A Theoretische Grundlagen

A 1 Einführung

Jede Kultur besitzt zur Verständigung ihrer Menschen und zur Dokumentation ein Informationssystem, das Rechnen und Schreiben ermöglicht. Deshalb sind dort überall Symbole für Rechen- und Textzeichen zu finden und Vorschriften zu ihrer sinnvollen Kombination bzw. Verarbeitung (Rechenregeln und Wortzusammenhänge). Diese Informationen mit Hilfe von Rechenautomaten zu speichern, zu verarbeiten und weiterzugeben, ist Gegenstand der Informatik (engl.: Computer Science). Bild A-1 zeigt die Bereiche der Informatik, wie sie auch im vorliegenden Buch behandelt werden.

Zunächst werden die Grundlagen der *Informationstheorie* behandelt. Hier werden die entscheidenden *Begriffe* definiert, Maßstäbe zur *Messung* von Information vorgestellt, die *Kodierungsmethoden* behandelt und Fragen der *Ergonomie* bei Datenverarbeitungsanlagen beantwortet.

Um dem Zweck der Informatik, Informationen mit Hilfe von Rechnern zu verarbeiten, gerecht werden zu können, müssen *einerseits* die Geräte *(Hardware)* dafür bereitgestellt werden und andererseits die zu bearbeitenden Probleme über *Softwaresysteme* (Betriebs- und Datenbanksysteme) automatengerecht aufbereitet werden. Ein wichtiges Gebiet ist dabei die Lehre von den Methoden und Werkzeugen zur qualitäts- und kostenoptimalen Software-Erstellung *(Software-Engineering)*. Um Informationen über ganz spezielle Bereiche zu erhalten (z. B. über Märkte durch Markt-Informationssysteme), werden *Informationssysteme* entworfen. Die *betrieblichen Informationssysteme* bestehen bei der *rechnerintegrierten Fabrikation* (CIM: Computer Integrated Manufacturing) aus den Bausteinen Produktionsplanungs- und -steuerungssystemen (PPS) und der Betriebsdatenerfassung (BDE). Zu den *kommerziellen Informationssystemen* zählen das Rechnungswesen, die Kostenrechnung und spezielle Auswertungen betriebswirtschaftlich wichtiger Daten. Damit viele Bereiche miteinander kommunizieren können (z. B. die Konstruktion mit der Fertigung oder der Lieferant mit dem Kunden),

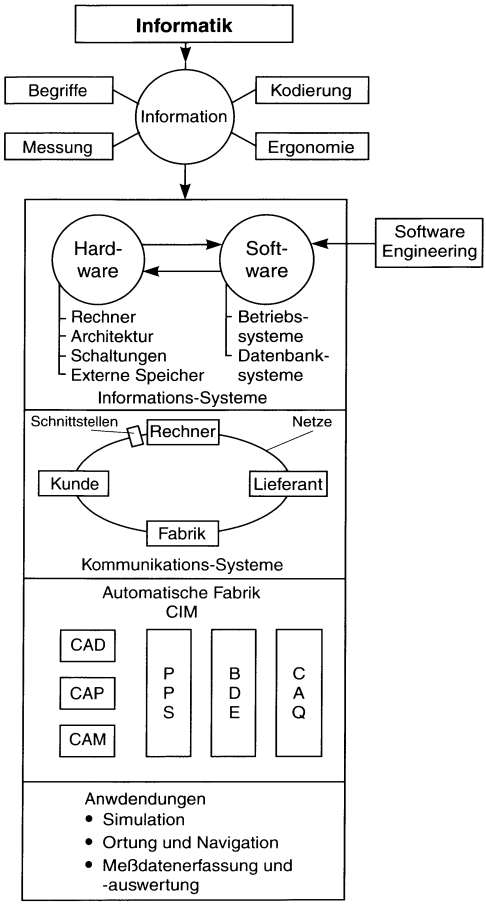


Bild A-1. Bereiche der Informatik.

müssen *Kommunkationssysteme* aufgebaut werden. Hierbei sind besonders die *Schnittstellen* zwischen den einzelnen Kommunikationseinheiten und die Verbindung untereinander durch *Bussysteme* und *Netze* von besonderer Bedeutung. Besonders wichtig für den Ingenieur sind im einzelnen die rechnergestützten Bausteine der Konstruktion (*CAD:* Computer Aided Design), der Fertigungsplanung (*CAP:* Computer Aided Planning), der Fertigung selbst (*CAM:* Computer Aided Manufacturing) und dem rechnergestützten Qualitätsmanagement (*CAQ:* Computer Aided Quality Management). Weitere ingenieurspezifische Anwendungsfelder sind die *Simulation von Fer-*

E. Hering et al., *Handbuch der praktischen und technischen Informatik*

tigungsprozessen, die *Ortung und Navigation* als Beispiele für sicherheitskritische Software sowie die *Meßdatenerfassung und -auswertung.*

A 2 Informationstheorie

A 2.1 Grundbegriffe

Die beiden zentralen Begriffe sind *Nachricht* und *Information.* Der Unterschied liegt, wie Bild A-2 zeigt, darin, daß eine Nachricht erst durch *Interpretationsregeln* zu einer Information wird. Das bedeutet, die Nachricht wird *verstanden.* Wenn man beispielsweise die japanische Sprache nicht beherrscht, dann können japanische Nachrichten nicht als Information verstanden werden. Diese Interpretationsregeln haben eine ganz wichtige Bedeutung: Gleiche Nachrichten führen bei unterschiedlichen Interpretationsregeln zu *verschiedenen Informationen.* Deshalb ist es wichtig, daß für Nachrichten genau vereinbarte Interpretationsregeln gelten.

Eine Information besitzt, wie Bild A-2 weiter zeigt, unterschiedliche Aspekte: Sie kann mit *Informationsmessungen* statistisch ermittelt werden, sie kann unterschiedliche *Bedeutungen* besitzen *(Semantik)* und ihre *Wirkung* (Pragmatik) kann verschieden sein. Im folgenden wird in der *Informationstheorie* ausschließlich der statistische Gehalt von Informationen behandelt.

In Bild A-3 ist die *Informationsverarbeitung* als *Systemkette* dargestellt. Die Informationen gehen von einer *Informationsquelle* aus, die *Signale* aussendet. Solche Signale sind meßbare *physikalische Größen,* beispielsweise Strom, Spannung oder Druck. Diese Signale werden *gewandelt,* d. h. kodiert und anschließend *übertragen.* Auf dieser Strecke kann eine *äußere Störung* auftreten, welche die Signalverarbeitung verfälschen kann. Nach der Übertragung wird das Signal *rückgewandelt (dekodiert)* und steht als Information zur weiteren Auswertung zur Verfügung *(Informationssenke).*

Die Informationstheorie bezieht sich demnach insbesondere auf folgende drei Bereiche:

1. Messung von Information

Damit wird es möglich, die *Größe von Speichern* zu berechnen und entsprechend des Einsatzzweckes zu bauen.

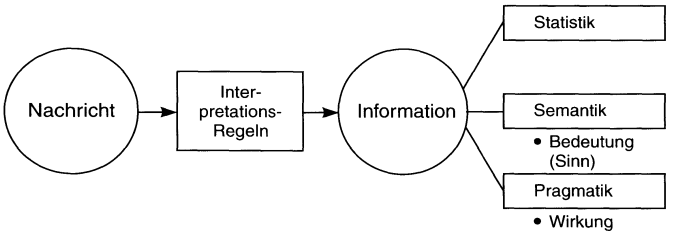


Bild A-2. Nachricht und Information.

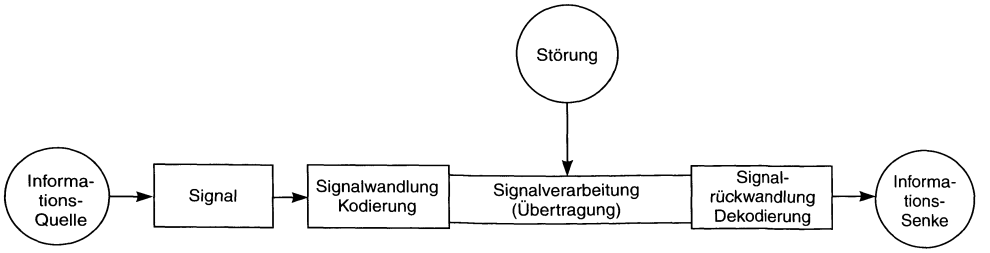


Bild A-3. Informationsverarbeitung als Prozeßkette.

2. Optimierung des Prozesses

Das informationsverarbeitende System nach Bild A-3 muß so ausgelegt werden, daß möglichst *viel Information* von der Quelle zur Senke übertragen wird. Eine wichtige Rolle spielt dabei die *optimale Kodierung* (Abschn. A 4.4.3.3).

3. Obere Grenze der Informationsverarbeitung

Dabei wird bestimmt, welche Informationsmenge in welcher Zeit höchstens übertragen werden kann. Ein Maß dafür ist die *Kanalkapazität.* Sie gibt die *Leistungsfähigkeit* von Systemen der Informationsverarbeitung an.

Die oben erwähnten Punkte werden besonders wichtig, wenn verschiedene Systeme aneinander *gekoppelt* werden, wie dies bei *Kommunikationssystemen* der Fall ist (Abschn. F). Die Systeme müssen über ihre *Schnittstellen* (Abschn. F 2) so aneinander angepaßt werden, daß die Weitergabe der Informationen *ohne Verzögerung* und *Verfälschung* geschehen kann.

A 2.2 Messung des Informationsgehaltes

Wie bereits erwähnt, werden nur die mathematischen Grundlagen der Informationstheorie nach SHANNON (C.E. SHANNON, geb. 1916) behandelt. Das Wesen der Information besteht darin, *Unsicherheit zu beseitigen.* Denn, bevor eine bestimmte Information eintrifft, sind viele Möglichkeiten wahrscheinlich. Doch nach dem Eintreffen der Information ist die Unsicherheit bezüglich des Eintreffens dieser Information beseitigt. Allgemein kann deshalb gesagt werden:

Information ist beseitigte Unsicherheit.

Als Maß für die Information ist die *Zunahme der Wahrscheinlichkeit* für die richtige Voraussage eines Ereignisses. An einem Beispiel soll dies erläutert werden: Es wird angenommen, daß es nur zwei Möglichkeiten, nämlich „Nein" (0) und „Ja" (1) gäbe. Jede Möglichkeit hat die gleiche Eintreff-Wahrscheinlichkeit, nämlich p = 0,5. Ist die Information vorhanden (z. B. „Ja"), dann wird $p_0 = 0$ und $p_1 = 1$. Das bedeutet, mit *Sicherheit* ist „Ja" eingetreten. Eine der wichtigsten Fragen in der Informationstheorie ist, *wieviele Entscheidungen* notwendig sind, um eine Information aus einer Vielzahl von Nachrichten auszuwählen. Die elementare Entscheidung besteht immer aus *zwei Möglichkeiten.* Bild A-4 zeigt die Auswahl eines Zeichens aus den acht Zeichen „a" bis „h".

Soll eine bestimmte Nachricht, beispielsweise „e" ausgewählt werden, dann wird zum Ursprung O zurückgegangen. Zuerst wird entschieden, daß die Nachricht in der „unteren" Hälfte zu finden ist. Damit kommen nur noch die Zeichen „e", „f", „g" und „h" in Frage. Als nächstes wird „oben" eingegeben, womit nur noch die Zeichen e und f zur Auswahl stehen. Mit der weiteren Eingabe von „oben" wird das gewünschte Zeichen „e" ausgewählt. Es wurde gezeigt, wie mit jeweils *zwei verschiedenen Zuständen* eine Auswahl getroffen werden konnte. Solche *Kodierungen* werden *binäre Kodierungen* genannt und meist mit den beiden Signalen *0* und *1* dargestellt (Abschn. A 4). Bild A-4 zeigt auf der rechten Seite die entsprechenden Kodierungen. Es ist zu erkennen, daß mit *drei Binärsignalen* genau *acht* Informationen ausgesucht werden können. Dies wird im *Dualsystem* erkennbar; denn $2^3 = 8$.

Die Grundfrage der Informationstheorie, mit wieviel Binärzeichen eine spezielle Information ausgewählt werden kann, läßt sich somit beantworten:

Mit *m Binärzeichen* kann aus 2^m Nachrichten eine spezielle ausgewählt werden.

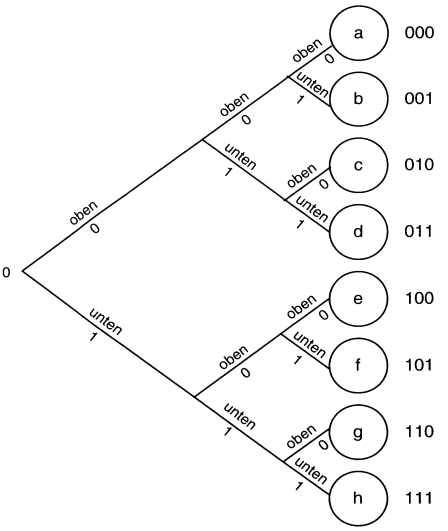


Bild A-4. Kodierung von Buchstaben mit 0 und 1 (Dualsystem).

Dies zeigt der Aufbau des *Dualsystems* in Tabelle A-1, das hier nur kurz dargestellt wird (Abschn. A 4):

Tabelle A-1. Auswahl der Informationen mit dem Dualsystem

Anzahl Binärsignale	Menge der Informationen
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$

Umgekehrt kann auch die Frage beantwortet werden, *wieviele Binärzeichen* zur eindeutigen Identifizierung einer speziellen Information aus n Informationen benötigt werden. Dazu muß der *Zweierlogarithmus* aus der Zahl der Informationen gebildet werden (Lesen der Tabelle A-1 von rechts nach links). Es gilt:

Mit ld(n) Binärzeichen kann eine spezielle Information aus n verschiedenen Informationen ausgewählt werden.

Dabei bedeutet ld den *Logarithmus dualis,* d. h. den Logarithmus zur Basis 2($\log_2 n$). Für die Umrechnung zum Zehnerlogarithmus (log) gilt:

$$ld(n) = 3,32 \log(n).$$

Es wird der *Entscheidungsgehalt* H_0 aus einer Anzahl von n Möglichkeiten errechnet zu:

$$H_0 = ld(n). \qquad \text{(A-1)}$$

Die Einheit ist Bit (von binary digit). Dies ist die *Maßeinheit* der *Information.* Es kann gezeigt werden, daß es kein Verfahren gibt, das mit weniger Abfragen auskommt (bei gleicher Häufigkeit der Informationen).

Beispiel:

A 2-1: Wie oft muß mindestens gefragt werden, um eine Zahl zwischen 1 und 50 erraten zu können?

Lösung:
Die Anzahl der Mindestabfragen errechnet sich aus: ld(50) = 5,6. Das heißt, es sind mindestens 6 Fragen zu stellen.

Haben die jeweiligen Informationen jedoch *unterschiedliche relative Häufigkeiten* (oder Wahrscheinlichkeiten p), dann kann man mit *weniger Binärzeichen* auskommen. Dies zeigt Bild A-5.

Dabei werden die *relativ häufig* vorkommenden Zeichen („a" und „b") mit *möglichst wenig Binärzeichen* kodiert. Anders ausgedrückt kann man sagen:

Der Informationsgehalt einer *häufig* vorkommenden *Information* ist *gering* und der einer *seltenen Information groß.*

Wie Bild A-5 zeigt, muß für die Information mit der relativen Häufigkeit p_i der Informationsgehalt von $ld(1/p_i)$ übertragen werden. So genügt für das Zeichen „a" ein Binärzeichen. Für das Zeichen „d" sind wegen der Seltenheit bereits drei Binärzeichen erforderlich.

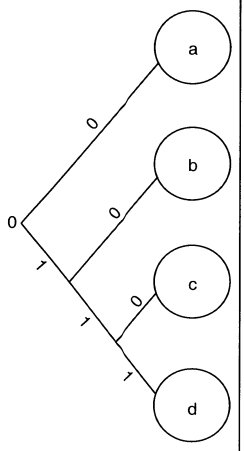
Treten n Nachrichten mit den einzelnen Wahrscheinlichkeiten p_i auf, dann errechnet sich die *gesamte Unbestimmtheit* der Zustände, d. h. der *mittlere Informationsgehalt* oder die *Entropie H* zu:

$$H = \sum p_i ld(1/p_i) = -\sum p_i ld(p_i). \qquad \text{(A-2)}$$

Diese Gleichung bestätigt, daß der *Informationsgehalt* eines Ereignisses umso*größer* ist, je *seltener* dieses vorkommt. Diese Gleichung ist auch in der Thermodynamik zu finden. Die *thermodynamische Wahrscheinlichkeit,* bestimmte Moleküle in bestimmten Teilvolumina zu finden, wird *Entropie* genannt und mit obiger Formel berechnet. Deshalb wird auch hier der Begriff der *Entropie* verwendet.

