Freie Universität Berlin
Proseminar Technische Informatik
Georg Wittenburg, M.Sc.

Seminararbeit
Fehlertoleranz und Zuverlässigkeit in der Technik

von Sebastian Oliver Kalwa (4221843)

1. Einleitung

Seit Jahrzehnten nimmt die Bedeutung der Technik im Leben des Menschen zu. Wir haben aber auch schon oft genug erlebt, dass diese Technik nicht mehr funktionieren wollte wie sie sollte. Aus Kostengründen sind viele Geräte so konstruiert, dass der Ausfall eines Teils zum Ausfall des gesamten Gerätes führt. Während ein kaputter DVD-Player wegen einer beim Transport abgelösten Teil im Innern ärgerlich ist, ist die Funktionstüchtigkeit mancher Dinge (manchmal auch unter schwereren Bedingungen) wie z.B. Flugzeuge auch bei Ausfall mehrerer Komponente überlebenswichtig für die Menschen, die gerade dieses nutzen.

Ein wichtiger Punkt um diese Fehlertoleranz zu erreichen ist die Redundanz. Im Flugzeug im Cockpit sind alle wichtigen Informationssysteme mehrfach vorhanden, so dass ein Pilot bei Ausfall eines Systems nahezu problemlos auf andere ausweichen kann. Allerdings gibt es auch in der Natur Redundanz wie es bereits der Mensch vormacht. Viele Organe sind paarweise vorhanden, so dass der Körper auf Teile verzichten kann. Wie man merkt ist Zuverlässigkeit kein neues Thema und hat die Lebewesen selbst schon vor Millionen von Jahren interessiert.

Ein Beispiel aus Quelle 5, das ich nennen möchte ist die Maschine Therac-25, die in der Strahlentherapie eingesetzt wurde. Redundanzmaßnahmen in der Hardware wurden durch fehlerhafte Sicherheitsmaßnahmen in der Software ersetzt, so dass es zu mehreren Strahlenunfällen mit Todesfolge kam.

Als Weiteres Beispiel sind vielleicht explodierende Handyakkus bekannt, da am Kurzschluss-Schutz gespart wurde. Dies führte in der Vergangenheit sogar zu Todesfällen (siehe Quelle Quelle 6).

Anwendungsbereiche für fehlertolerante Systme nach Quelle 2:

1. Hochzuverlässige Systeme müssen jederzeit verfügbar sein. Das Telefonnetz darf z.B. nur wenige Minuten im Jahr ausfallen. Flugüberwachungssysteme müssen zusätzlich noch in Echtzeit reagieren können

2. Wartungsfreie Systeme müssen von alleine möglichst lange durchhalten. Als Beispiel kann man Satelliten nennen, da hier eine Wartung sehr kompliziert und teuer ist.

3. Systeme mit festgelegten Wartungsintervallen müssen diese überstehen können. Ein Transportschiff wird nur im Heimathafen gewartet und muss die Zeit dazwischen seinen Ziel- und Heimatort erreichen können

4. Hochleistungsrechner haben eine große Zahl von parallel arbeitenden Komponenten, so dass einzelne Komponentenausfälle wahrscheinlich sind

5. Sicherheitsrelevante Anwendungen wie der Bordcomputer eines modernen Kampfflugzeugs muss während dem Flug immer funktionieren, da das Flugzeug instabile Flugeigenschaften hat und somit für einen Menschen ohne Computerhilfe praktisch unlenkbar ist.

In Abschnitt 2 folgen zunächst die Grundlagen mit den Grundbegriffen nach Quelle 2 und den mathematischen Zusammenhängen nach Quelle 1. In Abschnitt 3 haben wir eine Einführung in Systemfunktionen, Zuverlässigkeitsschaltbilder und –funktionen nach Quelle 1. Abschließend folgt in Abschnitt 4 die Redundanz.

2. Grundlagen

2.1 Grundbegriffe

Die *Fehlervermeidung* behandelt den Umgang mit Fehlern zur Entwicklungszeit. Methoden dazu sind Software Engineering, Reviews, Testen, Lokalisierung oder Analyse.

Der Begriff *Fehlertoleranz* aus dem Titel selbst bedeutet, dass ein System bei begrenzter Anzahl fehlerhafter Subsysteme seine Funktion erfüllen kann. Als Beispiel haben wir oben schon das Flugzeug in dem z.B. auch der Ausfall des Stromkreises für ein Triebwerk undramatisch ist, da es mindestens einen weiteren unabhängigen Stromkreis gibt. Diese Arbeit wird sich eher mit Fehlertoleranz als mit Fehlervermeidung beschäftigen.

