicht eingeführt werden.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* Formularentwürfe und Dialog-/Report-Design erfolgen in unterschiedlichen (Teil-) Projekten mit wenig oder ohne Kommunikation.
* Keine Analyse der Informationszugriffspfade beim Entwurf.
* Brüchiges Verhältnis zwischen Systementwurf und Beschreibung der Administrativ- und Geschäftsprozesse.

## Testbarkeit

### Schaden

In der eigenen Organisation:

* Softwarequalität ist schlecht.
* Viele Unterbrechungen/Störungen.
* Test wird teurer bzw. gar undurchführbar.

Bei anderen Organisationen:

* Nicht anwendbar.

### Erhöhte Fehlerwahrscheinlichkeit

* Keine Designstandards.
* Schlechte Dokumentation.

## Benutzerfreundlichkeit

### Schaden

In der eigenen Organisation:

* Geringere Produktivität.
* Erhöhte Lernkurve / extra Trainingskosten.
* Höhere Belastung des Helpdesk / der Anwendungsmanager.
* Höhere Wahrscheinlichkeit der Überanstrengung (RSI-Syndrom).

Bei anderen Organisationen:

* Bei vorhandenen Konkurrenzprodukten wird die benutzerfreundlichere Option gewählt.

### Erhöhte Fehlerwahrscheinlichkeit

* Benutzer ist nicht in das Design der Benutzeroberfläche eingebunden.
* Keine Gestaltungsvorgaben (Styleguide) für die Entwickler.
* Geringe Entwicklungsdisziplin: Standards werden nicht verwendet, z.B. Windows Standard OK-Schalter (links) und Abbruch-Schalter (rechts) werden vertauscht.