Wie Bild A-5 zeigt, liegt der mittlere Informationsgehalt bei $H = 13/4$ Bit. Wären alle Informationen gleich verteilt (p_i jeweils 1/4), dann würden $H = 2$ Bit benötigt. Daraus ist zu erkennen, daß es sinnvoll sein kann, Kodes zu *optimieren* (Abschn. A 4.4.3.3), wie dies beim *Morse-Alphabet* geschieht. Der häufig vorkommende Buchstabe „e" ist ein Punkt (.), während der relativ selten vorkommende Buchstabe „y" aus 4 Zeichen besteht (-.- -).

Auf die Frage nach der *maximalen Entropie* H_{max} läßt sich folgende wichtige Antwort finden:



relative Häufigkeit p_i	Kode	Anzahl Binär-zeichen	$\mathrm{ld}\left(\frac{1}{p_i}\right)$	Informationsgehalt $H = p_i \mathrm{ld}\left(\frac{1}{p_i}\right)$
$\frac{1}{2}$	0	1	$\mathrm{ld}\left(\frac{1}{1/2}\right) = 1$	$\frac{1}{2} \cdot 1 = \frac{1}{2}$
$\frac{1}{4}$	10	2	$\mathrm{ld}(4) = 2$	$\frac{1}{4} \cdot 2 = \frac{1}{2}$
$\frac{1}{8}$	110	3	$\mathrm{ld}(8) = 3$	$\frac{1}{8} \cdot 3 = \frac{3}{8}$
$\frac{1}{8}$	111	3	$\mathrm{ld}(8) = 3$	$\frac{1}{8} \cdot 3 = \frac{3}{8}$
$\Sigma p_i = 1$				$\Sigma p_i \mathrm{ld}\left(\frac{1}{p_i}\right) = 1\frac{3}{4}$

Bild A-5. Informationsgehalt und Häufigkeit von Zeichen.

Die *Entropie* ist dann *maximal,* wenn alle Ereignisse *gleich wahrscheinlich* sind.

Beispiel:

A 2-2: a) Mit wieviel Wägungen ist es möglich, aus 25 Kugeln die eine Kugel zu ermitteln, die schwerer ist als die anderen? b) Aus maximal wievielen Kugeln ist die schwerere noch zu ermitteln?

Lösung:
a) Aus Gl. (A-1) folgt für die Anzahl der Entscheidungen: $H = ld(25) = 4,65$ Bit. Da eine Wägung maximal drei Ergebnisse hat (leichter, schwerer, gleich schwer), ist die maximale Information einer Wägung $H_w = ld(3) = 1,59$ Bit. Die Anzahl der Wägungen errechnet sich dann aus: 4,65 Bit/1,59 Bit = 2,9, d. h. 3 Wägungen.

b) Mit 3 Wägungen und 3 Ergebnissen pro Wägung lassen sich aus maximal $3^3 = 27$ Kugeln die schwerere herausfinden.

Die deutsche Sprache enthält 30 Zeichen (26 Zeichen, 3 Umlaute und das Leerzeichen). Wären alle Zeichen gleich häufig, so entspräche dies einem mittleren Informationsgehalt von $ld(30) = 4,9$ Bit. Wegen der unterschiedlichen relativen Häufigkeit der Buchstaben (z. B. ist „e" sehr häufig und „q" selten) ergibt sich ein mittlerer Informationsgehalt von 4,11 Bit.

Tabelle A-2 zeigt die relativen Häufigkeiten der Buchstaben. In Tabelle A-3 sind die mittleren Informationsgehalte der verschiedenen Sprachen zusammengestellt.

Die Differenz zwischen dem maximalen Informationsgehalt H_{max} und dem tatsächlichen H_{real} gibt die *Redundanz* ΔH an:

$$\Delta H = H_{max} - H_{real}. \qquad \text{(A-3)}$$

Die *Redundanz* enthält somit eine Nachricht, die *keine Information* darstellt und zum eindeutigen Übertragen von Information *unnötig* oder *überflüssig* ist. Für die Kommunikation sind aber redundante Teile sehr wichtig, weil dadurch fehlerhafte Informationen noch richtig verstanden werden können, wie folgender Satz zeigt:

„Fr...t Eu h d s Le ns".

Tabelle A-2. Mittlerer Informationsgehalt von Buchstaben der deutschen Sprache

Blank	0.151490		
e	0.147004	b	0.015972
n	0.088351	z	0.014225
r	0.068577	w	0.014201
i	0.063770	f	0.013598
s	0.053881	k	0.009558
t	0.047301	v	0.007350
d	0.043854	ü	0.005799
h	0.043554	p	0.004992
a	0.043309	ä	0.004907
u	0.031877	ö	0.002547
l	0.029312	j	0.001645
c	0.026733	y	0.000173
g	0.026672	q	0.000142
m	0.021336	x	0.000129
o	0.017717		

Tabelle A-3. Mittlerer Informationsgehalt der unterschiedlichen Sprachen

Sprache	mittlerer Informationsgehalt
deutsch	4,11 Bit
russisch	4,36 Bit
englisch	4,04 Bit
französisch	3,97 Bit

Sprachen enthalten zur Sicherheit der Informationsübermittlung *viel Redundanz*. Für die deutsche Sprache errechnet sich die Redundanz zu:

$$\delta H = 4{,}9\,\text{Bit} - 4{,}11\,\text{Bit} = 0{,}79\,\text{Bit}\,.$$

Die *relative Redundanz* δh beschreibt den prozentualen Unterschied zum maximalen Informationsgehalt nach Gl. (A-4):

$$\delta h = (H_{\max} - H_{\text{real}})/H_{\max}. \qquad \text{(A-4)}$$

Sie beträgt für die deutsche Sprache:

$\delta h = (4{,}9\,\text{Bit} - 4{,}11\,\text{Bit})/4{,}9\,\text{Bit} = 0{,}16$ (oder 16 %).

Es kann auch allgemein mit $H = H_{\text{real}}$ und $H_0 = H_{\max}$ ein *Wirkungsgrad* η definiert werden:

$$\eta = H/H_0 = 1 - \delta h. \qquad \text{(A-5)}$$

Auch bei der technischen Informationsübermittlung werden *redundante Kodierungen* eingesetzt, die *Fehler erkennen* (z. B. durch ein Prüfbit) und sogar *korrigieren* können (Abschn. A 4.4.3.2, Hamming-Abstand).

A 2.3 Informationsfluß und Kanalkapazität

Unter *Informationsfluß* I versteht man die *Informationsmenge* dM, die *pro Zeiteinheit* dt übertragen wird:

$$I = dM/dt. \qquad \text{(A-6)}$$

Die Einheit ist Bit/Sekunde (Bit/s).

Die kleinste Zeiteinheit, die zum Übertragen eines Meßwertes gebraucht wird, ist die *Taktzeit* oder *Einschwingzeit* T_E. Wird ein *analoges Signal digital* mit der Abtastzeit t_A abgetastet, dann ergibt sich als *Grenzfrequenz* $f_g = 1/2t_A$. Den Zusammenhang zwischen Einschwingzeit T_E bzw. der *Tastfrequenz* f_a und der Grenzfrequenz f_g liefert das Shannonsche *Abtasttheorem*:

$$T_E = 1/2f_g \text{ oder } f_a = 2f_g. \qquad \text{(A-7)}$$

Das bedeutet, daß die Tastfrequenz f_a mindestens *doppelt* so groß sein muß wie die Signalfrequenz f_g, oder daß je *Sinusschwingung* mindestens *2 Abtastwerte* liegen müssen, wie Bild A-6 zeigt.

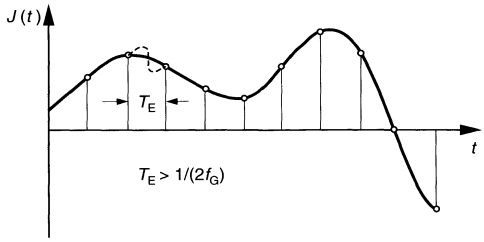


Bild A-6. Shannonsches Abtasttheorem.

Aus dem Zusammenhang nach Gl. (A-7) kann der *mittlere Informationsfluß* I aus der Entropie H und der Einschwingzeit T_E wie folgt errechnet werden:

$$I = H/T_E. \qquad \text{(A-8)}$$

Beispiel:

A 2-3: Das Telefonnetz besitzt eine Grenzfrequenz f_g von 3,4 kHz. Wie groß muß die Abtastzeit für die Umwandlung in digitale Signale mindestens sein?

Lösung:
Nach dem Shannonschen Abtasttheorem (Gl. (A-7)) gilt:

$T_E = 1/(2 \cdot 3400Hz) = 147\mu s$.

Die Tastzeit muß also mindestens $147\mu s$ betragen. In Wirklichkeit wird alle $125\mu s$ abgetastet.

Die *Kanalkapazität C* gibt den *größtmöglichen Informationsfluß* über einen Übertragungskanal an. Meist wird weniger Informationsfluß I übertragen, als möglich wäre, so daß gilt:

$$I \leq C. \qquad \text{(A-9)}$$

Eine Möglichkeit, die Kanalkapazität auszuschöpfen, besteht in der Übertragung durch einen *optimalen Kode* (Abschn. A 4.4.3.3).

A 2.4 Algorithmen

A 2.4.1 Begriff des Algorithmus

Die Bezeichnung *Algorithmus* geht auf den Araber Muhammed Al-Choresmi (oder Ibn Musa Al-Chwarismi) zurück, der um das Jahr 800 in der Stadt Khiva (heutiges Usbekistan) lebte. Er beschrieb in seinem Buch „Regeln zur Wiederherstellung und zur Reduktion" die Behandlung von Gleichungen. Damit gelang es ihm, den komplizierten Erbfall zu erklären, der sich ergab, wenn ein reicher Araber starb, der vier Frauen unterschiedlichen Standes und verschieden vielen Kindern hinterließ.

Ein Algorithmus (Bild A-7) kann folgendermaßen definiert werden:

Ein *Algorithmus* ist ein *Verfahren* zur Lösung von Problemen. Dazu wird angegeben, welche elementaren Schritte in welcher Reihenfolge zu erledigen sind.

Dieses Bild zeigt, daß Algorithmen nur für *lösbare* Probleme gefunden werden können. Zu Beginn des 20. Jahrhunderts stellte der berühmte Mathematiker Hilbert (D. HILBERT, 1862 bis 1945) eine Liste der bisher ungelösten Probleme auf und war der Meinung, daß diese mit genügend Scharfsinn gelöst werden könnten. Erst im Jahre 1931 gelang es Gödel (K. GÖDEL, 1906 bis 1978) mit seinem *Unvollständigkeitstheorem* nachzuweisen, daß es sehr viele Probleme gibt, für die es keine Algorithmen gibt. Das bedeutet aber auch, daß die meisten Probleme gar nicht mit Rechnern gelöst werden können, auch wenn die technische Entwicklung noch so stürmisch fortschreitet. Die Theorie der Algorithmenbildung ist in der Informatik sehr wichtig. Wegen der schwierigen mathematischen Behandlung werden in diesem Buch in einfacher Weise die Eigenschaften von Algorithmen und deren Durchführbarkeit besprochen.

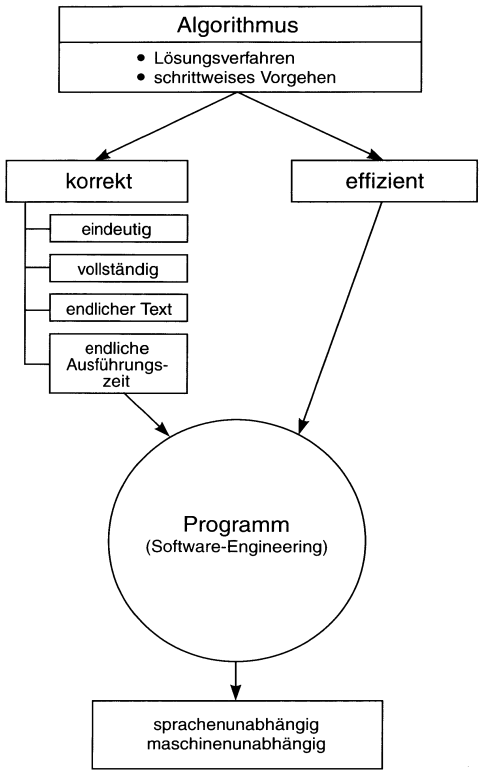


Bild A-7. Eigenschaften eines Algorithmus.

Algorithmen begegnet man beinahe überall im Leben, seien es *Gebrauchsanweisungen* für technische Geräte, *Rezepte* zum Zubereiten von Speisen, *Vorschriften* zum Einnehmen von Medikamenten oder sonstige *Regeln.* Als Beispiel dient das Rezept zur Herstellung des schwäbischen Leibgerichtes *Spätzle (für vier Personen)*:

1. Topf mit 2l heißem Wasser aufsetzen und 1 Teelöffel Salz und 1 Eßlöffel Öl hineingeben.
2. Spätzlesteig anrühren: 1 Pfund Mehl, 4 Eßlöffel Gries, 4 Eier, 1/4 l Wasser und 1 Kaffeelöffel Salz mit Rührquirl zu Teig rühren.

3. Spätzles-Maschine, Sieb und Kochlöffel naß machen.
4. Teig in Spätzles-Maschine füllen.
5. Spätzle in kochendes Wasser drücken, aufkochen lassen.
6. Spätzle mit Sieb aus kochendem Wasser entfernen. Heißes Wasser darüberlaufen lassen.
7. Wenn der Teig verbraucht ist, alle Geräte in kaltes Wasser tauchen und abwaschen.

Die meisten Algorithmen des täglichen Lebens sind selten genau ausformuliert und lassen einen erheblichen Spielraum für unterschiedliche Interpretationen, wie dies für die Zwecke der Informatik unbrauchbar ist.

A 2.4.2 Eigenschaften des Algorithmus

Wie Bild A-7 in der Mitte zeigt, muß ein Algorithmus *korrekt* und *effizient* sein. Die *Korrektheit* bezieht sich dabei auf folgende Punkte

Eindeutigkeit

Es muß sichergestellt werden, daß bei mehrfacher Ausführung desselben Problems durch unterschiedliche Ausführungsorgane immer *dieselbe Lösung* herauskommen muß.

Vollständigkeit

Ein Algorithmus kann sich nur auf Vorgänge beziehen, die genau festgelegt sind, und es ist nicht erlaubt, Daten zu verwenden, die erst in späteren Prozessen errechnet werden. Insofern läßt ein Algorithmus keinen Raum für Intuition und Kreativität.

Endliche Textlänge (statische Finitheit)

Die Anzahl der in einem Algorithmus verwendeten Texte muß endlich sein, ebenfalls die Bezeichnungen innerhalb der Anweisungen. Damit stellt sich aber ein durchaus ernst zu nehmendes arithmetisches Problem, beispielsweise bei der Verwendung eines unendlichen Dezimalbruchs wie der Zahl "∞".

Endliche Ausführungszeit (dynamische Finitheit)

Ein Algorithmus muß so geschrieben sein, daß das Problem in einer endlichen Zeit gelöst wird. Auch hier gibt es bei einfachen arithmetischen Zahlen bereits Probleme, beispielsweise bei $\sqrt{3} = 1,7320\ldots$ Diese Zahl hat unendlich viele Stellen und der Algorithmus ist deshalb nie mit der Berechnung fertig. Selbst wenn dieses numerische Problem gelöst ist, darf ein Algorithmus *keine Endlosschleife* enthalten, sondern muß nach einer endlichen Anzahl von Wiederholungen zum Schluß kommen.

Effizienz

Neben der Korrektheit des Algorithmus spielt auch die *Effizienz* des Algorithmus eine Rolle. Sie beschreibt den Zeit- und den Speicherverbrauch unterschiedlicher Algorithmen. Kann zwischen verschiedenen Algorithmen gewählt werden, dann ist immer dem Vorrang einzuräumen, der den geringsten Zeit- und Speicherverbrauch besitzt.

A 2.4.3 Realisierung des Algorithmus

Wie Bild A-7 weiter zeigt, wird ein Algorithmus in der Informatik durch ein entsprechendes *Programm (Software)* in eine solche Form übergeführt, daß eine Maschine die Anweisungen in der entsprechenden Reihenfolge abarbeiten kann. Dazu verwendet ein Programm entsprechende Daten- und Programmstrukturen und Werkzeuge des *Software-Engineering* (Abschn. D), um den Algorithmus effizient und korrekt programmieren zu können.