Das Wort *Fehler* bedeutet schwammig ausgedrückt die Abweichung vom Richtigen. Im englischen gibt es für Fehler drei Begriffe, die aber Unterschiede in der Bedeutung haben:

> *Fault* ist die Fehlerursache oder der Fehlergrund. Das kann z.B. ein fehlerhafter Algorithmus sein.
> *Error* ist die Folge eines Fault, also ein Fehlerzustand des Systems. Dies könnten falsche Variablenwerte sein.
> *Failure* ist das Systemversagen was von einem Error hervorgerufen wird und schlimme Folgen haben kann.

Insgesamt ergibt sich folgende kausale Kette:

Fehlertoleranz versucht die Kette zu unterbrechen.

Der zweite Begriff aus dem Titel dieser Arbeit ist die *Zuverlässigkeit*, die die Eigenschaft eines Objekts ist in einem Zeitraum seine geforderte Funktion zu erfüllen. Wenn man also angibt, dass in 10 Jahren 5% der Autos einer Automarke trotz korrekter Wartung im normalen Betrieb einfach komplett ausfallen und eine Reparatur in der Werkstatt benötigen, so ist das eine Angabe zur Zuverlässigkeit.

Bereits bei diesem Begriff merkt man vielleicht, dass dieses Themengebiet viel mit Wahrscheinlichkeitsrechnung zu tun hat. Welche Bauteile führen zum Ausfall und wie wahrscheinlich ist, dass diese ausfallen? Wie viele Geräte sind zu einem Zeitpunkt t noch übrig? Viele Fragen benötigen die Wahrscheinlichkeitstheorie, auf die in diesem Kapitel verstärkt eingegangen wird.

2.2 Mathematische Grundlagen

Wir wollen die Zuverlässigkeit eines Gerätes betrachten und nehmen dazu 100 identische Geräte in Betrieb. Nun beobachten wir die Zahl der Geräte die in einem Monat ausfallen und die Anzahl der insgesamt noch funktionierenden Geräte. Dabei erhalten wir folgende Tabelle mit den relativen Werten und zwei Graphen mit den Absolutwerten:

Tabelle 1: Statistik der noch funktionierenden Geräte

Monat	0	1	2	3	4	5	6	7	8	9	10	11
Ausfälle im Monat	0	0,04	0,13	0,22	0,16	0,12	0,04	0,06	0,05	0,07	0,06	0,05
Noch funktionierend	1	0,96	0,83	0,61	0,45	0,33	0,29	0,23	0,18	0,11	0,05	0

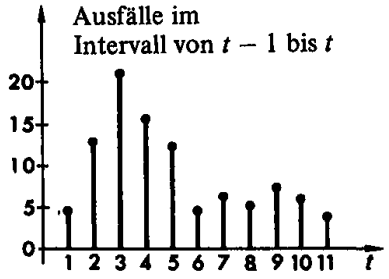


Abb. 1[1]

Abb. 2 [1]

Bei einer hinreichend großen Zahl von getesteten Geräten kann man diese Graphen auf alle Geräte verallgemeinern und somit die Wahrscheinlichkeit ablesen, dass ein zufällig gewähltes Gerät zum Zeitraum t funktioniert. Außerdem kann man den Graphen eine Kurve annähern und somit die Wahrscheinlichkeit für einen bestimmten Zeitpunkt ablesen, dass das Gerät noch funktioniert. Die zur an das rechte Bild angenäherten Kurve gehörende Funktion $v(t)$ nennt man *Überlebenswahrscheinlichkeit*. Diese Funktion hat die Eigenschaft $v(0)=1$, da zu

Beginn alle Geräte funktionstüchtig sind und $v(\infty)=0$, da nicht kaputtgehende Geräte nicht realisierbar sind.

Abb. 1 gibt uns an wie viele Geräte absolut gesehen in einem Intervall ausfallen. Uns interessiert nun die „relative Sterblichkeit“, also wie viele Geräte ausfallen in Abhängigkeit zu der Anzahl der noch funktionierenden Geräte. Die Zugehörige Formel lautet:

$$\lambda(t) = \frac{v(t-1) - v(t)}{v(t-1)}$$

$v(t-1) - v(t)$ ist die Zahl der in diesem Intervall kaputtgegangenen Geräte und *v(t–1)* die Anzahl der zuletzt noch funktionierenden Geräte. Wenn *v(t)* stetig und differenzierbar ist, dann ist

$$\lambda(t) = -\frac{v'(t)}{v(t)}$$

Diese Funktion nennt man *Ausfallrate*.

Betrachten wir nu zwei Überlebenswahrscheinlichkeiten und ihre zugehörigen Ausfallraten:

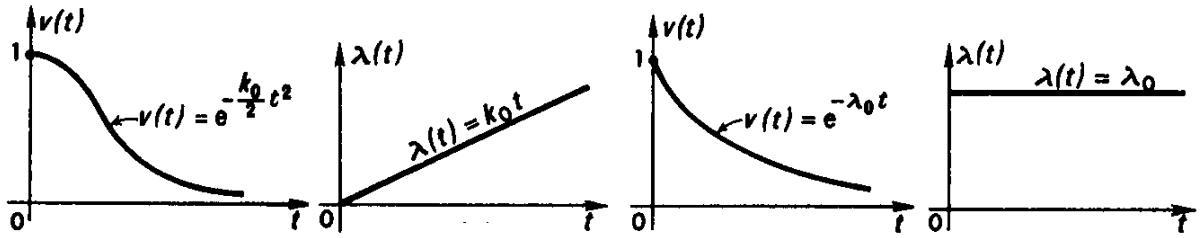


Abb. 3-6 [1] verschiedene Ausfallraten

Abb. 3 entspricht in etwa der Überlebenswahrscheinlichkeit vom ersten Beispiel zu Abb.1 und und entspricht allgemein Geräten mit Abnutzungserscheinungen. Die Ausfallrate nimmt proportional zu t zu. Abb. 4 entspricht Geräten, die zufällige Schaden erleiden. Ihre Ausfallrate ist dabei konstant.

Ein wichtiger Wert ist die *mittlere Lebensdauer* eines Gerätes mit der Überlebenswahrscheinlichkeit $v(t)$. Er entspricht dem Erwartungswert ist somit

$$\bar{t} = \sum_{t=1}^{\infty} t[v(t-1) - v(t)]$$

Ein Gerät, das bereits das Alter *a* erreicht hat, soll Alter a+t erreichen. Aus der Beziehung für bedingte Wahrscheinlichkeiten erhält man

$$v_a(t) = \frac{v(a+t)}{v(a)}$$

Angenommen wir haben mehrere Geräte gleicher Sorte, aber unterschiedlichen Alters. Den Geräten ist nicht anzusehen wie alt sie sind, man weiß aber wie viele man von welchem Alter hat. Wir wollen trotzdem wissen, wie wahrscheinlich es ist, dass ein Gerät nach einer Zeit t noch funktioniert. Dazu nehmen wir einfach von allen Altersgruppen anteilsmäßig zur Gesamtmenge die Überlebenswahrscheinlichkeit und erhalten die Formel

$$v(t) = \frac{S_1}{S} v_{a_1}(t) + \frac{S_2}{S} v_{a_2}(t) + \cdots + \frac{S_r}{S} v_{a_r}(t)$$

3. Systemfunktionen, Zuverlässigkeitsschaltbilder und -funktionen

3.1 Systemfunktionen

Wir können jetzt die Zuverlässigkeit von Geräten bestimmen und sie vergleichen. Jetzt möchten wir mehrere Geräte, von denen wir die Zuverlässigkeit vielleicht kennen, zu einem neuen Gerät kombinieren. Dazu nehmen wir uns ein Gerät e. e ordnen wir eine Zustandsvariable x die, die 1 ist, wenn x funktioniert und 0 ist, wenn x nicht funktioniert. Nehmen wir jetzt an das Gerät besteht aus zwei Komponenten e_1 und e_2 mit den beiden Zustandsvariablen x_1 und x_2. $\varphi(x_1,x_2)$ soll eine Funktion sein, die in Abhängigkeit von x_1 und x_2 angibt, ob e funktioniert.

e soll nun nur funktionieren, wenn e_1 und e_2 funktionieren. Wir suchen nun die Funktion $\varphi(x_1,x_2)$, die nur 1 ist, wenn x_1 und x_2 1 sind und sonst 0 ist. Dazu setzen wir φ einfach als

$$\varphi(x1, x2) = x_1 x_2$$

So eine Funktion nennt man *Systemfunktion*. Sie gibt den Zustand des Geräts in Abhängigkeit der Komponenten an.

Nun soll e genau dann funktionieren, wenn e_1 oder e_2 funktionieren, also suchen wir eine Funktion $\varphi(x_1,x_2)$, die dann 1 ist, wenn x_1 oder x_2 1 ist. Analog zum Satz der Inklusion und Exklusion auf Mengen erhält man

$$\varphi(x_1, x_2) = 1 - (1 - x_1)(1 - x_2) = x_1 + x_2 - x_1 x_2$$

Allgemein sei e ein Gerät aus n Komponenten e_1 ... e_n. Eine Systemfunktion von e ist eine Funktion $\varphi(x_1,\ldots,x_n)$, die für alle möglichen binären x_i den Wert 0 oder 1 annimmt.

Die Funktion $\varphi(x_1,x_2,x_3) = x_1x_2 + x_1x_3 - x_1x_2x_3$ ist eine Systemfunktion wie man durch Anlegen einer Wertetabelle schnell überprüfen kann. Die Funktion $\varphi(x_1,x_2,x_3) = x_1 + x_3 - x_1x_2x_3$ ist aber keine Systemfunktion, da z.B. $\varphi(1,0,1) = 2$ ist.