Es ist ganz wichtig, darauf hinzuweisen, daß die Algorithmen prinzipiell *sprachenunabhängig* und *maschinenunabhängig* sind. Programme müssen so geschrieben werden, daß die Maschine diese *versteht* (korrekter Satzbau oder *Syntax)*, daß die Anweisungen *sinnvoll* sind *(semantisch* korrekt; z. B. ist der 35. April keine semantisch korrekte Angabe), und daß die Anweisungen *logisch* richtig sind. Wie hier gezeigt wird, treten in Programmen auch diese drei Fehlerklassen auf:

- *syntaktische Fehler,*
- *semantische Fehler* und
- *logische Fehler.*

Zur Suche dieser Fehler (engl.: *bug,* was eigentlich Käfer bedeutet) werden spezielle Werkzeuge eingesetzt, die man *Debugger* nennt. Während syntaktische Fehler sehr leicht zu finden sind, sind bereits semantische Fehler schwieriger zu erkennen. Sehr schwer ist es, logische Fehler zu finden (z. B. Verwendung der falschen Formel für die Berechnung des Flächeninhalts A eines Dreiecks

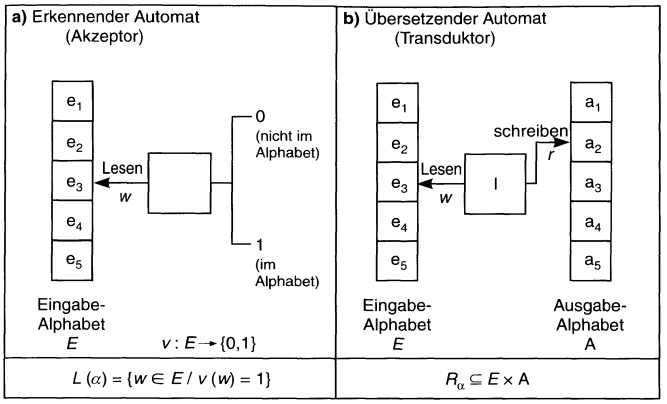


Bild A-8. Aufbau und Eigenschaften von Automaten.

als Produkt von Grundseite g multipliziert mit der Höhe h. $A = g \cdot h$ anstelle von $A = 0{,}5 \cdot g \cdot h$).

A 2.5 Automatentheorie

Im täglichen Leben begegnet man vielen Automaten: Bankautomaten, um Geld zu erhalten oder andere Automaten für Dinge des täglichen Bedarfs, wie Getränke, Essen, Süßigkeiten, Tabakwaren, Fahrkarten oder Telefon. Auch im häuslichen Bereich ist man von Automaten umgeben: Beispiele hierzu sind Kaffeeautomaten, Geschirrspül- und Waschmaschinen. Ferner sind die Automaten zur Steuerung von Produktionen oder zur Datenverarbeitung zu nennen. Im folgenden wird versucht, die Gemeinsamkeiten dieser Automaten mit Begriffen und deren Definitionen zu beschreiben. Dabei werden keine mathematischen Beweise geführt, sondern die Zusammenhänge an Beispielen veranschaulicht.

A 2.5.1 Einteilung der Automaten

Automaten können allgemein folgendermaßen beschrieben werden:

> Automaten sind Geräte, die nach einer Eingabe eine bestimmte Ausgabe tätigen.

Ein Getränkeautomat, wie er in Abschnitt A 2.5.2 besprochen wird, gibt nach erfolgter Geldeingabe das gewünschte Getränk aus. In dieser Weise geben auch Rechenautomaten mit Hilfe eines Programms und den erfolgten Eingabedaten eine Menge von Ausgabedaten aus. Wie Bild A-8 zeigt, wird zwischen *erkennenden* Automaten *(Akzeptor)* und *übersetzenden* Automaten *(Transduktor)* unterschieden.

Erkennender Automat

Ein *erkennender Automat (Akzeptor)* merkt, ob ein eingelesenes Wort w innerhalb der Sprache L liegt. Der Automat α berechnet hierzu eine Ausgabefunktion $v: E \rightarrow \{0,1\}$. Ist das Wort w ein Element des Eingabealphabetes E, dann ist das Wort erkannt, d. h. der Ausgangszustand ist 1 (Wort ist im Alphabet). Im anderen Fall ist der Ausgangszustand 0 (Wort ist nicht im Alphabet). Formal kann dies für die Sprache $L(\alpha)$ folgendermaßen ausgedrückt werden:

$$L(\alpha) = \{w \in E \, v(w) = 1\}.$$

Übersetzender Automat

Übersetzende Automaten *(Transduktoren)* berechnen eine Relation R zwischen allen Elementen des Eingangsalphabetes E und allen Elementen des Ausgangsalphabetes A (das *kartesische Produkt)*. Demnach gilt:

$$R_\alpha \subseteq E \times A$$

Übersetzer von Sprachen, beispielsweise für die Sprache PASCAL, können als *erkennende* Automaten angesehen werden, wenn sie ein eingegebenes Programm nur auf die syntaktische Richtigkeit prüfen. Wird das syntaktisch korrekte Programm

in Maschinensprache übersetzt, dann ist das ein *übersetzender* Automat.

A 2.5.2 Endlicher Automat

Ein *endlicher Automat* ist ein *mathematisches Beschreibungsmodell,* mit dem Eingaben *sofort* (d. h. ohne Verzögerung) zu Ausgaben verarbeitet werden. Die Eingabevariablen E, die internen Zustände des Automaten und die Ausgabevariablen A bestehen aus endlichen Mengen, und der Automat besitzt in diesem Modell *keinen Speicher.* Am Beispiel des Getränkeautomaten wird das allgemeine mathematische Modell hergeleitet.

Getränkeautomat

Wie in Bild A-9 im Teilbild a zu erkennen ist, kann der Benutzer des Automaten durch Einwurf eines Geldbetrages G eines von zwei möglichen Getränken, nämlich Bier (B) oder Limonade (L) erhalten. Das Geld G kann durch Drücken des Rückgabeknopfes R wieder zurückerhalten werden. Wird eine Wahltaste $W1$ oder $W2$ gedrückt, ohne daß genügend Geld eingeworfen wurde, dann ertönt ein Warnton Y.

An diesem Automaten sind *drei Zustände* zu erkennen:

1. Eingabezustand E

Der Eingabezustand wird durch eine *endliche Menge* von *Eingabezeichen* E beschrieben. Im vorliegenden Fall ist:

$$E = \{W1, W2, G, R\} \qquad \text{(A-10).}$$

Das bedeutet: Es kann die Wahl-Taste $W1$ (Bier) oder $W2$ (Limo) gedrückt werden, ferner der Rückgabeknopf R, und es kann eine Geldmenge G eingegeben werden.

2. Interner Zustand I

Der Getränkeautomat befindet sich in jedem Augenblick in einem bestimmten internen Zustand. Dieser ist allgemein:

$$I = \{v, w\} \qquad \text{(A-11).}$$

Das bedeutet, der Automat hat die beiden Zustände: Geld ausreichend vorhanden (v) oder Automat wartet (w).

3. Ausgangszustand A

Der Ausgangszustand A wird durch eine endliche Menge von *Ausgangszeichen* beschrieben. Es gilt im vorliegenden Beispiel:

$$A = \{B, L, X, Y\} \qquad \text{(A-12).}$$

Das bedeutet: Es kann Bier (B), Limonade (L), Geld (X) herauskommen oder das Warnsignal (Y) ertönen.

Mit dieser Zustandsbeschreibung ist es aber *nicht möglich,* den *dynamischen Ablauf,* d. h. das *Verhalten* des Automaten zu beschreiben. Dazu dient, wie Bild A-9 b bzw. Bild A-9 c zeigt, die *Zustandstabelle* und der *Zustandsgraph.*

In der *Zustandstabelle* werden die *Eingangsvariablen* in den Zeilen mit den *internen Zustände* in den Spalten so verknüpft, daß die *Ausgabe* erkennbar wird. Der Eintrag in die Tabelle besteht aus einem Paar: das *erste Zeichen* gibt den *Folgezustand* des Automaten an und das *zweite Zeichen* zeigt die nach dem Übergang erfolgte *Ausgabe* an. In diesem Sinne läßt sich das erste Element der Zustandstabelle in Bild A-9 b folgendermaßen deuten: Wird die Eingabetaste W1 (Wunsch nach Bier) gedrückt und der Automat ist im inneren Zustand v (Geld ist vorhanden), dann geht er in den Zustand „warten" (w) über und gibt das Bier (B) aus: dargestellt durch die Kombination w/B. (Das Ausgabezeichen „-" steht für die leere Ausgabe).

Im *Zustandsgraph* wird das Verhalten des Automaten grafisch beschrieben. Dabei geht man in folgenden Schritten vor:

1. Schritt: Interne Zustände sind Kreise

Die *internen Zustände I* werden als *Kreise* (Knoten) gezeichnet. Es empfiehlt sich, den Startzustand zu markieren und die Endzustände durch Doppelkreise zu kennzeichnen.

2. Schritt: Verbinden der Zustände

Von den Knoten gehen Pfeile aus (gerichtete Kanten). Die *Pfeilspitzen* geben den *neuen internen Zustand* an, in den der Automat übergeht. Allgemein ist anzumerken, daß von jedem Knoten so viele Pfeile ausgehen, wie Eingabezeichen vorhanden sind (im vorliegenden Fall sind dies vier).

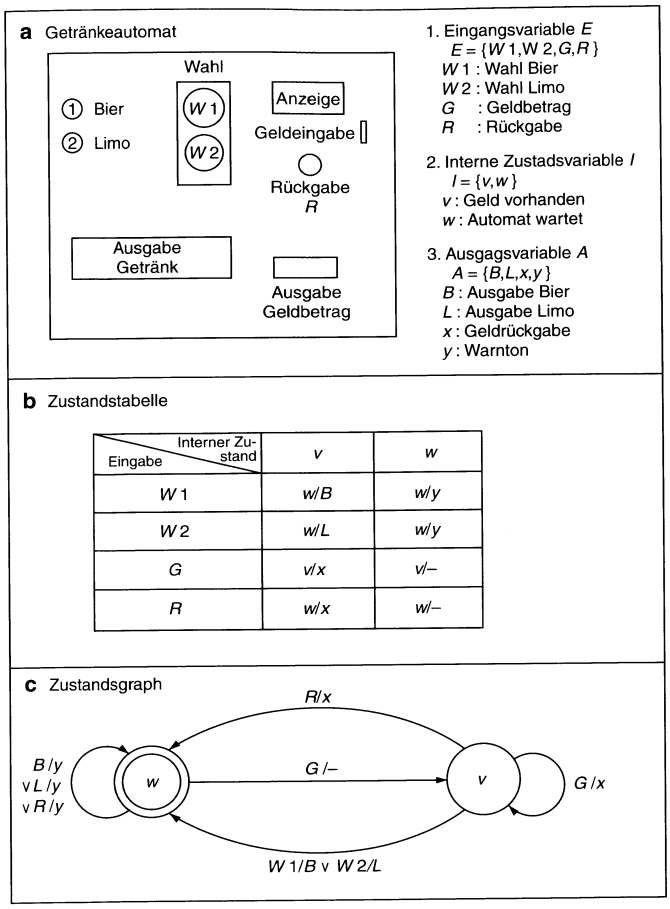


Bild A-9. Getränkeautomat als endlicher Automat.

3. Schritt: Beschriftung der Kanten

Die Kante wird durch die *Eingabevariable* und die *Ausgangsvariable* (getrennt von einem Querstrich) beschriftet.

Die untere Kante in Bild A-9 (W1/B v W2/L) kann deshalb folgendermaßen gelesen werden:

„Es ist genügend Geld eingeworfen worden (Zustand v). Wird die Taste W1 (Bierwunsch) gedrückt, dann wird B (Bier) ausgegeben, und der Automat ist im Wartezustand (Zustand w) - oder (v) es wird die Taste W2 gedrückt (Limowunsch), dann wird L (Limo) ausgegeben und der Automat wartet anschließend (Zustand w)."

Mathematisches Modell

Das mathematische Modell ist in Tabelle A-4 zusammengefaßt.

Danach ist ein endlicher Automat "α" darstellbar als *6-Tupel*:

$$\alpha = \{E, I, K, A, u, v\} \qquad \text{(A-13)}.$$

Seine Eigenschaften sind:

1. Endliche Zahl von Eingangsvariablen

Die *endliche Zahl* von Eingangsvariablen bilden die Eingangszustände:

Tabelle A-4. Eigenschaften des endlichen Automaten

Endlicher Automat $\alpha = \{E, I, K, A, u, v\}$		
Variable		
E:	Eingangsvariable $\{e_1, e_2, e_3 \ldots\}$	
I:	momentane, interne Zustandsvariable $\{i_1, i_2, i_3, \ldots\}$	
K:	künftige, interne Zustandsvariable $k_1, k_2, k_3, \ldots\}$	
A:	Ausgangsvariable $\{a_1, a_2, a_3, \ldots\}$	
Funktionen		
u:	Übergangsfunktion	u: $E \times I \rightarrow K$
v:	Ausgangsfunktion	v: $E \times I \rightarrow$ A (MEALY-Automat)
		v: $E \times K \rightarrow$ A (MOORE-Automat)
Automaten-Gleichungen		
	$K = u\ (E, I)$	
	$A = v\ (E, I)$	MEALY-Automat
	$A = v\ (E, K)$	MOORE-Automat

Typen

vollständig definiert: Übergangs- und Ausgangsfunktionen sind vollständig definiert.

MEALY-Automat: Vollständige Zustandstabellen sind vorhanden. Jedem Gesamtzustand ist ein innerer Zustand bzw. ein Ausgangszustand zugeordnet.

MOORE-Automat: Zwei vollständige Funktionstabellen: Die erste ordnet jedem Gesamtzustand einen künftigen inneren Zustand zu; die zweite der Kombination von momentanem Eingangszustand und künftigem inneren Zustand einen momentanen Ausgangszustand.

unvollständig definiert: Eine Zustandstabelle ist unvollständig.

trivial: Der Ausgangszustand ist nur abhängig vom Eingangszustand.

$E = \{e_1, e_2, e_3, \ldots\}$.

Aus diesen Elementen werden *Folgen* von Eingangszuständen gebildet, die dem Automaten *nacheinander* zugeführt werden.

2. Endliche Zahl von Ausgangsvariablen

Der Automat besitzt ebenfalls eine *endliche* Zahl von *Ausgangsvariablen*.

$A = \{a_1, a_2, a_3, \ldots\}$.

Die *Folge der Ausgangsvariablen* ist eine *Reaktion* auf die Folge der Eingangsvariablen. Das bedeutet, daß die Ausgangsvariable nicht nur vom momentanen Eingangszustand abhängt, sondern auch von allen bisher aufgetretenen Eingangszuständen.

3. Endliche Zahl von internen Zuständen

Mit einer endlichen Zahl von internen Zuständen können die Eingangszustandsfolgen registriert werden. Es sind die *momentanen* internen Zustände I von den *künftigen* internen Zuständen K zu unterscheiden.

4. Übergangsfunktion *u*

Aus der Kombination aller Eingangszustände mit allen momentanen internen Zuständen lassen sich die künftigen internen Zustände angeben:

$u: E \times I \rightarrow K$.

Diese Tabelle beschreibt den *Gesamtzustand* des endlichen Automaten.

5. Ausgangsfunktion v

Je nach Automaten-Typ wird die Ausgangsfunktion anders gebildet: Beim MEALY-Automaten wird die Ausgangsfunktion A bestimmt durch die momentanen Eingangsvariablen E und die momentanen internen Zustände I. Es gilt: $A = v(E,I)$ oder $E \times I \rightarrow A$. Beim MOORE-Automaten ist die Ausgangsfunktion aus der Kombination aller Eingangszustände mit den *künftigen* Zuständen gebildet. Es gilt $v: E \times K \rightarrow A$.

6. Automatengleichungen

Die *schaltalgebraische* Beschreibung über Boolesche Algebra oder Logikpläne (Abschn. A 4.3) der Übergangs- bzw. Ausgangsfunktionen wird *Automatengleichung* genannt.

A 2.5.3 Anwendung von endlichen Automaten

Mit endlichen Automaten können eine Reihe von Problemen beschrieben und gelöst werden. Eine Spezialität liegt im Bereich der *Sprachen* und ein anderes Anwendungsfeld im Entwurf *logischer Netzwerke.*

Sprache

Ein wichtiges Problem ist das *Wortproblem,* d. h. die Erkennung einer Symbolfolge. Beispielsweise soll ein gegebener Text nach einem bestimmten Stichwort durchsucht werden. Ein anderer wichtiger Fall ist die Prüfung einer Programmiersprache auf Fehler. Dies geschieht durch die *lexikalische Analyse* im *Scanner.* Dabei werden die Sprachelemente (z. B. für PASCAL) in folgende Bereiche zerlegt:

- *Wortsymbole* (z. B. begin, end, case, if, then, do);
- *Variablenbezeichnungen* (z. B. Bezeichnung von Variablen, Datentypen, Prozeduren, Funktionen);
- *Interpunktionszeichen* (z. B. Klammern, Strichpunkte, Kommata).

Entwurf logischer Netzwerke

Mit *Zustandszeitdiagrammen,* mit *Funktionstabellen* und mit *logischen Schaltungen* werden häufig Netzwerke entworfen. Falls die Funktionstabellen Widersprüche erzeugen, müssen diese durch die Einführung von *inneren Variablen* beseitigt werden. Bei komplizierten Aufgabenstellungen, beispielsweise wenn ein Zustandsdiagramm nicht vorhanden ist oder mehrere innere Variable erforderlich werden, ist es mit der Methode der endlichen Automaten relativ einfach, diese zu entwerfen.