Den Zustand der Komponenten von e kann man durch einen Vektor darstellen, der aus Nullen und Einsen besteht. Der Vektor $(x_1,x_2,x_3,x_4) = (1,0,0,1)$ gibt z.B. an, dass e_1 und e_4 funktionieren und e_2 und e_3 nicht.

Seien $(x'_1,...,x'_n)$ und $(x_1,...,x_n)$ zwei Vektoren eines Geräts e. Der erste Vektor majorisiert den zweiten wenn für alle i gilt: $x'_i \geq x_i$.

Beispiele: $(1,0,1)$ majorisiert $(1,0,0)$
$(1,1)$ majorisiert $(0,0)$, aber
$(1,0)$ und $(0,1)$ sind nicht vergleichbar

Sei φ eine Systemfunktion. Wenn für alle Vektorpaare bei denen $(x'_1,...,x'_n)$ den Vektor $(x_1,...,x_n)$ majorisiert, gilt

$\varphi\,(x'_1,...,x'_n) \geq \varphi\,(x_1,...,x_n)$, dann ist φ eine *monotone Systemfunktion*.

Dieser Satz bedeutet nichts anderes, als dass ein Gerät, das funktioniert, durch zusätzliche funktionierenden Komponenten auch weiterhin funktioniert. Ein Beispiel für keine monotone Systemfunktion wäre $\varphi(x_1,x_2) = 1 - x_1x_2$. Das Gerät funktioniert immer, außer wenn beide Komponenten funktionieren, was nicht der Funktionsweise von Geräten in der Realität entspricht. Geräte mit ausfallenden Komponenten haben deshalb eine monotone Systemfunktion.

3.2 Zuverlässigkeitsschaltbilder

Wir betrachten nun einen elektrischen Dipol mit einem Eingang 0 und einem Ausgang Z. Mit dessen Hilfe haben wir nun eine anschauliche Darstellung der monotonen Systemfunktion $\varphi(x_1,x_2) = x_1x_2$:

Abb.7 [1]

Der Strom fließt nur von 0 nach Z, wenn e_1 und e_2 funktionstüchtig sind. Diese Art von Darstellung als elektrisches Schaltbild nennt man *Zuverlässigkeitsschaltbild*.

Nun ein Zuverlässigkeitsschaltbild von $\varphi(x_1,x_2) = 1 - (1 - x_1)(1 - x_2)$, also wenn e_1 oder e_2 funktioniert.

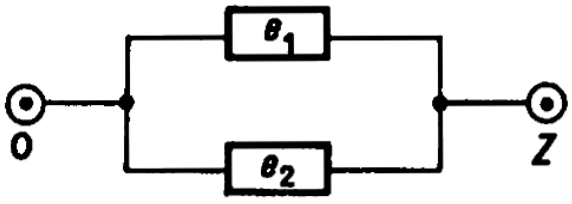
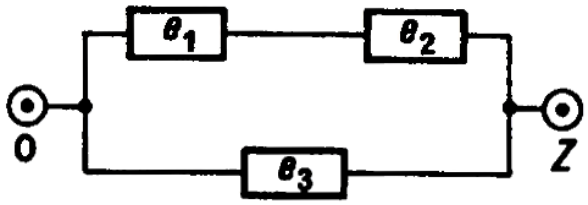


Abb. 8 [1]

Als drittes Beispiel eine Kombination aus beiden vorherigen, die Systemfunktion $\varphi(x_1,x_2,x_3) = 1 - (1 - x_1x_2)(1 - x_3)$. e funktioniert hier nur, wenn e_3 oder e_1 und e_2 funktionieren:



Abb, 9 [1]

Anhand Zuverlässigkeitsschaltbildern hat man auch eine anschauliche Darstellung für die nächsten beiden Begriffe, den Weg und den Schnitt einer Systemfunktion.

Für die Menge $E=\{e_1,...,e_n\}$ der Komponenten ist eine Teilmenge $E'=\{e_{i1},...,e_{in}\}$ ein *Weg*, wenn $x_{i1} = ... = x_{in} = 1$ hinreichend dafür ist, dass das Gerät funktioniert. Analog dazu ist E' ein *Schnitt*, wenn $x_{i1} = ... = x_{in} = 0$ hinreichend dafür ist, dass das Gerät nicht funktioniert.

E' ist ein *minimaler Weg*, wenn keine Teilmenge von E' einen Weg liefert und E' ist ein *minimaler Schnitt*, wenn keine Teilmenge von E' einen Schnitt liefert.

Im letzen Zuverlässigkeitsschaltbild ist $\{e_1,e_2,e_3\}$ sowohl Weg als auch Schnitt, aber kein minimaler. Ein minimaler Weg sind $\{e_1,e_2\}$, sowie $\{e_3\}$ und ein minimaler Schnitt $\{e_1,e_3\}$ sowie $\{e_2,e_3\}$.

Von einer Systemfunktion kommt man relativ schnell auf ein Zuverlässigkeitsschaltbild indem man alle minimalen Wege sucht, im Schaltbild die Komponenten eines minimalen Wegs in Reihe schaltet und dann alle diese Reihenschaltungen parallel schaltet. Durch eine Wertetabelle erfährt man, dass die Wege von $\varphi(x_1,x_2,x_3) = x_1x_3 + x_2x_3 - x_1x_2x_3$ $\{e_1,e_3\}$, $\{e_2,e_3\}$, sowie $\{e_1,e_2,e_3\}$ sind. Die minimalen Wege sind $\{e_1,e_3\}$ und $\{e_2,e_3\}$, also ist das zugehörige Schaltbild

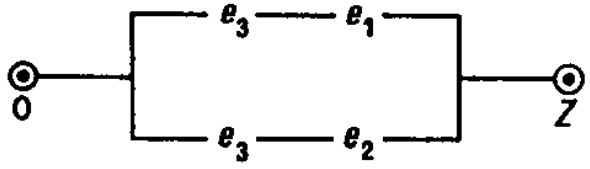


Abb. 10 [1]

Ähnlich kann man auch vorgehen um die Systemfunktion eines Zuverlässigkeitsschaltbildes zu erhalten. Dazu sucht man wieder die minimalen Wege (diesmal im Schaltbild) und zeichnet das vereinfachte Schaltbild. Da es sich nur um und- und oder-Verknüpfungen handelt, muss man diese nur noch zur Systemfunktion kombinieren. Zum oberen Bild erhält man die Funktion

$$\varphi(x_1,x_2,x_3) = 1 - (1 - x_1x_3)(1 - x_2x_3) = x_1x_3 + x_2x_3 - x_1x_2x_3^2$$

Die Exponenten kann man weglassen, da man sich das als Reihenschaltung der Komponente mit sich selbst vorstellen kann, was am Ergebnis aber nichts ändert. Wenn x_3 bei einem Durchlauf in der Reihe funktioniert, wird die Komponente auch beim nächsten Durchlauf funktionieren. Also erhält man:

$$\varphi(x_1,x_2,x_3) = x_1x_3 + x_2x_3 - x_1x_2x_3$$

3.3 Zuverlässigkeitsfunktion

Die ganze Zeit wollen wir nur wissen welche Überlebenswahrscheinlichkeit ein Gerät aus mehreren Komponenten mit jeweils bekannter Überlebenswahrscheinlichkeit hat. Dazu betrachten wir erstmal die Systemfunktion $\varphi(x_1,x_2) = x_1x_2$.

Die Wahrscheinlichkeit, dass e funktioniert bei einer Wahrscheinlichkeit von p_1 für ein funktionierendes e_1 und p_2 für ein funktionierendes e_2 ist einfacherweise.

$$h(p_1, p_2) = p_1p_2$$

So eine Funktion heißt *Zuverlässigkeitsfunktion*.

Ähnlich ist es bei der Systemfunktion $\varphi(x_1,x_2) = 1 - (1 - x_1)(1 - x_2) = x_1 + x_2 - x_1x_2$. Hier ist die Zuverlässigkeitsfunktion

$$h(p_1, p_2) = p_1 + p_2 - p_1p_2$$

Etwas komplizierter ist es bei der Systemfunktion

$$\varphi(x_1,x_2,x_3) = 1 - (1 - x_1x_3)\ (1 - x_2x_3) = x_1x_3 + x_2x_3 - x_1x_2x_3^2$$

$$\text{vereinfacht:}\quad \varphi(x_1,x_2,x_3) = x_1x_3 + x_2x_3 - x_1x_2x_3$$

Bei binären Zahlen macht es keinen Unterschied ob ich x_3^2 oder x_3 nehme, aber bei reellen schon. Hier nimmt man die gleiche Begründung wie bei 2.2. Man stelle sich das als Reihenschaltung der Komponente mit sich selbst vor. Wenn die Komponente mit Wahrscheinlichkeit p funktioniert, dann funktioniert sie beim nächsten Durchlauf mit Wahrscheinlichkeit 1. Also nimmt man beim einsetzen der Wahrscheinlichkeiten immer die ausmultiplizierte und von den Exponenten befreite Version hier:

$$h(p_1, p_2, p_3) = p_1p_3 + p_2p_3 - p_1p_2p_3$$

Als Beispiel möchten wir jetzt wissen, wie die Wahrscheinlichkeit ist, dass e nach 6 Monaten funktioniert, wenn die Wahrscheinlichkeiten dafür, dass e_1, e_2 und e_3 nach 6 Monaten noch funktionieren, p_1=0,5 p_2=0,2 und p_3=0,8 sind.