A 2.5.4 Kellerautomat

Während bei *endlichen* Automaten nur endliche Folgen verarbeitet werden können, ist es für weitere Probleme von Vorteil, wenn die Beschränkungen wenigstens nach einer Richtung hin wegfallen. Dies ist beispielsweise zur syntaktischen Überprüfung von Programmiersprachen wichtig. Dies leistet der *Kellerautomat.* Unter *Keller* oder *Stapel* (engl.: *stack)* versteht man eine Datenstruktur, die am *gleichen Ende zu- und abnimmt* (Bild A-10). Wie bei einem Stapel wird das neue Element am oberen Ende hinzugefügt und an demselben Ende wieder weggenommen. Dieses Verwaltungsprinzip heißt auch *LIFO* (last in first out: das zuletzt abgelegte Element wird als erstes wieder entfernt).

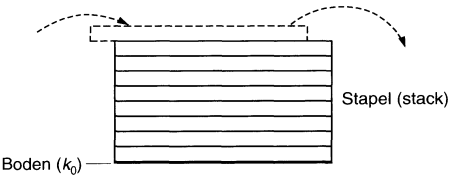


Bild A-10. Datenstruktur eines Stacks.

Ein *Kellerautomat* ist ein endlicher Automat, der um einen *Keller erweitert* wurde. Der Keller hat im wesentlichen zwei Vorteile: Zum einen können *Zustandsübergänge* als Folge des *zuletzt* gemerkten Zeichens stattfinden und zum anderen kann sich der Automat unendlich viele Zeichen merken, indem er den Keller auf- und abbaut. Die Wirkungsweise des Kellerautomaten wird an einem einfachen Rechenbeispiel $(5+(5+1)/2)/2$ für korrekte Klammerung erklärt (Bild A-11).

In Bild A-11 ist oben das *Eingabeband* zu sehen, das den arithmetischen Ausdruck enthält. An der Seite befindet sich das *Kellerband.* Das unterste Element ist das Grundzeichen „k_0". Der Kellerautomat beginnt mit seiner Arbeit. Das erste Zeichen für „(" (offene Klammer) wird gelesen und auf den Kellerpeicher gelegt, dann das zweite Zeichen „(". Diesen Zustand zeigt Bild A-11 unten. Wird anschließend das erste Zeichen

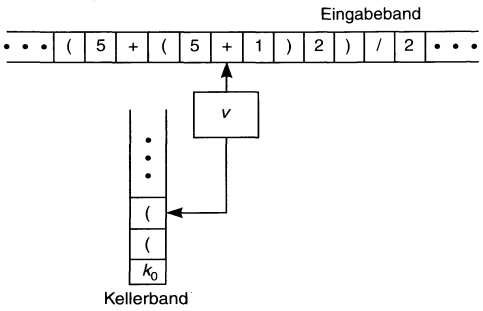


Bild A-11. Schema des Kellerautomaten.

für „Klammer zu" „)" gelesen, dann wird das letzte Zeichen „(" vom Kellerspeicher entfernt. Beim letzten „)" verschwindet auch die letzte „(", so daß das Grundelement k_0 wieder zurückbleibt. In diesem Fall war die Klammerung korrekt. Falls Klammern übrigbleiben oder fehlen, erscheint eine entsprechende Fehlermeldung. Ähnliche Untersuchungen zur Syntax sind beispielsweise bei arithmetischen Operationen ($+, -, \times, /$ müssen zwischen zwei Variablen stehen) und anderen formalen Konstrukten durch einen Kellerautomaten möglich. Voraussetzung dafür ist eine Sprache, die *kontextfrei* ist, wie alle Computerprogrammiersprachen. Das bedeutet, daß die Zeichen *nicht von den Zeichen ihrer Umgebung,* sozusagen vom Kontext, abhängig sind.

A 2.5.5 Turing-Maschine

Dieses Automatenmodell wurde von Turing (A.M. TURING, 1912 bis 1954) entwickelt und erlaubt es, jeden Algorithmus zu bearbeiten. Die Turing-Maschine besteht aus einem nach

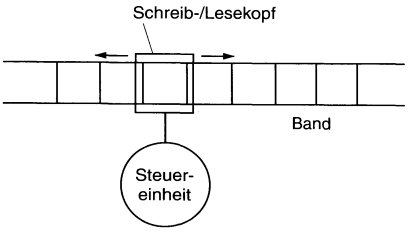


Bild A-12. Schema der Turing-Maschine.

beiden Seiten hin *unbegrenzten Band,* einem *Schreib/Lesekopf* und einer *Steuereinheit* (Bild A-12). Im Gegensatz zum Kellerautomaten, bei dem nur das oberste Element zugänglich war, ist bei der Turing-Maschine jedes Element zugänglich.

Die Turing-Maschine kann ein Feld lesen oder beschreiben. Die Steuereinheit befiehlt die Bewegungsfolgen, die als *endlicher Automat* entwickelt sein kann, so daß seine *Zustandstabelle* angibt, welche Bewegungen (vorwärts und rückwärts) in Abhängigkeit der bereits erfolgten Eingaben erfolgen. Durch schrittweises Lesen bzw. Schreiben (auch als Zwischenspeichern) wird ein Algorithmus auf der Maschine ausgeführt.

A 3 Systemanalyse und Systementwicklung

A 3.1 System, Systemanalyse und Systementwicklung

Ein *System* kann man folgendermaßen definieren (Bild A-13):

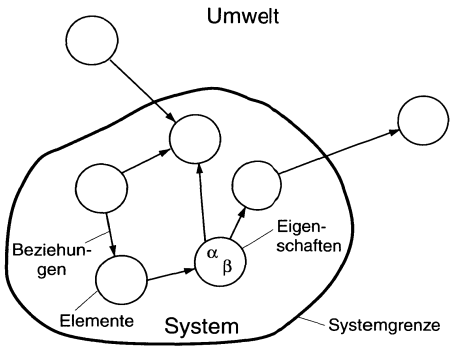


Bild A-13. Offenes, dynamisches System und seine Eigenschaften.

Unter einem System versteht man eine Vielzahl von *Elementen,* die miteinander in *Beziehung* stehen. Das System wird noch bestimmt durch die *Eigenschaften der Elemente,* deren *Beziehungen* und die *Systemgrenzen.*

Bezogen auf ein *Software-System* ergeben sich folgende Entsprechungen (Tabelle A-5):

Systeme werden häufig in folgende Klassen eingeteilt:

Tabelle A-5. Der Systembegriff in der Software

Bestandteile des Systems	Entsprechung in der Software
Elemente	*Daten* (Objekte der Datenverarbeitung) und die *Aktionen*, die diese Daten verändern (z.B. Anweisungen eines Programms) oder erzeugen (z.B. durch mathematische Funktionen).
Eigenschaften	Attribute der einzelnen Elemente (z.B. Wertebereich von Zahlen oder die Art der Zeichen wie numerisch).
Beziehungen	*Statisch* oder *dynamisch* in Bezug auf die Daten (z.B. statische und dynamische Link-Bibliotheken).
Grenzen	Festlegen des *Umfangs* einer Software. Es wird bestimmt, welche Teile der Software außerhalb der festgelegten Grenzen liegen. Wichtig ist, bereits bei der Softwareerstellung die *Schnittstellen* zu den anderen Systemen zu beachten.

- *Dynamische Systeme*
  Systeme, bei denen ein Austausch zwischen den Elementen stattfindet, beispielsweise Material-, Informations- oder Energieflüsse.
- *Offene Systeme*
  Systeme, die mit der Umgebung in Beziehung stehen.
- *Hierarchische Systeme*
  Systeme, die eine klare Gliederung in Untersysteme zulassen.

In der *Systemanalyse* wird ein Problem systematisch analysiert und durch geeignete Methoden der gewünschten Lösung zugeführt. In der Informatik befaßt sich die Systemanalyse mit der Entwicklung *computergestützter Informationssysteme,* die technische, wirtschaftliche und organisatorische Aufgaben lösen. Dabei werden nicht nur Daten verarbeitet, sondern durch die Interaktion mit den Teilnehmern Informations- und Kommunikationssysteme entwickelt, die sich in Richtung einer *integrierten Informationstechnik* entwickeln (Abschn. E 1, Bild E-1 und Bild E-4 sowie Abschn. F 1).

Die Systemanalyse ist ein Bestandteil der *Systementwicklung.* Wie Bild A-14 verdeutlicht, wird die Entwicklung durch die *Planung* des Systems und die *Organisation* der Elemente des Systems bestimmt. Um die Systeme mit vertretbaren Kosten, in der erforderlichen Qualität und in annehmbarer Zeit realisieren zu können, müssen entsprechende *Methoden* (z.B. des Software-Engineering nach Abschnitt D) eingesetzt und die Arbeiten dokumentiert werden. Im einzelnen geht man von der *Systemumgebung* aus und ermittelt die *Zielspezifikation* des Anwenders. In weiteren Schritten der *stufenweisen Verfeinerung* werden die einzelnen Systemelemente spezifiziert. Im Anschluß daran beginnt der eigentliche Entwicklungsprozeß. Er beginnt mit der *Soll-Konzeption* für die *Organisation* der Informationsflüsse und der Kommunikationsstrukturen und endet mit der *Systemimplementierung* der erarbeiteten *Hard- und Softwarelösung.*

Bei der Entwicklung komplexer Software werden in der Praxis häufig die Kosten und die Zeiten überschritten und Programme fertiggestellt, die nicht nur unwirtschaftlich sind, sondern auch häufig den Wünschen des Anwenders nicht entsprechen. Deshalb sollte man sich bei der Systementwicklung die unterschiedlichen Probleme der Anwender und der Entwickler ständig vergegenwärtigen (Tabelle A-6).

A 3.2 Phasenkonzepte

Informationssysteme sind in der Regel komplexe Systeme. Sie zu entwerfen, zu entwickeln und zu realisieren stellt hohe Ansprüche an das Projekt-

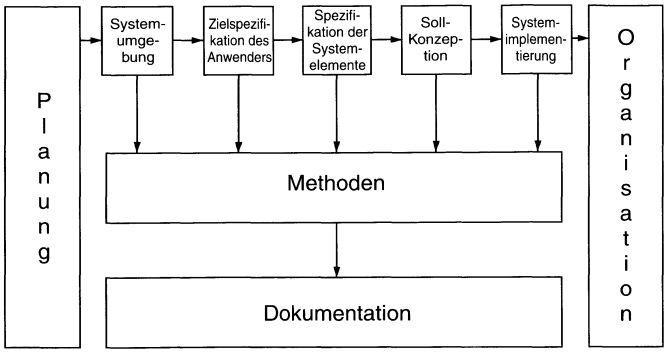


Bild A-14. Schema der Systemanalyse.

Tabelle A-6. Probleme der Anwender und der Entwickler bei der Systemanalyse

Problemart	Anwender	Entwickler
Fachlich	Schwierigkeiten bei der vollständigen Beschreibung der Systeme.	Komplexität bei Software steigend (mangelhafte Kenntnis über die Methoden und Werkzeuge des Software-Engineering).
	Zu geringe Kenntnis von den Möglichkeiten der Umsetzung der Fachanforderungen in Software.	Kürzer werdende Realisierungszeiten.
Menschlich	Fachabteilungen sprechen eine andere Sprache als Personen der DV (unterschiedliche Ziele und unterschiedliche Beurteilung gleicher Ziele).	Änderung von Anforderungen während der Realisierung.
	Fachabteilungen werden nicht in die Systementwicklung einbezogen.	Unsicherheit in Bezug auf die Methoden des Software-Engineering.
	Fachabteilungen müssen sich auf die Arbeit der Entwickler verlassen.	
Wirtschaftlich	Software entspricht nicht den fachlichen Anforderungen.	Termine und Kosten werden nicht eingehalten.
	Viele Fehler in der Testphase.	Mangelhaftes Projektmanagement.

Management. Es empfiehlt sich dabei, in Teilschritten, sogenannten *Phasen,* vorzugehen. Diese Phasen bauen so aufeinander auf, daß die nächste erst begonnen werden kann, wenn die vorhergehende abgeschlossen ist. Auf diese Weise sind Zeit, Kosten und Funktionalität der einzelnen Abschnitte gut zu kontrollieren.

Wie Bild A-15 zeigt, kann man zehn *Entwicklungs-Phasen* unterscheiden, wobei die ersten vier Phasen zur Systemanalyse gehören.

1. Vorstudie

In dieser Phase werden die zu betrachtenden Arbeitsgebiete und die zu erreichenden Ziele festgelegt *(Systemabgrenzung)*, ferner die Informationen zusammengestellt *(Systemerhebung)*, die man braucht sowie die Methoden, wie man diese Informationen ermittelt (z. B. Mengen- und Zeitgerüst sowie der Kosten einer Fertigung mit der Erhebungsmethode von Fragebögen). Die gesammelten Informationen werden anschließend

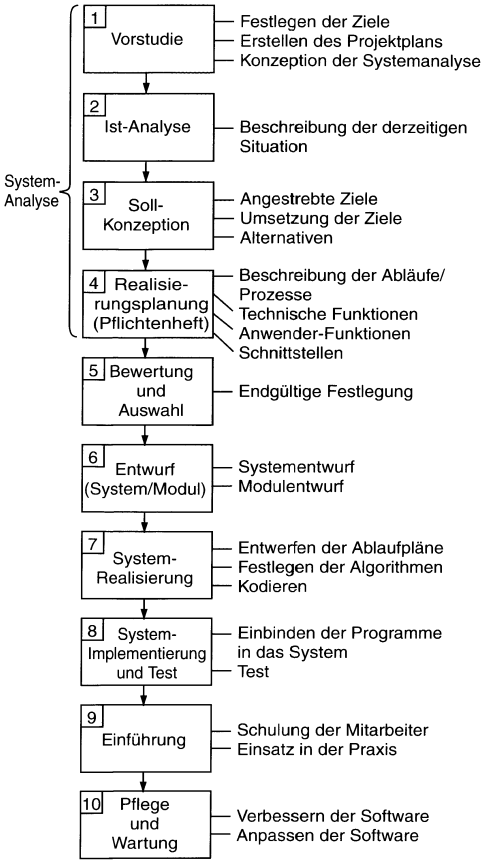


Bild A-15. Phasenmodell der Systementwicklung.

ausgewertet *(Faktenanalyse)*. Mit *ABC-Analysen* wird festgestellt, welche Aufgaben die wichtigsten sind.

2. Ist-Analyse

Hier wird die derzeitige Situation beschrieben (z. B. Dateien, Datenflüsse, Kosten, Zeiten, Zuverlässigkeit).

3. Soll-Konzeption

In der nächsten Phase wird festgelegt, welche Aufgaben das zu entwickelnde System erfüllen muß *(Festlegen der Ziele bzw. der Funktionalitäten)* und mit welchen *Lösungsansätzen* diese Ziele erreicht werden können.

4. Realisierungs-Planung (Pflichtenheft)

An dieser Stelle wird festgelegt, mit welchen *technischen Mitteln* (Hardware und Software), unter welchen *organisatorischen Voraussetzungen,* in welcher *Zeit* und mit welchen *Kosten* die gewünschte Soll-Konzeption zu realisieren ist. Dies wird in einem *Pflichtenheft* zusammengestellt.

5. Bewertung und Auswahl

Während in der Realisierungs-Planung verschiedene Alternativen denkbar sind, müssen in dieser Phase die einzelnen Varianten bewertet und ausgewählt werden.

6. Entwurf

In dieser Phase wird die ausgewählte Soll-Konzeption in ein DV-System umgesetzt. Das DV-System wird insgesamt als System und in seinen einzelnen Bestandteilen (Modulen) entworfen. Das bedeutet, es werden die Daten- und Programmstrukturen entworfen, die Algorithmen festgelegt sowie die Zugriffsverfahren und die Speicherungsstrukturen definiert.

7. System-Realisierung

In dieser Phase werden die Datenstrukturen und ihre Beziehungen festgelegt, die Ablauflogik entworfen und die Algorithmen festgelegt. Anschließend erfolgt die *Kodierung*.

8. System-Implementierung und Test

Die kodierten Programmteile werden in dieser Phase in das bestehende DV-System eingebunden. Die Programmteile werden für sich und in der Systemumgebung getestet.

9. Einführung

In dieser Phase wird das entwickelte EDV-System dem Betrieb übergeben, die Dokumentation ausgehändigt und die Schulungsinhalte und -termine für die Mitarbeiter festgelegt.