$$h(0{,}5; 02; 0{,}8) = 0{,}5 * 0{,}8 + 0{,}2 * 0{,}8 - 0{,}5 * 0{,}2 * 0{,}8 = 0{,}48$$

Mit Aufstellen der Systemfunktion und Kenntnis der Zuverlässigkeit kommt man relativ schnell auf die Zuverlässigkeitsfunktion eines Gerätes. Eigentlich ist die Zuverlässigkeitsfunktion nur ein anderes Wort für die Überlebenswahrscheinlichkeit,

da beide die Wahrscheinlichkeit eines Gerätes angeben und die p_i auch nur von t abhängen. Allerdings gibt man hier meistens nicht p_i in Abhängigkeit von t, sondern ein festes p_i an.

Eine interessante Frage bei der Komponentenwahl ist immer, ob sich ein Aufpreis im Vergleich zur Kostenerhöhung lohnt. Wir haben das Beispiel

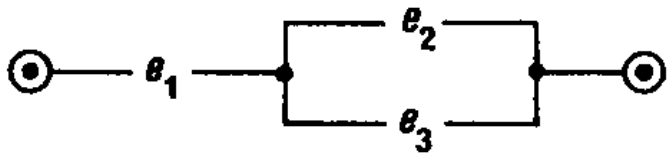


Abb. 11 [1]

mit der Systemfunktion $h(p_1,p_2,p_3) = p_1p_3 + p_1p_3 - p_1p_2p_3$

Mit e_2 und e_3 sollen die Zuverlässigkeit von 0,8 und einen Preis von 100€ haben. Bei e_1 haben wir die Wahl zwischen einer Komponente für 50€ und Zuverlässigkeit 0,9 und einer Komponente für 71€ und der Zuverlässigkeit 0,99. Im ersten Fall beträgt die Zuverlässigkeit 0,864 bei einem Preis von 150€ und im zweiten Fall 0,950 bei 171€. Die Zuverlässigkeit ist damit um 10% gestiegen für eine Preissteigerung von 14%. Ob sich dieser Mehrpreis lohnt, hängt natürlich vom jeweiligen Anwendungszweck ab.

4 Redundanz

4.1 Redundanzarten

Bei der Konstruktion neuer Geräte sollte immer darauf geachtet werden, dass das System möglichst wenig Komponenten mit geringer Zuverlässigkeit hat, die einen *Single Point of Failure* sind, also zu einem sofortigen Ausfall führen. Sollten diese Komponenten nicht ersetzbar sein, sollte man hier *Redundanz* einsetzen, also das mehrfache Vorhandensein der gleichen Komponente, die alle gemeinsam ausfallen müssen, damit das Gerät funktionsuntüchtig wird. Folgende Berechnung erfolgt nach Quelle 1.

Wie erhöht man die Zuverlässigkeit des gesamten Systems auf schnelle Art und Weise? Die einfachste Methode ist es zwei oder mehr dieser Geräte parallel zu nutzen, so dass bei Ausfall von einem Gerät die Anderen dessen Funktion übernehmen können. Zur Berechnung der Zuverlässigkeitsfunktion dieses großen Gerätes können wir die einzelnen Geräte als Komponenten mit Zuverlässigkeit $p_1=...=p_n=p$ eines größeren Gerätes betrachten. Wir erhalten

$$h(p_1, \dots, p_n) = 1 - (1 - p_1) * \dots * (1 - p_n) = \ 1 - (1 - p)^n$$

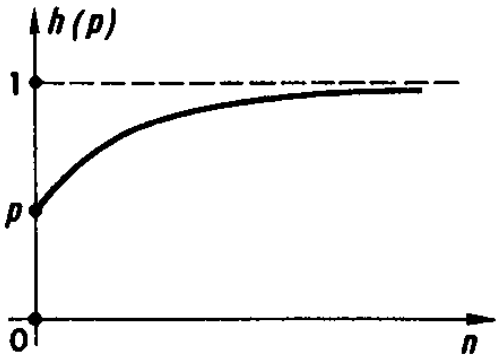


Abb. 12 [1]: Zuverlässigkeitszunahme bei Redundanz

Die Zuverlässigkeit nimmt wie im Graphen dargestellt zu.

Man kann zeigen, dass es effektiver ist anstatt der Geräte dessen Komponenten zu vervielfachen. Man kann sich das bei einem kleinen Beispiel sehen:

Ein Gerät besteht aus sehr vielen Komponenten mit gleicher Überlebenswahrscheinlichkeit und geht genau dann kaputt, wenn mindestens zwei Komponenten ausfallen. Wenn wir nur die Geräte verdoppeln, müssen in beiden Geräten jeweils zwei Komponenten ausfallen. Wenn wir die Komponenten selbst verdoppeln, müssen diese immer paarweise ausfallen, damit diese Komponente als ausgefallen gilt. In diesem Beispiel müssen also zwei Paare von Komponenten ausfallen, was bei der bereits erwähnten sehr großen Anzahl von Komponenten für die ersten Ausfälle eher unwahrscheinlich ist.

Wenn man weiß, dass ein größerer Teil nur von einer Komponente ausgeht, kann man sie entweder durch eine zuverlässigere ersetzen oder aber diese mehrfach einbauen, so dass im Fehlerfall eine gleichartige Komponente trotzdem noch zur Verfügung steht

Diese Art von Redundanz heißt *strukturelle Redundanz*. Es sind zusätzliche Objekte vorhanden, so dass sie den Ausfall einer anderen ausgleichen können. Es muss sich dabei nicht einmal um die Gleiche handeln. Eine strukturelle Redundanz liegt z.B. Flugzeug vor, wenn ein Triebwerk von mehreren unabhängigen Stromkreisen versorgt wird.

Eine Art „Abstimmen“ über das Ergebnis ist auch eine strukturelle Redundanz. Dabei nimmt man z.B. drei verschiedene Komponenten, die ein Ergebnis liefern.

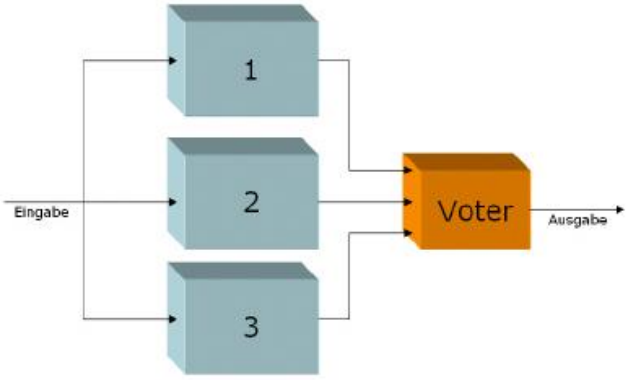


Abbildung 13 [3]: Zuverlässigkeitssteigerung durch Abstimmen

Wenn mindestens zwei davon das gleiche liefern, wird dieses genommen, da die Wahrscheinlichkeit, dass zwei verschiedene das gleiche falsche liefern, gering ist. Sollten alle drei verschiedene Ergebnisse liefern, gilt das gesamte Bauteil aus diesen drei Komponenten als Defekt. Auf diese Art erhöht man auch die Zuverlässigkeit, da eine weitere möglichst zuverlässige Komponente die anderen Komponenten untereinander überprüft.

Strukturelle Redundanz bezieht sich nicht nur auf Geräte, sondern auch auf Daten. Wenn Daten zweimal vorhanden sind, gilt das auch als strukturelle Redundanz.

Dieser und folgende Redundanzeinteilungen in 4.1 und 4.2 erfolgen nach Quelle 2.

Nicht nur auf Hardwareebene gibt es Redundanz, sondern auch auf der Ebene der Software. Man kann z.B. mehrere Betriebssysteme oder mehrere Implementierungen eines Algorithmus einsetzen. Sollte z.B. eine Ausgabe nicht stimmen können, so kann auf ein anderes System oder einen anderen Algorithmus umgestiegen werden. Angenommen es soll eine Liste sortiert werden. Ein falsch implementierter Quicksort-Algorithmus kommt allerdings zu Erkenntnis, dass die 5 vor der 3 kommt. Daraufhin kann z.B. auf einen Heapsort-Algorithmus zurückgegriffen werden, der ebenfalls implementiert wurde. Hier sind wir wieder auf eine andere Form der Redundanz gestoßen, die *Zeitredundanz*. Die Redundanz hält sich nicht zeitlich neutral im Hintergrund, sondern erfordert weitere Zeit im Fehlerfall. Diese Redundanz kann sinnvoll sein, wenn man nicht weiß, dass ein Fehler auftritt, ihn aber ausbügeln möchte, falls er doch unerwarteter weise aufgetreten ist (weil er z.B. beim Testen der Software übersehen wurde). Eine Überarbeitung des Codes ist natürlich anschließend trotzdem empfehlenswert. Die „Abstimmungs-Redundanz“ aus 3.2 kann auch bei Software eingesetzt werrden. Auf modernen Rechnern kann diese auch immer einfacher eingesetzt werden, da durch Zunahme von Prozessorgeschwindigkeit und –anzahl die parallele Abarbeitung mehrerer Algorithmen immer schneller läuft.