10. Pflege und Wartung

Das System muß während des Praxiseinsatzes optimiert werden, und die vorhandenen Fehler

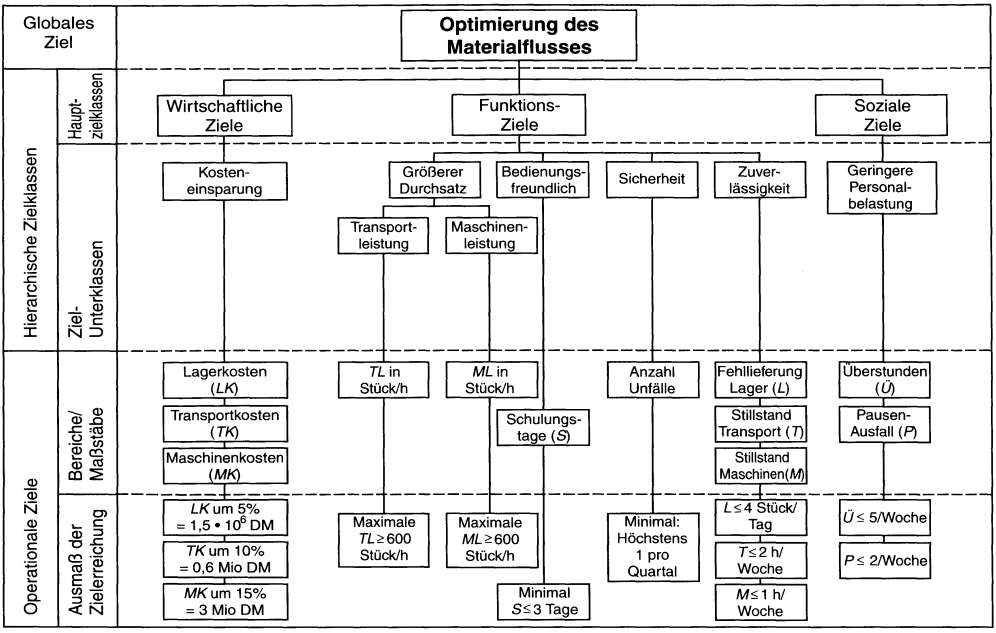


Bild A-16. Beispiel für eine Zielfindung und -festlegung.

müssen behoben werden. Neue Anforderungen zwingen zu ständigen Anpassungen der einmal entworfenen Systeme.

A 3.2.1 Vorstudie

Die Vorstudie dient nach Bild A-15 drei Zwecken:

- Bestimmung der Ziele,
- Erstellen des Projektplanes und
- Konzeption der Systemanalyse.

Bestimmung der Ziele

Wenn keine Ziele vereinbart werden, dann weiß man nicht, was man tun soll. Deshalb ist die wichtigste Aufgabe der Vorstudie als Basis für die gesamte Systementwicklung die Bestimmung der Ziele (es sei denn, die Ziele wurden bereits vorher festgelegt). Die Ziele werden zwischen Auftraggeber und -nehmer vereinbart und dienen der *Orientierung* während der Systementwicklung. Die Ziele müssen, meist ausgehend von einem globalen Ziel, als *konkrete Aufgabenstellungen (operationale Ziele)* formuliert werden. Nur dann kann der Erfolg als Maß und Grad der Zielerreichung festgestellt werden. Sinnvollerweise wird vom allgemeinen (globalen) Ziel Stufe um Stufe folgendermaßen verfeinert:

- Globale Zielfestlegung,
- hierarchische Zielgliederung in Haupt- und Unterklassen,
- operationale Ziele festlegen und Maßstäbe bestimmen sowie
- Festlegen des Ausmaßes der Zielerreichung (quantifiziertes Ziel).

Bild A-16 zeigt dieses Vorgehen an Hand der Optimierung des Materialflusses.

Weil es so sehr wichtig ist, die richtigen Ziele zu finden und sie auch klar zu formulieren, wurde eine Checkliste entwickelt (Tabelle A-7).

Es muß jedoch an dieser Stelle darauf hingewiesen werden, daß die in der Vorstudie ermittelten Ziele *nicht endgültig* sein müssen. Während der Ist- und Soll-Analyse kann sich herausstellen, daß sich Verschiebungen ergeben. Auf diese Flexibilität muß bereits bei der Erstellung und Formulierung der Ziele geachtet werden.

Tabelle A-7. Checkliste zur Festlegung von Zielen

Anforderung	Bemerkung	Beispiel
Unterscheidung in Muß-Ziele und Kann-Ziele (operationale Ziele)	Dadurch wird erkennbar, was unbedingt notwendig ist und was wünschenswert ist, d. h. auf welche Anforderungen verzichtet werden könnte.	Muß-Ziele: Anwesenheitszeiten erfassen. Kann-Ziele: Krankheits-Statistik erstellen.
Hierarchische Strukturierung der Ziele	Unterteilung des Zielsystems in Unterziele.	Unterteilung in: • technische Ziele, • wirtschaftliche Ziele, • soziale Ziele.
Formulierung muß lösungneutral erfolgen	Ziele müssen sich an Wirkungen ausrichten, nicht aber an konkreten Lösungen.	Fehlerfreie und eindeutige Erfassung der Anwesenheitsdaten (nicht: Erfassung durch Magnetkarte).
Beschreibung der Konflikte und deren Lösung	Es gibt Teilziele, die im Widerspruch zueinander stehen. Diese Widersprüche müssen erkannt und eine Lösung angegeben werden.	Programm soll sehr umfangreich sein, schnell zu erlernen und darf nur 5000,– DM kosten.
Zusammenstellung der positiven Wirkungen und der negativen Wirkungen	Die mit den Zielen erreichten Wirkungen müssen in positive und negative Wirkungen unterteilt werden.	Positive Wirkungen: Gerechter Zeitlohn. Negative Wirkungen: Möglichkeit der Personen-Überwachung.

Erstellen des Projektplanes

Wie bei jedem Projekt (s. Abschn. D 8) müssen folgende vier Aspekte berücksichtigt werden:

- *Tätigkeiten*
  (Was soll gemacht werden?).
- *Verantwortliche*
  (Wer soll das machen?).
- *Termine*
  (Bis wann muß es fertig sein?).
- *Kosten*
  (Welche Kosten dürfen höchstens enstehen?).

Um den Projektplan grafisch zu erstellen, bedient man sich der Methode der Netzplantechnik (NPT nach DIN 69 900). Die einzelnen Tätigkeiten werden nacheinander oder parallel dargestellt. Tätigkeiten, die nacheinander ablaufen, können erst begonnen werden, wenn die vorherige Tätigkeit fertig ist. Parallele Tätigkeiten können gleichzeitig bearbeitet werden. Um die gesamte Projektzeit möglichst gering zu halten, sollte man möglichst viele Tätigkeiten parallel einplanen. Für jede Tätigkeit wird ein Kästchen mit den oben aufgeführten Informationen erstellt (Bild A-17).

Neben dem Festlegen der Tätigkeiten ist die *Organisation* der Projektgruppe von besonderer Bedeutung für den Projekterfolg. Folgende Punkte müssen besonders beachtet werden:

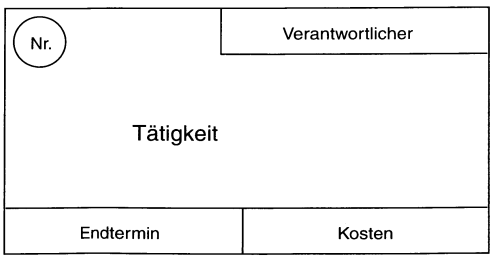


Bild A-17. Tätigkeitselement eines Netzplans.

Tabelle A-8. Kriterien zur Bewertung von Mitglieder in einem Projekt-Team

Kriterien zur Fachkompetenz	gut	mittel	schlecht
Erfahrung im Projektmanagement			
Spezialist für ...			
Übersicht über Methoden und Werkzeuge			
.			
.			
.			
Kriterien zur Team-Kompetenz			
Überzeugungsvermögen			
Durchsetzungsvermögen			
Verhandlungsgeschick			
Kooperationsfähig			
Zuverlässig			
Belastbar			
Produktivität			
.			
.			
.			

- *Zusammensetzung*
  Für die Zusammensetzung des Projekt-Teams ist darauf zu achten, daß einerseits die *Fach-Qualifikation* stimmt und andererseits die Teamfähigkeit der Personen sichergestellt ist. Tabelle A-8 zeigt Kriterien zur Beurteilung der Mitarbeiter in einem Projekt.
- *Werkzeuge*
  Es müssen der Stand der Aktivitäten und der Projektfortschritt gemessen werden. Dazu dienen Netzpläne, Tabellen und Grafiken. Es muß festgelegt werden, mit welchen Werkzeugen welche Aktivitäten kontrolliert werden. Ferner werden die Formen der Präsentationstechniken festgelegt sowie die Projektarbeit systematisch geplant.
- *Berichtswesen*
  Im Berichtswesen wird festgehalten, welche Informationen erfaßt und zu welchen Mitarbeitern mit welchen Werkzeugen und in welcher Ausführlichkeit (Detaillierungsgrad) berichtet werden soll. Im Berichtswesen stehen vor allem klare Informationen (Zahlen) zu den angefallenen Zeiten (und den abgeschätzten Restzeiten bis zur Fertigstellung), zu den aufgezehrten Kosten (und den Restkosten bis zur Fertigstellung) und der Qualität des Produktes.
- *Konzeption der Systemanalyse*
  Die in Bild A-15 gezeigten Phasen zwei bis vier der Systemanalyse müssen genauestens geplant werden.

A 3.2.2 Ist-Analyse

Mit der Ist-Analyse wird die augenblickliche Situation beschrieben:

- Aufgaben und Tätigkeiten,
- Abläufe,
- Informationen und Kommunikation,
- Mengengerüst und Datenvolumen,
- Sachmittel: Maschinen, Geräte, Räume,
- Arbeitsplätze: Arbeiten, Leistungen, Qualifikationen und
- Kosten.

Dabei gilt es, folgendes zu tun:

- Abgrenzen des Systems von der Umwelt,
- Formulierung der einzelnen Aufgaben,
- Ermitteln der Einflußgrößen,
- Ermitteln der Schwachstellen,
- Suchen nach Ursachen und
- Prüfen der Zuverlässigkeit von Informationen.

Für die Aufgabe der Optimierung des Materialflusses nach Bild A-16 bietet sich folgende

Tabelle A-9. Checkliste zur Optimierung des Materialflusses

Tätigkeiten	Ergebnis	Dokument
Erfassen der Dateien		
Transportdatei		DT 1
Maschinendatei		DM 1
Datei der Personalzeiten		DP 1
Datei der Kostenarten für Lager, Transport und Maschinen		DK 1
Erfassen der wichtigen Daten (Zeit-, Mengen- und Kostengerüst):		
Transport-Daten:	400 Stück/h	
Maschinen-Daten:	500 Stück/h	
Personal-Daten:	10 Überstunden pro Woche 4 Pausenausfälle pro Woche	
Lagerkosten:	1,725 Mio DM	
Transportkosten:	0,667 Mio DM	
Maschinenkosten:	3,53 Mio DM	
Erfassen der Datenflüsse		DF 1
Erfassen der Geräte und Maschinen		
Lager		DL 2
Transport		DT 2
Maschinen		DM 2
Meßgeräte		DG 1
Erfassen der Arbeitsplätze		DA 1
Erfassen der Kosten		s. DK 1
Prüfen des Ist-Zustandes	Ja/Nein	
Prüfung auf Vollständigkeit Prüfung auf Zuverlässigkeit Prüfen auf Richtigkeit		
Liste der fehlenden Informationen:		

Checkliste an (Tabelle A-9). Dabei sind die Ist-Zustände für die einzelnen Tätigkeiten entweder als Zahlenwert direkt verfügbar oder werden in dem aufgeführten Dokument bereitgestellt. Nach der Erfassung des Ist-Zustandes muß kritisch überprüft werden, ob die Ist-Zustände *vollständig, zuverlässig* und *richtig* sind. Je nach Ergebnis müssen die fehlenden Informationen nachträglich erfaßt werden.

A 3.2.3 Soll-Konzeption

Die Soll-Konzeption umfaßt zunächst die Festlegung der Ziele *(Bedarfs-Analyse)*. Dabei müssen folgende Punkte berücksichtigt werden:

- Forderungen des Auftraggebers,
- Änderungen der bestehenden Aufgaben und Funktionen,
- Neue Aufgaben und Funktionen,
- Neue Systemabgrenzung,
- Informationen, die zur Erfüllung notwendig sind,
- Informationsquellen.

Anschließend wird in der Soll-Konzeption festgehalten:

- Alternative Realisierungskonzepte,
- Kosten,
- Zeiten,
- Risiken.

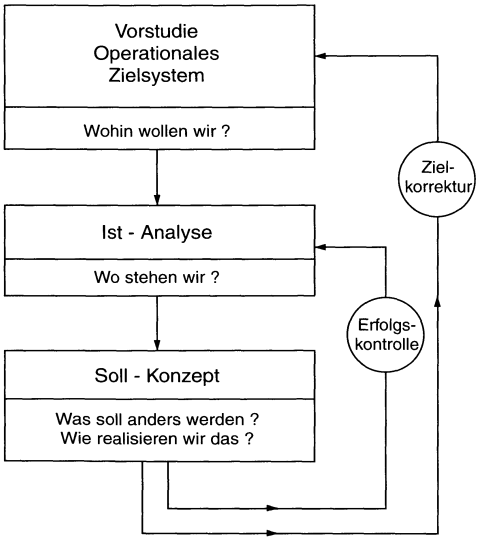


Bild A-18. Regelkreisstruktur der Systemanalyse.

Bild A-18 zeigt die Regelkreisstruktur der Systemanalyse von der Vorstudie bis zur Soll-Konzeption im Zusammenhang. Es ist zu erkennen, daß die *Vorstudie* ein *operationales Zielsystem* (z. B. nach Bild A-16) liefern muß. Dann kann in einem *Soll-Ist-Vergleich* der Erfolg gemessen werden. Bei der Entwicklung der Realisierungsalternativen kann sich herausstellen, ob die in der Vorstudie angestrebten Ziele überhaupt erreichbar sind, oder ob eine *Zielkorrektur* erfolgen muß.

A 3.2.4 Realisierungs-Planung (Pflichtenheft)

Das Pflichtenheft (Überführung der Kundenanforderungen (Lastenheft), Systembeschreibung, Anforderungsdefinition oder requirements specification genannt) ist das Ergebnis der Systemanalyse nach Bild A-15. Es soll der Nachweis geführt werden, daß das System

- die Forderungen des Anwenders (Lastenheft) korrekt und vollständig erfüllt,
- die Forderungen realisierbar sind und
- wie die Lösungen zu prüfen sind.

Das Pflichtenheft dient der *Kommunikation* zwischen Anwender und dem Entwickler. Deshalb muß es in einer Sprache geschrieben werden, die beide verstehen können. Das bedeutet, daß auf entwicklerspezifische Ausdrücke verzichtet werden muß. Beim Pflichtenheft für Software (Abschn. D 1) bedeutet dies, daß auf alle DV-spezifischen Ausdrücke verzichtet werden muß, wenn der Anwender die Anforderungen und die Lösungen verstehen will.

Ein Pflichtenheft enthält den in Tabelle A-10 dargestellten, prinzipiellen Aufbau.

Nicht im Pflichtenheft stehen dürfen konkrete Realisierungen, wie bei Software beispielsweise Programmstrukturen, spezielle Datenstrukturen und ausführliche Programmabläufe.

A 3.2.5 Bewertung und Auswahl

Zur Bewertung und zur Auswahl von Alternativen dient die *Nutzwert-Analyse*. Für die einzelnen Alternativen werden zunächst die Kriterien zur Bewertung festgelegt. Diese werden dann entsprechend ihrer Bedeutung aus Sicht des Anwenders bewertet. Das Produkt aus Gewicht und Punktzahl ergibt den Einzelnutzwert für jedes Kriterium. Der Gesamtnutzwert errechnet sich aus der Summe aller Einzelnutzwerte. Die Alternative mit dem höchsten Gesamtnutzwert ist die beste. Wird eine ideale Alternative mitgeführt, die jeweils den höchsten Einzelnutzwert erhält, dann kann ermittelt werden, zu wieviel Prozent die einzelnen Alternativen einem idealen Produkt am nächsten kommen.

Als Beispiel wird eine Aufgabe aus Bild A-16 gewählt, die Alternativen sucht und bewertet, um die *Transportleistung zu erhöhen.* Als Alternativen stehen zur Verfügung: Ein 2. Gabelstapler (2. G), ein neuer Gabelstapler mit doppelter Ladefläche (Gn) und ein fahrerloses Transportsystem (FTS). Mitgeführt wird eine ideale Alternative (Ideal).

Allgemein geht man bei der Nutzwert-Analyse in folgenden Schritten vor:

1. Aufstellen eines Kriterienkataloges

Die Kriterien sind in Tabelle A-11 zu sehen.

2. Bestimmung der Prioritäten und Errechnen der Gewichtung

Für die Kriterien werden Prioritäten aus Anwendersicht vergeben (wichtigste Priorität = 1). Es ist darauf zu achten, daß möglichst wenig gleiche Prioritäten vorkommen, um die unterschiedliche

Tabelle A-10. Prinzipielle Gliederung eines Pflichtenheftes

Gliederungspunkt	
1	Einleitung
1.1	Vorwort
1.2	Inhaltsverzeichnis
1.3	Abkürzungs- und Sachwortverzeichnis
1.4	Version
2	Beschreibung des Systems
2.1	Abgrenzung des Systems
2.2	Systembeschreibung
2.2	Systemanforderung
2.2.1	Betriebsarten
2.2.2	Leistungsdaten
2.2.3	Verhalten bei Störungen
3	Beschreibung der Maschinen und Geräte
3.1	Transportmaschinen
3.2	Bedienmaschinen
3.3	Prozeßmaschinen
3.3	Hardware-Konfiguration
3.4	Eingesetzte Software
4	Beschreibung der Systemkomponenten und Funktionen
4.1	Darstellung der Systemstruktur
4.2	Beschreibung der Funktionen
5	Beschreibung der technologischen Funktionen
6	Beschreibung der Anstoßereignisse
7	Beschreibung der Prozeß-Schnittstellen
7.1	Schnittstellen zwischen den Teilsystemen
7.2	Bussysteme, Netze und Kopplungen
8	Beschreibung der Dialog-Schnittstellen
8.1	Bildschirmmasken
8.2	Ausgabe der Listen und Protokolle
8.3	Grafikausgaben
9	Notwendige Betriebsmittel
9.1	Räume
9.2	Hardware
9.3	Systemsoftware
10	Projektablauf
10.1	Tätigkeitsplanung
10.2	Personalplanung
10.3	Terminplanung
10.4	Kostenplanung
10.5	Meilensteine
10.6	Prüfungen
11	Abnahme des Systems
11.1	Ablauf
11.2	Protokoll
11.3	Regelung zur Nachbesserung und Gewährleistung
12	Einführung
12.1	Übergabe des Systems
12.2	Schulung der Mitarbeiter

Tabelle A-11. Kriterien, Prioritäten und Gewichtung für Alternativen

Kriterien	Priorität	Gewicht
Raumbedarf	3	6
Emissionen	4	5
Sicherheit	2	7
Umbauarbeiten	8	1
Betriebskosten	6	3
Energieverbrauch	5	4
Reparaturen	7	2
Zuverlässigkeit	1	8

Bedeutung der einzelnen Kriterien deutlich erkennen zu lassen. Aus den Prioritäten lassen sich die Gewichtungen errechnen. Die höchste Priorität (1) erhält das höchste Gewicht. Dies wird folgendermaßen errechnet: Man addiert die höchste und die niedrigste Priorität (im Beispiel: $1 + 8 = 9$). Wird davon die die Priorität des jeweiligen Kriteriums abgezogen, so erhält man das Gewicht dieses Kriteriums (z. B. Kriterium „Sicherheit“: $9 - 2 = 7$, d. h. das Gewicht ist 7). In Tabelle A-11 sind die Kriterien, ihre Prioritäten und Gewichte zusammengestellt.

3. Bewertung der Kriterien

Es werden für die Noten 1 (sehr gut) bis 6 (nicht erfüllt) entsprechend Tabelle A-12 Punkte vergeben.

4. Errechnen der Einzelnutzwerte für jedes Kriterium als Produkt aus dem Gewicht und der Bewertungszahl

5. Berechnung des Gesamtnutzwertes als Summe der Einzelnutzwerte

Tabelle A-12. Noten und Punktbewertung

Note	Punkte	Bemerkung
1	100	Optimal. In allen Belangen den Anforderungen entsprechend.
2	80	Kleinere Nachteile und Mängel. Leicht unter den gestellten Anforderungen.
3	60	Spürbare Mängel und Nachteile. Deutlich unter den gestellten Anforderungen.
4	40	Erhebliche Mängel und Nachteile. Weit unter den gestellten Anforderungen.
5	20	Kaum brauchbare Lösung.
6	0	Funktion nicht vorhanden oder nicht brauchbar.

6. Berechnung des Prozentsatzes der Ideallösung

Tabelle A-13 zeigt das Ergebnis. Dabei wurden die Merkmale nach fallendem Gewicht sortiert, um die wichtigsten Merkmale zuerst zu sehen. Es ist zu erkennen, daß die beste Alternative das Fahrerlose Transportsystem (FTS) ist, gefolgt von der Alternative, einen größeren Gabelstapler anzuschaffen. Im vorliegenden Fall wird empfohlen, ein FTS anzuschaffen. In Bild A-19 zeigt die grafische Auswertung als Nutzwert-Profil-Analyse.

Für alle anderen Aufgaben gemäß Bild A-16 werden ebenfalls Alternativen gesucht und analog bewertet. Zum Schluß sind alle Alternativen ausgewählt worden, so daß mit der nächsten Phase begonnen werden kann.

Tabelle A-13. Nutzwert – Analyse

Kriterien	Gewicht	Ideal		Zweiter Gabelstapler		Größerer Gabelstapler		FTS	
		Punkte	Nutzwert	Punkte	Nutzwert	Punkte	Nutzwert	Punkte	Nutzwert
Zuverlässigkeit	8	100	800	80	640	60	480	80	640
Unfallsicherheit	7	100	700	20	140	60	420	70	490
Raumbedarf	6	100	600	40	240	80	480	80	480
Emissionen	5	100	500	20	100	40	200	100	500
Energieverbrauch	4	100	400	40	160	80	320	80	320
Kosten	3	100	300	80	240	80	240	20	60
Reparaturen	2	100	200	30	60	60	120	70	140
Umbauarbeiten	1	100	100	100	100	100	100	20	20
Summe			3600		1680		2360		2650
Erfüllungsgrad					46,67%		65,56%		73,61%

A 3.2.6 Entwurf des gesamten Systems und seiner Teile

Hierbei wird sowohl der gesamte Systementwurf erarbeitet als auch die einzelnen Module. Es gilt beispielsweise die Optimierung des Materialflusses nach Bild A-16 sowohl bei den einzelnen Modulen (d. h. operationalen Zielen), als auch im Hinblick auf die Hauptzielebene (wirtschaftlich, funktional und sozial), d. h. des gesamten Systems.

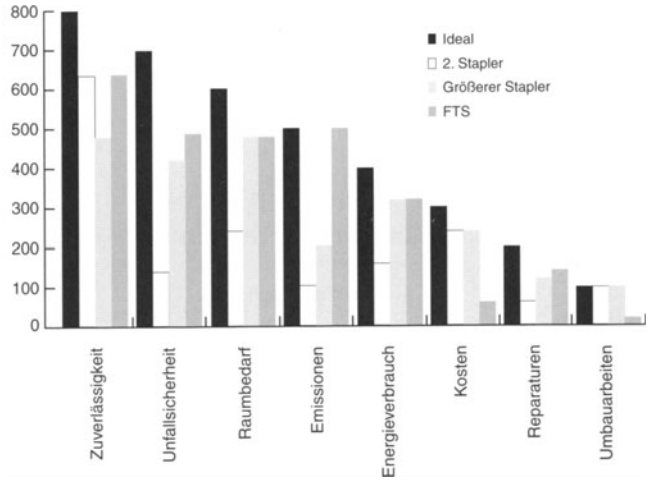


Bild A-19. Nutzwert-Profilanalyse.

A 3.2.7 System-Realisierung, System-Implementierung und Test, Einführung

Es handelt sich hier um die konkrete Lösung der Probleme und den Test. Bei der Software handelt es sich um den Entwurf der logischen Abläufe und der Modellierung der Daten sowie die Kodierung der Probleme, den Test und die Einführung. Diese Teile werden ausführlich im Abschnitt über Software-Entwicklung (D 1) abgehandelt.

A 3.2.8 Pflege und Wartung

Das System muß in der Praxiserprobung von seinen Fehlern befreit (gewartet) und den neuen Bedingungen angepaßt werden (Pflege).

A 3.2.9 Beispiel für eine Systemanalyse und Systementwicklung

In einer chemischen Fabrik wird Tonerde hergestellt. Dieses pulverförmige Produkt wird in Säcken zu 50 kg abgefüllt. Zum Versand werden jeweils 20 Säcke auf eine Palette gepackt und auf Lkw verladen (durchschnittlich 20 Paletten pro Lkw). Pro Sack ist ein Über- bzw. Untergewicht von 0,5 kg zulässig. Es kam zu folgenden Reklamationen: Die Lkws waren zu schwer beladen, und die Kunden beklagten Untergewicht. Es sollen Lösungen zur Verringerung der Reklamationen gefunden werden.

Dabei geht man enstprechend Bild A-16 vor:

1. Aufgabenbeschreibung, Zielvereinbarung und Festlegen der Meßgrößen

Die Aufgabe bestand darin, die unzulässig große Gewichtsabweichung (nach beiden Seiten) auf ihre Ursachen hin zu untersuchen und daraus Lösungsvorschläge zu erarbeiten. Das Ziel war, die zulässigen Gewichtstoleranzen (+/- 0,5 kg) einzuhalten. Als Meßgröße wurde die Anzahl der Reklamationen vereinbart.

2. Sammlung von Informationen zur Aufgabe und

3. Ordnen und Schwerpunkte bilden

Die Teilnehmer schrieben auf die Notizkarten alles, was ihnen spontan zur Aufgabenstellung einfiel. Beispielsweise

- unterschiedliche Feuchtigkeit der Tonerde,
- ungenaue Waage,
- Staubeinfluß auf die Waage,
- Unterschiede zwischen manueller und automatischer Abfüllung,
- unterschiedlicher Mengenzufluß über das Fördersystem,
- Aufstellort der Wiegeanlage,
- unterschiedliches Eigengewicht der Paletten,

- Verhältnis der reklamierten Versendungen zu nicht reklamierten.

Erfahrungsgemäß kommt es zu Mehrfachnennungen, die durch den dritten Schritt zusammengefaßt werden.

4. Vertiefte Untersuchung des ausgewählten Schwerpunktes

Bild A-20 zeigt das Ursache-Wirkungs-Diagramm der Aufgabe mit den Einflußgrößen: Maschine, Mensch, Produkt und Material und Bild A-21 ein genaueres Diagramm der Einflußgrößen.

5. Lösungsmöglichkeiten entwickeln

Ausgehend von den wesentlichen Ursachen, wurden folgende Maßnahmen abgeleitet (Tabelle A-14).

6. Bewerten der Lösungsmöglichkeiten, Auswahl des Lösungsvorschlages

Tabelle A-15 zeigt die Lösungsvorschläge.

Es wurde die heutige Kostensituation auf Grund der Reklamationsanzahl und des Aufwandes zur Ursachenbeseitigung gegenübergestellt:

Gesamtaufwand im ersten Jahr: 48 000,– DM
Gesamtreklamationen: jährlich 90 (15% von 600 Auslieferungen pro Jahr);
Kosten je Reklamation: 800,– DM;
Reklamationskosten pro Jahr: 72 000,– DM.

Da im zweiten Jahr nur noch Lohnkosten in Höhe von 5 000,– DM anfallen, ist es sinnvoll, alle Maßnahmen durchzuführen.

7. Vorstellung des Lösungsvorschlages im Betrieb, Entscheidung der Verantwortlichen

Die Ergebnisse der Arbeit wurden in der Kantine allen Betroffenen, insbesondere der Geschäftsleitung präsentiert. Die dabei verwendeten Schaubilder blieben anschließend zur Information der Betriebsöffentlichkeit noch zwei Wochen hängen. Die Geschäftsleitung gab noch im Verlauf der Präsentation die benötigten Mittel frei.

8. Umsetzen und Überprüfen der Zielerreichung

Die beschlossenen Maßnahmen wurden im Verlauf von drei Monaten umgesetzt. Nach weiteren zwei Monaten war eine deutliche Abnahme der Reklamationen festzustellen. Ein Jahr nach der Präsentation war die Reklamationsrate auf 15 pro Jahr (hochgerechnet) gesunken. Damit wurde bereits im ersten Jahr ein Einspareffekt von 12 000,- DM erreicht (60 000,- DM Einsparung abzüglich 48 000,- DM Kosten).

A 4 Zahlensysteme und Kodierungsmethoden

A 4.1 Zahlensysteme

Der Umgang mit Zahlen in unserem täglichen Leben beschränkt sich in der Regel auf das *dezimale Zahlensystem.* Trotz seiner enormen Leistungsfähigkeit ist es für die digitale Verarbeitung in Rechnersystemen ungeeignet. Zum Einsatz kommen hier *binärer Zahlensysteme* mit den zwei Zuständen 0 und 1.

Allen Zahlensystemen liegt folgendes Bildungsgesetz zugrunde (Gl.(A-14)):

$$Z = \sum_i X_i Y^i \quad i \in Z \quad 0 \leq X \leq Y \qquad \text{(A-14)}$$

Dabei ist X das *Argument,* das den Ziffernvorrat (z.B. im Dezimalsystem 0, 1, ...,9) angibt, Y die Basis des Zahlensystems (im Dezimalsystem 10) und i die Stelle (i = 0,1, ... n). Der Ziffernvorrat X (im Dezimalsystem 0 bis 9) muß stets um 1 kleiner als die Basis Y des Zahlensystems sein.

Die Auflösung der Summenformel nach Gl. (A-14) zeigt den Aufbau des Zahlensystems deutlich.

$$Z = \ldots X_3 Y^3 + X_2 Y^2 + X_1 Y^1 + X_0 Y^0 + X_{-1} Y^{-1} \ldots \qquad \text{(A-15)}$$

Argumente, die einen negativen Exponenten besitzen (in Gl. (A-15) beispielsweise $X_{-1}Y^{-1}\ldots$), ergeben in jedem Zahlensystem die *Nachkommazahlen.* Der Umgang mit den so entstandenen *Gleitkommazahlen* und die Handhabung im *binären Zahlensystem* wird im Abschnitt A 4.2.2 erläutert.

In der Digitaltechnik gibt es nur die beiden Zustände „*wahr*" (1) und „*nicht wahr*" (0). Es ist ein *binäres* Zahlensystem (d. h. es besteht aus zwei unterschiedlichen Zuständen), das als *Dualsystem* (Ziffern 0 und 1) bezeichnet wird.

Im dualen Zahlensystem ist die Basis stets 2. Das Argument einer jeden Stelle kann den Wert „0" oder „1" einnehmen.

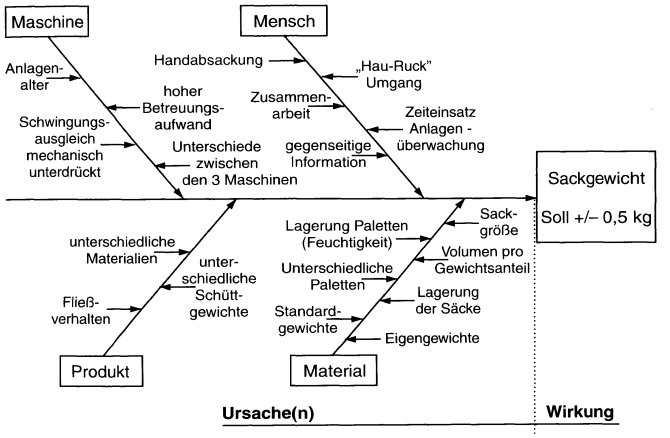


Bild A-20. Ursache-Wirkungs-Diagramm (global).

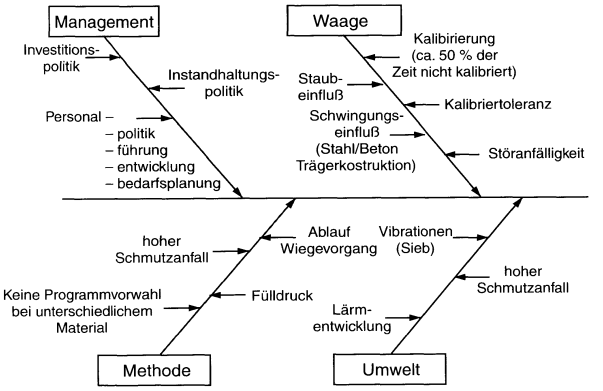


Bild A-21. Ursache-Wirkungs-Diagramm (speziell).

Tabelle A-14. Entwickeln von Möglichkeiten

Waage	• Abstützung der Trägerkonstruktion der Waage zur Verminderung von Vibrationen, • Abdeckung der Wiegemechanik, um diese staubarm zu halten, • Regelmäßige Kalibrierüberwachung der Waagen im Rahmen der vorbeugenden Instandhaltung.
Material	• Paletten trocken lagern (unter einem Dach), • Paletten-Eigengewicht und Sackgewicht anpassen und in das EDV-System eingeben.
Methode	• Einstellung der (manuell gesteuerten) Maschinen nach Produkt.

Tabelle A-15. Lösungsvorschläge

Lösungsmöglichkeit	Aufwand	Nutzen
Vibrationsarme Ausführung der Waage	Einbau einer T-Trägers (1000,– DM)	Ausschaltung der Störeinflüsse durch Vibrationen
Abdeckung der Wiegemechanik	Abdecken aller wiegemechanischen Funktionsteile (2000,– DM)	Kaum feststellbar, da beim Waagenhersteller keine tonerderesistenten Abdeckungen zu finden waren
Lagern der Holzpaletten unter Dach	Traglufthalle aus Hallenplatz (40000,– DM)	Einfluß der Paletten ausgeschaltet
Regelmäßiges Eichen der Waagen	Lohnkosten 100,– DM/Woche	Waagen sind exakt; Zeitersparnis gegenüber nachsorgender Instandhaltung

Tabelle A-16 zeigt eine Übersicht über die gebräuchlichsten Zahlensysteme, die auf einer dualen Darstellung beruhen, im Vergleich zum Dezimalsystem. Dabei ist die Wertigkeit der ersten vier Stellen (Y^0 bis Y^3) der einzelnen Zahlensysteme dargestellt, sowie das Zahlensystem in allgemeiner Formulierung angegeben. Die letzte Spalte zeigt die Summenschreibweise der einzelnen Zahlensysteme.