Eine weitere Form der Redundanz ist aus der Datenübertragung und -sicherung bekannt. Daten werden so modifiziert, so das die Überprüfung auf Fehler oder sogar Korrektur von Fehlern ermöglicht wird, während der ursprüngliche Sinn erhalten

bleibt. Dies ermöglicht auch das Versenden unter schwierigen Umständen, wie der Kommunikation einer Raumfähre mit der Erde. Dies wird hauptsächlich erreicht durch die *Informationsredundanz*.

Den Daten kann ein Bit hinzugefügt werden, der zur Überprüfung dient wie die letzte Ziffer im Personalausweis. In diesem wird dazu die erste Ziffer mit 7 multipliziert, die zweite mit 3, die dritte mit 1, die vierte wieder mit 7 usw. Nun werden von allen so berechneten Zahlen die Einerstellen genommen und aufaddiert. Die Einerstelle dieser Summe ist die letzte Ziffer und somit die Prüfziffer im Personalausweis. Bei zufälliger Manipulation der Personalausweisnummer ist die Wahrscheinlichkeit 10%, dass die Prüfziffer weiterhin korrekt ist. Bei einer Erweiterung auf zwei Prüfziffern liegt diese Wahrscheinlichkeit nur noch bei 1%.

Andere Fehlererkennungsalgorithmen sind z.B. CRC und FEC, die bei der Datenübertragung im Internet und im Netzwerk eingesetzt werden. Diese wurden in TI 3 erläutert (siehe TI 3 Skript WS08/09).

Als letze Redundanzform gibt es die *funktionale Redundanz*, diese fügt zum System weitere Funktionen hinzu, ohne denen das System auch lauffähig wäre. Diese können z.B. Testfunktionen sein.

4.2 Dynamische und Statische Redundanz

Bisher waren bei den Beispielen die redundanten Elemente immer aktiv und konnten nichts anderes als die im Notfall zu ersetzenden Komponenten tun. Diese Art von Redundanz heißt *statische Redundanz*. Der Ersatzstromkreis des Flugzeugtriebwerks arbeitet zwar, bringt aber selbst keine Vorteile außerhalb der Zuverlässigkeit. Ebenso bringt ein Ersatzrad im Auto keine Vorteile solange die andern vier Reifen noch in Ordnung sind.

Eine andere Form von Redundanz ist die *dynamische Redundanz*. Es sind noch immer Komponenten vorhanden, die sich gegenseitig ersetzen können. Allerdings arbeiten diese Komponenten unabhängig voneinander an verschiedenen Aufgaben. Als Beispiel kann ein Mehrprozessorsystem genannt werden in dem ein Prozessor auch ausfallen darf solange mindestens einer funktionstüchtig bleibt. Die Prozessoren sind zueinander redundant und jeder kann die Aufgaben des anderen übernehmen. Allerdings liegt die zusätzliche Leistung nicht brach, sondern wird ausgiebig genutzt, was das System beschleunigt. Im Fehlerfall kann trotzdem ein einzelner Prozessor die Aufgaben der anderen übernehmen und alleine weiterarbeiten.

5 Fazit

Fehlertoleranz ist ein wichtiger Aspekt der Technik. Viele Systeme arbeiten in Bereichen in denen ein Fehler große finanzielle Schäden oder sogar zu Verlust von Menschenleben führen könnte. Deshalb sollte in diesen Bereichen niemals bei der Zuverlässigkeit gespart werden.

Fehlertoleranz sollte lieber immer erweitert, anstatt verändert oder abgebaut werden, schließlich sind die Hersteller eines Gerätes oder einer Software auch für dessen Fehler und dessen Folgen verantwortlich. Auch bei normalen Geräten sollte eine ausreichende Zuverlässigkeit gegeben sein, da dies ärgerlich ist und man auch beachten sollte, dass man sich dadurch ein negatives Image aufbaut und die enttäuschten Kunden die eigene Ware in Zukunft meiden.

Literaturverzeichnis:

[1] Zuverlässigkeit in der Technik, Arnold Kaufmann
R. Oldenburg Verlag München Wien 1970

[2] Fehlertolerante Rechensysteme, Prof. Dr. Winfried Görke
R. Oldenburg Verlag München Wien 1989

[3] Fehlertoleranz, Jürgen Ruf und Thomas Kropf
http://www-ti.informatik.uni-tuebingen.de/~ruf/seminar0102/Fehlertoleranz.pdf

[4] Deutscher Personalausweis
http://www.pruefziffernberechnung.de/P/Personalausweis-DE.shtml

[5] Sicherheit: Therac-25, Lutz Prechelt
http://www.inf.fu-berlin.de/inst/ag-se/teaching/V-AWS-2008/22_Sicherheit.pdf

[6] Tod durch Handy-Explosion, Markus C. Schulte von Drach
http://www.sueddeutsche.de/wissen/194/425951/text/