Im *Dualsystem* (Abschn. A 4.1.1) nennt man das Argument X auch *Bit,* ein Kurzwort, das aus dem englischen *binary digit* (binäre Einheit) abgeleitet wird.

Beispiel:

A 4-1: Setzt man in die Gl. (A-15) die Basis der einzelnen Zahlensysteme ein, so erhält man eine einfache Umrechnung in das bekannte Dezimalsystem. Zur Veranschaulichung wird die dezimale Zahl $Z_D = 269{,}3_D$ und die Binäre Zahl $Z_B = 0101{,}0_B$ nach Gl. (A-15) in ihre Argumente mit entsprechender Wertigkeit aufgelöst:

$$Z_D = 269{,}3_D$$
$$Z_D = \ldots 0 \cdot 10^3 + 2 \cdot 10^2 + 6 \cdot 10^1 + 9 \cdot 10^0 + 3 \cdot 10^{-1} + 0 \cdot 10^{-2} + \ldots$$
$$Z_D = \ldots 0 + 200 + 60 + 9 + 0{,}3 = 269{,}3_D$$

Für die dualen Zahlen ergibt sich:

$$Z_B = 0101{,}0_B$$
$$Z_B = \ldots 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + \ldots$$
$$Z_B = \ldots 0 + 4 + 0 + 1 + 0 = 5_D$$

Alle weiteren nicht aufgeführten Stellen haben stets das Argument „0", so daß diese Stellen keinen Beitrag zum Zahlenwert leisten.

A 4.1.1 Duales Zahlensystem

Das *duale Zahlensystem* ist das einfachste Zahlensystem, das sich realisieren läßt. Es hat die Basis 2, weshalb nach Gl. (A-14) das Argument die Werte 0 und 1 annehmen kann.

Da die Argumente stets kleiner als die Basis sein müssen, bleiben für das Dualsystem mit der Basis 2 lediglich die Zahlen 0 und 1 übrig. Würde man ein Zahlensystem mit der Basis 1 wählen, könnte das Argument nur noch den Wert 0 annehmen, womit sich kein Zahlensystem mehr aufbauen läßt.

Durch das Einsetzen der Basis 2 in die Gl. (A-14) erhält man für das Dualsystem:

$$Z = \sum_i X_i 2^i \quad i \in Z \quad 0 \le X < 2 \quad \text{also } X \in [0,1] \tag{A-16}$$

Die Zahlenkolonnen des dualen Zahlensystems unterliegen keinen Grenzen. Eine *acht Bit* breite Dualzahl besitzt beispielsweise die Argumente D0 bis D7, mit denen ein dezimaler Zahlenumfang von 0 bis 255 ($= 2^8 - 1$) dargestellt werden kann.

Tabelle A-16. Übersicht über die Zahlensysteme

Zahlensysteme		Allgemeine Darstellung: Wertigkeit der Stellen nach Gleichung (A-2)				Summen-Gleichung
		Y^3	Y^2	Y^1	Y^0	
	Wert:	10^3	10^2	10^1	10^0	$\Sigma X_i 10^i$
Dezimalzahl	dezimal	1000	100	10	1	i
	dezimal	1000	100	10	1	
	Wert:	2^3	2^2	2^1	2^3	$\Sigma X_i 2^i$
Dualzahl	dual	1000_B	100_B	10_B	1_B	i
	dezimal	8	4	2	1	
	Wert:	8^3	8^2	8^1	8^{30}	$\Sigma X_i 8^i$
Oktalzahl	oktal	1000_o	100_o	10_o	1_o	i
	dezimal	512	64	8	1	
	Wert:	16^3	16^2	16^1	16^0	$\Sigma X_i 16^i$
Hexadezimalzahl	hexadezimal	1000_H	100_H	10_H	1_H	i
	dezimal	4096	256	16	1	

Tabelle A-17. Zahlenbereich einer 8-Bit Dualzahl

	Most Significant Bit MSB ◄						Least Significant Bit LSB	
Argument und Wertigkeit Dezimalwert	D7 2^7	D6 2^6	D5 2^5	D4 2^4	D3 2^3	D2 2^2	D1 2^1	D0 2^0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
127	0	1	1	1	1	1	1	1
128	1	0	0	0	0	0	0	0
129	1	0	0	0	0	0	0	1
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
•								
253	1	1	1	1	1	1	0	1
254	1	1	1	1	1	1	1	0
255	1	1	1	1	1	1	1	1

Die Wertigkeit der Argumente ergibt sich aus ihrer Stelle, wie Tabelle A-17 verdeutlicht:

In Tabelle A-17 steht das Bit mit der *höchsten Wertigkeit* stets in der *linken Spalte.* Dieses Bit wird als *Most Significant Bit (MSB)* bezeichnet. Dagegen befindet sich das *niederwertigste* Bit (D0) in der *Spalte* ganz rechts. Man nennt es *Least Significant Bit (LSB).*

Da bei binären Zahlen die Zählweise bei 0 beginnt, ist die größte darstellbare Zahl stets um eins kleiner als die Potenz des Arguments k zur Basis zwei. Die größtmögliche Zahl Z_{max} die bei einer bekannten Anzahl k von Argumenten darstellbar ist, läßt sich durch folgende Gleichung berechnen:

$$Z_{max} = 2^k - 1 \qquad \text{(A-17)}$$

Beispiel:

A 4-2: Eine vierstellige Dualzahl kann insgesamt 2^4 also 16 Werte annehmen. Da der Darstellungsbereich bei „0" beginnt, kann sie nur die natürlichen Zahlen bis einschließlich 15 beschreiben, eben 0, 1, 2, 3,... 12, 13, 14 und 15, insgesamt also 16 Werte. Der höchsten Wert ergibt sich nach (Gl. A-17) zu $2^4 - 1 = 15$.

Große Dualzahlen werden oft in Feldern von 8, 16 oder 32 Bit zusammengefaßt. Man spricht dann von einem *Byte* (8 Bit), *Word* (16 Bit) oder *Double Word* oder *Long Word* (32 Bit). Auch sind bereits einzelne Rechnerstrukturen mit einer Wortbreite von 64 Bit zu finden. Die gebräuchlichsten Bezeichnungen hierfür sind *Double Long* oder *Extended Long.* Bei einer Wortbreite von nur 4 Bit spricht man von einem *Nibble* (Abschn. A 4.1.2).

4 Bit = Nibble
8 Bit = Byte
16 Bit = Word
32 Bit = Long Word
64 Bit = Double Long

A 4.1.2 Hexadezimales Zahlensystem

Die Darstellung von großen Dezimalzahlen im dualen Zahlensystem hat sich als unübersichtlich und fehlerträchtig herausgestellt. Deshalb hat man einzelne Bits zusammengefaßt und auf der Basis des Darstellungsbereiches dieser Bitgruppe ein neues Zahlensystem aufgebaut.

Als sinnvolle Teilung zeigte sich die Zusammenfassung von *4 Bit* des Dualsystems. Dadurch können $2^4 = 16$ Zustände dargestellt werden. Zur Kennzeichnung werden die 10 Zahlen des Dezimalsystems (0 bis 9) und die 6 Buchstaben des Alphabetes (A bis F) herangezogen. Deshalb nennt man dieses Zahlensystem Hexadezimalsystem. Seine Basis ist 16 und es gilt:

$$Z = \sum_i X_i \, 16^i \quad i \in Z$$
$$0 \le X < 16 \qquad \text{(A-18)}$$
$$\text{also } X \in [0,15]$$

Bei der Zusammenfassung von 4 Bit spricht man auch von einem *Halbbyte* oder einem *Nibble.* Dieses *Nibble* kann als einstellige hexadezimale Zahl gerade diese 16 Zahlen (von 0 bis 15_D) darstellen.

Unter *Halbbyte* oder *Nibble* versteht man die Zusammenfassung von *4 Bit.* Damit können 16 Zustände dargestellt werden.

Tabelle A-18 zeigt das Halbbyte einer vierstelligen Dualzahl (D0 bis D3) sowie die 16 möglichen Werte des Argumentes X der Hexadezimalzahl nach Gl. (A-18). Auf diese Weise lassen sich beispielsweise *16 Bit* breite Dualzahlen durch eine *vierstellige* Hexadezimalzahl darstellen *(16 Bit = 4 Nibbles bzw. 4 Halbbytes).*

A 4.2 Erweiterungen zum binären Zahlensystem

A 4.2.1 Negative Zahlen

Für die Darstellung *negativer Zahlen* muß ein *weiteres* Bit als Vorzeichenbit zur Verfügung gestellt werden. Dieses *Vorzeichenbit* besitzt den Wert *„0"* bei einer *positiven Zahl,* den Wert 1 bei einer *negativen Zahl.* Die einfachste Art, eine negative Zahl darzustellen, ist die *Vorzeichen-Betrags-Darstellung* (VBD).

Zur Veranschaulichung wird in der Vorzeichen-Betrags-Darstellung die dezimale Zahl 11 im Dualsystem sowohl positiv als auch negativ dargestellt:

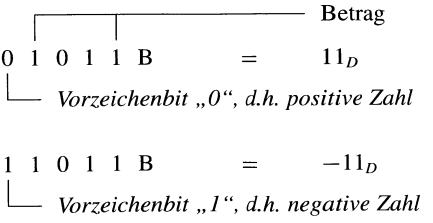


Tabelle A-18. Darstellung des Wertebereiches des Argumentes einer Hexadezimalzahl

duale Darstellung				dezimale Darstellung		hexadezimale Darstellung
D3	D2	D1	D0	Z1	Z0	H0
2^3	2^2	2^1	2^0	10^1	10^0	16^0
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	2	2
0	0	1	1	0	3	3
0	1	0	0	0	4	4
0	1	0	1	0	5	5
0	1	1	0	0	6	6
0	1	1	1	0	7	7
1	0	0	0	0	8	8
1	0	0	1	0	9	9
1	0	1	0	1	0	A
1	0	1	1	1	1	B
1	1	0	0	1	2	C
1	1	0	1	1	3	D
1	1	1	0	1	4	E
1	1	1	1	1	5	F
vierstellige Zahl				zweistellige Zahl		einstellige Zahl

Dieses Beispiel verdeutlicht, daß die negative Zahl $-11D$ sich nicht von der positiven Dualzahl 27_D unterscheidet. Bei der Verwendung eines Vorzeichenbits muß deshalb der Entwickler durch die Angabe des Darstellungsbereiches für Eindeutigkeit sorgen. Eine mit Vorzeichen versehene fünfstellige Dualzahl hat den Wertebereich von -16 bis $+15$ und kann somit nie den Wert $+27$ einnehmen, wie eine vorzeichenlose fünfstellige Dualzahl (Darstellungsbereich: 0 bis $+31$).

Für die Verarbeitung in Prozeßsteuerungen oder in Signalverarbeitungs-Rechnern hat sich obige Darstellung von negativen Zahlen als ungeeignet erwiesen. Hier geht man allgemein in die Darstellung als *Zweierkomplement (ZK)* über. Die Bedeutung des Vorzeichenbits bleibt dabei erhalten: positive Zahlen werden mit einer führenden „0" gekennzeichnet, negative Zahlen mit „1". Die nachfolgenden Bits bei den negativen Zahlen bilden jedoch das Zweierkomplement zur positiven Zahl. Unter dem Zweierkomplement versteht man die *Ergänzung* der *positiven Zahl* auf die *Basis* des Zahlensystems. So gilt für das Zweierkomplement des Hexadezimalsystems:

> Das Zweierkomplement (ZK) einer Hexadezimalzahl ist die Ergänzung auf ihre Basis 16. (A-19)

Zur Vervollständigung soll hier der Begriff des *Einerkomplements* erklärt werden. Es stellt die *Differenz* der bestehenden Zahl zur *maximal darstellbaren Zahl* dar und wird durch eine einfache *Inversion* (0 wird 1 und 1 wird 0) in der binären Schreibweise gewonnen.

> Das Einerkomplement (EK) einer Hexadezimalzahl ist die Ergänzung zur höchsten Zahl 15. Es ergibt sich aus dem Inversen der Dualzahl.

Beispiel:

A 4-3: Um den Umgang mit den Komplementzahlen zu veranschaulichen, soll das Zweierkomplement zur hexadezimalen Zahl 9_H gesucht werden:

$1\ 0\ 0\ 1_B \quad = 9_H$

Das Zweierkomplement zur Zahl 9_H errechnet sich aus der Ergänzung zur Basis 16. Dies ergibt eine Differenz von 7_H:

ZK: 0 1 1 1_B = 7_H

Zur Probe kann man nun die Zahl und ihr Zweierkomplement addieren, und es muß die Zahl null sowie ein Übertrag herauskommen:

	1	0	0	1_B	=	9_H
	0	1	1	1_B	=	7_H
1	0	0	0	0_B	=	10_H

└ Übertrag

Da das Argument der Hexadezimalzahl nur *vier Bit* breit ist, kann der Übertrag durch diese einstellige Zahl nicht mehr dargestellt werden, so daß das Ergebnis dieser Addition null ist. Die Bildung des Zweierkomplements wird rechnertechnisch aus dem *Inversen* der positiven Zahl gebildet, zudem noch „1" hinzuaddiert wird. Es wird also das *Einerkomplement* um 1 erhöht.

1	0	0	1_B	
0	1	1	0_B	Inverse zur positiven Zahl (Einerkomplement)
			1_B	Addition von 1
0	1	1	1_B	Zweierkomplement der positiven Zahl

Die Verwendung von Zahlen im Zweierkomplement wird stets in Verbindung mit einem *Vorzeichenbit* vorgenommen, welches die Zweierkomplement-Darstellung eindeutig kennzeichnet. Tabelle A-19 gibt die Zahlen einer *5 Bit* breiten Zahl und ihr Zweierkomplement wieder. Durch das Vorzeichenbit müssen *6 Bit* bereitgestellt werden. Auch fällt auf, daß die Darstellung der Zahl null bei den positiven Zahlen und im Zweierkomplement gleich ist. Aus diesem Grund ist es möglich, die negative Zahl -32_D mit nur fünf Bits darzustellen.

A 4.2.2 Festkomma- und Gleitkommazahlen

In obigen Beispielen ist man stets von der *Festkomma-Darstellung* einer Zahl ausgegangen. Sie ist gekennzeichnet durch eine bestimmte Anzahl von *Vorkomma-* und *Nachkommastellen.* Im allgemeinen arbeitet man im hexadezimalen und binären Zahlensystem *ohne* Nachkommastellen. Dies hat den Nachteil, daß der Zahlenbereich zwar begrenzt ist, aber für Steuerungszwecke ausreicht. Auf dem Zahlenstrahl in Bild A-22 sind im oberen Teil die dualen Zahlen in Abhängigkeit ihrer Breite aufgetragen. Eine 16-Bit Zahl erreicht ihren maximalen Wert bei $2^{16} - 1$, also bei 65 535.

Eine wesentliche Erweiterung des Zahlenbereiches bringt das Hinzufügen eines *Exponenten,* wie im unteren Teil von Bild A-22 dargestellt ist.

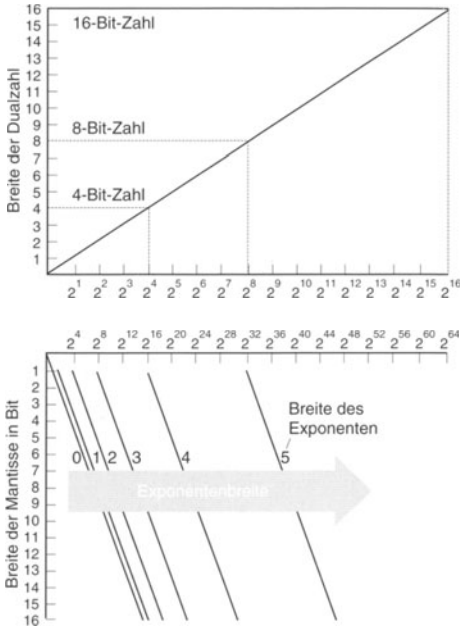
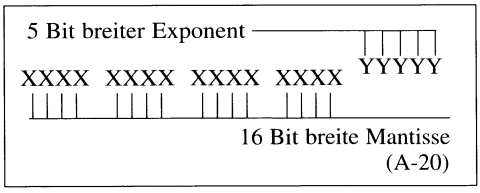
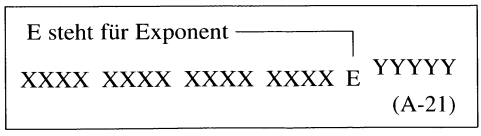


Bild A-22. Vergleich des Darstellungsbereichs von Dual- und Gleitkommazahlen.

Der Aufbau einer *binären Gleitkommazahl* ist dem einer Dezimalzahl gleich:



In der Regel wird eine andere Schreibweise benutzt:



Der *Exponent* ist dabei die *Hochzahl,* die angibt, wie oft die Basis mit sich selbst multipliziert werden muß (z.B. ist $2^3 = 2 \cdot 2 \cdot 2$). Die *Mantisse*

Tabelle A-19. Negative Dualzahlen in der Zweierkomplement Darstellung

positive Dualzahlen							negative Dualzahlen in Zweierkomplement Darstellung						
Dezimal-zahl	D5 VZ	D4 2^4	D3 2^3	D2 2^2	D1 2^1	D0 2^0	Dezimal-zahl	D5 VZ	D4 2^4	D3 2^3	D2 2^2	D1 2^1	D0 2^0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	−1	1	1	1	1	1	1
2	0	0	0	0	1	0	−2	1	1	1	1	1	0
3	0	0	0	0	1	1	−3	1	1	1	1	0	1
4	0	0	0	1	0	0	−4	1	1	1	1	0	0
5	0	0	0	1	0	1	−5	1	1	1	0	1	1
6	0	0	0	1	1	0	−6	1	1	1	0	1	0
7	0	0	0	1	1	1	−7	1	1	1	0	0	1
8	0	0	1	0	0	0	−8	1	1	1	0	0	0
9	0	0	1	0	0	1	−9	1	1	0	1	1	1
10	0	0	1	0	1	0	−10	1	1	0	1	1	0
11	0	0	1	0	1	1	−11	1	1	0	1	0	1
12	0	0	1	1	0	0	−12	1	1	0	1	0	0
13	0	0	1	1	0	1	−13	1	1	0	0	1	1
14	0	0	1	1	1	0	−14	1	1	0	0	1	0
15	0	0	1	1	1	1	−15	1	1	0	0	0	1
16	0	1	0	0	0	0	−16	1	1	0	0	0	0
17	0	1	0	0	0	1	−17	1	0	1	1	1	1
18	0	1	0	0	1	0	−18	1	0	1	1	1	0
19	0	1	0	0	1	1	−19	1	0	1	1	0	1
20	0	1	0	1	0	0	−20	1	0	1	1	0	0
21	0	1	0	1	0	1	−21	1	0	1	0	1	1
22	0	1	0	1	1	0	−22	1	0	1	0	1	0
23	0	1	0	1	1	1	−23	1	0	1	0	0	1
24	0	1	1	0	0	0	−24	1	0	1	0	0	0
25	0	1	1	0	0	1	−25	1	0	0	1	1	1
26	0	1	1	0	1	0	−26	1	0	0	1	1	0
27	0	1	1	0	1	1	−27	1	0	0	1	0	1
28	0	1	1	1	0	0	−28	1	0	0	1	0	0
29	0	1	1	1	0	1	−29	1	0	0	0	1	1
30	0	1	1	1	1	0	−30	1	0	0	0	1	0
31	0	1	1	1	1	1	−31	1	0	0	0	0	1
							−32	1	0	0	0	0	0

VZ = Vorzeichen (0: +; 1: −).

entspricht dem *Argument* der Zahlensysteme. Da es sich jedoch um eine *Gleitkommazahl* handelt, ist der Darstellungsbereich der Mantisse nicht nur auf die ganzen Zahlen beschränkt, sondern deckt den gesamten *reellen Zahlenbereich* innerhalb des benutzten Zahlensystems ab.

Die untere Hälfte in Bild A-22 zeigt den Zahlenbereich der Gleitkommazahlen in Abhängigkeit von ihrer Mantissenbreite und der Breite des Exponenten. Wird der Exponent gleich null gesetzt, so entspricht diese Gerade genau dem Zahlenbereich von Festkommazahlen (Gerade in Bild A-22 oben).

Gleitkommazahlen können ebenfalls im Zweierkomplement dargestellt werden. Dies erweitert den Zahlenbereich nochmals erheblich, es muß jedoch, wie oben bereits erwähnt, ein Vorzeichenbit geführt werden.

In der Regel wird das Vorzeichenbit den Dualzahlen vorangestellt. Vereinzelt gibt es jedoch Sy-

steme, die das Vorzeichenbit hinten anfügen (z.B. Siemens).

Die Basis des Exponenten ist zwei. So kann der Exponent im obigen Beispiel als größte positive Zahl $0\ 1\ 1\ 1\ 1_B$, also 15_D einnehmen. Die führende 0 gibt an, daß es sich um eine positive Zahl handelt. Zur Basis 2 gerechnet ergibt sich ein maximaler Multiplikator von 32767 $(= 2^{15} - 1)$. Die größte negative Zahl erhält man aus dem Zweierkomplement zu $1\ 0\ 0\ 0\ 0_B$, was -16_D entspricht. Die Mantisse wird dann mit $-65536 (= -2^{16})$ gewichtet. Der kleinste Exponent kann natürlich 0 sein, wodurch die Zahl stets den Wert der Mantisse annimmt. Mit Hilfe des negativen Exponenten können sehr kleine reelle Zahlen dargestellt werden. Negative Zahlen erhält man durch das Zweierkomplement der Mantisse. Die vier Möglichkeiten der Vorzeichenkombinationen sind durch den Zahlenstrahl in Bild A-23 aufgezeigt.

Bewegt sich die Zahl im Wertebereich der Mantisse, ist es immer möglich, den Exponenten null werden zu lassen. Wie beim geläufigen Dezimalsystem beeinflußt die *Kommastelle* den Exponenten (siehe hierzu auch die Beispiele unten).

Beispiel:

A 4-4: Der Begriff *Gleitkommazahl* wird an einer dezimalen Zahl und an einer binären Zahl veranschaulicht. Dazu soll der Wert der Zahlen *konstant* bleiben, der Exponent sich aber in Abhängigkeit der Kommastelle ändern. Am deutlichsten kann das bei den dezimalen Zahlen nachvollzogen werden:

$$\begin{aligned} 108_D &= 108 \cdot 10^{0D} = 10,8 \cdot 10^{1D} = 1,08 \cdot 10^2 \\ &= 0,108 \cdot 10^3. \end{aligned}$$

Das dies auch für das binäre Zahlensystem gilt, zeigt folgende Dualzahl:

$$\begin{aligned} 01101100 \cdot 2^{000B} &= 0110110,0 \cdot 2^{001B} \\ &= 011011,00 \cdot 2^{010B}. \end{aligned}$$

Aus diesem Beispiel wird auch der Begriff *Gleitkommazahl* ersichtlich. Beide Zahlen stellen die Zahl 108_D dar. Rückt man bei der binären Darstellung das Komma hinter das Vorzeichenbit, so erhält man die *normalisierte* Darstellung der Gleitkommazahl. Unter einer *normalisierten* Zahl versteht man eine Gleitkommazahl, bei der sich Mantisse und Exponent von der Wertigkeit nicht *überschneiden.* Dies ist dann der Fall, wenn die Mantisse Vorkommastellen besitzt wie beispielsweise 011,11... . Diese Vorkommastellen können ebenso die Werte 2^1 und 2^0 annehmen und zum Exponenten addiert werden. Gleiches gilt für Nachkommastellen, die nicht unmittelbar nach dem Komma folgen. Für sie gilt, daß sie in diesem Fall vom Exponenten abgezogen werden können. Als normalisiert gilt eine Mantisse also nur dann, wenn die einzige Vorkommastelle das Vorzeichenbit ist, und die erste Nachkommastelle sich vom Vorzeichenbit unterscheidet (z.B. $M = 0,1101011001$).

A 4.3 Grundlagen der Booleschen Algebra

A 4.3.1 Binäre Verknüpfungen

Die Verknüpfungen innerhalb der binären Zahlensysteme werden auf die Gesetze des britischen Mathematiker und Philosophen George Boole (G. Boole, von 1815 bis 1864) zurückgeführt. Es handelt sich dabei um einen *Formalismus,* der in der Lage ist, *logische Aussagen* und *Funktionen* zu beschreiben, die *zwei Zustände* einnehmen können. Handelt es sich um Schaltvorgänge, spricht man von der *Schaltalgebra,* die Grundlage für jeden Rechner und jedes Anwenderprogramm ist.

Die Boolesche Algebra kennt zwei zulässige Zustände:

wahr	= logisch 1	= Spannung vorhanden
nicht wahr	= logisch 0	= keine Spannung

(A-22)

Da ein Element der Booleschen Algebra diese beiden Zustände einnehmen kann, spricht man auch von *binären Elementen* (Abschn. A 4.1). Es gibt *drei binäre Basiselemente,* die *NICHT-Funktion* (Negation), die *UND-Funktion* (Konjunktion) und die *ODER-Funktion* (Disjunktion).

Eine Sonderform der ODER-Verknüpfung ist die *exklusive-ODER-Verknüpfung* (EXOR). Im Gegensatz zur obigen ODER-Funktion handelt es sich hierbei um ein *ausschließliches* ODER (entweder - oder), auch *Antivalenz* genannt. Die Antivalenz ist nur dann erfüllt, wenn sich die Eingangsvariablen unterscheiden.

In der Schaltalgebra wurde für die Antivalenz das Verknüpfungszeichen (Pluszeichen im Kreis) eingeführt. Die Verknüpfung selbst kann aus den bereits bekannten UND- und ODER-Verknüpfungen sehr einfach hergeleitet werden:

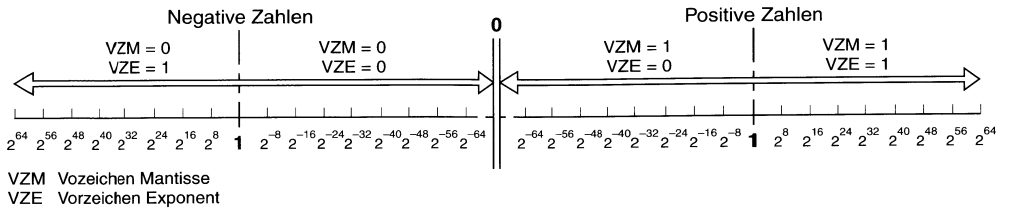


Bild A-23. Zahlenbereich einer Gleitkommazahl.

$$A = (\overline{E}_1 \cdot \overline{E}_2) + (E_1 \cdot E_2) \qquad \text{(A-23)}$$
$$A = E_1 \oplus E_2$$

Beide Gleichungen erfüllen die Wahrheitstabelle der Antivalenz in Bild A-24 (der Strich über dem Buchstaben bedeutet *Verneinung*, d.h. Negation).

A 4.3.2 Gesetze von Boole und De Morgan

Diese grundlegenden Verknüpfungen gehorchen den selben Rechenregeln, wie sie aus der Algebra bekannt sind. Boole hatte dies als erstes untersucht und sie in den folgenden Gesetzen der Schaltalgebra *(Boolesche Algebra)* zusammengefaßt.

1. Kommutativgesetz:

Das Kommutativgesetz erlaubt die Reihenfolge der Variablen innerhalb einer Operation zu verändern. Es gilt:

$$A + B = B + A \text{ und}$$
$$A \cdot B = B \cdot A \qquad \text{(A-24)}$$

2. Assoziativgesetz

Das Assoziativgesetz erlaubt die Vertauschung der Reihenfolge von gleichrangigen Operatoren:

$$A + B + C = (A + B) + C = A + (B + C) \text{ und}$$
$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C) \qquad \text{(A-25)}$$

3. Distributivgesetz

Das Distributivgesetz ermöglicht das Ausmultiplizieren von Klammerausdrücken. Dabei ist auf die Rangfolge der Operatoren zu achten. Es gilt:

$$A \cdot (B + C) = A \cdot B + A \cdot C \text{ oder } (A + B) \cdot (A + C) =$$
$$A + B \cdot C \qquad \text{(A-26)}$$

4. Absorptionsgesetz

Die Absorptionsgesetze sind das wichtigste Mittel bei der Vereinfachung von Gleichungen (siehe Distributivgesetz). Durch sie ist festgeschrieben, unter welchen Bedingungen Variable zu Konstanten werden, sich auslöschen oder sich selbst wiedergeben. Es gilt:

$$A + 0 = A$$
$$A + 1 = 1$$
$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot A = A$$
$$A + A = A$$

$$A + \overline{A} = 1$$
$$A \cdot \overline{A} = 0$$
$$A + (A \cdot B) = A$$
$$A \cdot (A + B) = A$$
$$A + \overline{A} \cdot B = A + B \qquad \text{(A-27)}$$

5. Doppelte Negierung

Wird eine Variable zweifach negiert, so heben sich die Negierungen auf. Somit gilt:

$$\overline{\overline{A}} = A \qquad \text{(A-28)}$$

Dies gilt auch dann, wenn die Variable mehrfach negiert ist. Beispielsweise reduziert sich eine dreifache Negierung der Variablen A auf eine einfache Negierung.

Beispiel:

A 4-5: Mit Hilfe von Tabelle A-20 soll die *ODER-Normalform* der Ausgangsvariablen Y gefunden werden. Diese soll anschließend mit den Gesetzen der Booleschen Algebra vereinfacht werden.

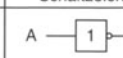
Eingangsvariable						Verknüpfung nach Boole	Bauelement	
	A	0	1	0	1		Schaltzeichen	Bezeichnung
	B	0	0	1	1			
Ausgangsvariable	Y	1	0	1	0	$Y = \overline{A}$		Inverter
	Y	0	0	0	1	$Y = A * B$		AND-Gatter
	Y	0	1	1	1	$Y = A + B$		OR-Gatter
	Y	0	1	1	0	$Y = A \oplus B$		EXOR-Gatter

Bild A-24. Übersicht über die Booleschen Verknüpfungen.

Tabelle A-20. Konjunktionstabelle zu Beispiel A 4–5

Eingangsvariablen				Ausgangsvariable	Vollkonjunktionen
A	B	C	D	Y	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	$\overline{A}$* B* $\overline{C}$* D
0	1	1	0	0	
0	1	1	1	1	$\overline{A}$* B* C* D
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	A* B* $\overline{C}$* D
1	1	1	0	0	
1	1	1	1	1	A* B* C* D

Das Beispiel enthält vier *Vollkonjunktionen* (In Tabelle A-20 grau hinterlegt), bei denen der Ausgang $Y = 1$ wird. Ihre ODER-Verknüpfung führt nun zur *ODER-Normalform*:

$$Y = (\overline{A} \cdot B \cdot \overline{C} \cdot D) + (\overline{A} \cdot B \cdot C \cdot D) + (A \cdot B \cdot \overline{C} \cdot D) + (A \cdot B \cdot C \cdot D) \qquad \text{(A-29)}.$$

Zur Verdeutlichung wurden in dieser ODER-Normalform die vier Vollkonjunktionen in Klammern gesetzt. Nach dem Distributivgesetz kann hier die Variable D ausgeklammert werden, da sie in allen Vollkonjunktionen vorhanden ist:

$$Y = ((\overline{A} \cdot B \cdot \overline{C}) + (\overline{A} \cdot B \cdot C) + (A \cdot B \cdot \overline{C}) + (A \cdot B \cdot C)) \cdot D \qquad \text{(A-30)}$$

Distributivgesetz

In den verbleibenden Konjunktionen kann die Variable C durch das Absorptionsgesetz ($\overline{C} + C = 1$) eliminiert werden, im weiteren ebenso die Variable A:

$$Y = ((\overline{A} \cdot B) + (A \cdot B)) \cdot D$$

Absorptionsgesetz

$$\underline{Y = B \cdot D} \qquad \text{(A-31)}.$$

Die zunächst sehr kompliziert aussehende ODER-Normalform für die Wahrheitstabelle läßt sich nach der Anwendung der algebraischen Regeln nach Boole durch eine *UND-Verknüpfung* der Variablen B und D realisieren (Gl. A-31).

6. Gesetze von De Morgan

Eine weitere wichtige Beziehung zwischen UND- und ODER-Verknüpfung fand der englische Mathematiker *De Morgan* (De Morgan, von 1806 bis 1871) und faßte sie in den beiden *Gesetzen von De Morgan* zusammen.

1. Gesetz von De Morgan

Negiert man eine ODER-Verknüpfung, so ist dies einer UND-Verknüpfung gleich, bei der die einzelnen Elemente negiert sind.

$$\overline{A + B + C + \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \ldots \qquad \text{(A-32)}$$

2. Gesetz von De Morgan

Negiert man eine UND-Verknüpfung, so ist dies einer ODER-Verknüpfung gleich, bei der die einzelnen Elemente negiert sind.

$$\overline{A \cdot B \cdot C \cdot} \ldots = \overline{A} + \overline{B} + \overline{C} + \ldots \qquad \text{(A-33)}$$

Beweis der De Morganschen Gesetze

Die De Morganschen Gesetze sind ein wichtiges Hilfsmittel in der Schaltalgebra bei der Optimierung von Gleichungen, in denen vor allem lange Negationen vorkommen. Diese Negationen können aufgelöst werden und ermöglichen so die Umrechnung von NOR-Schaltungen und NAND-Schaltungen (NOR = NOT-OR, NAND = NOT-AND, d.h. die Ausgänge der Basisverknüpfungen (OR und AND) sind negiert). Durch eine einfache Wahrheitstabelle läßt sich die Gültigkeit der Gesetze beweisen:

A	B	$A \cdot B$	$\overline{A \cdot B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

$\overline{A \cdot B} = \overline{A} + \overline{B}$

Beweis des 1. De Morganschen Gesetzes für zwei Eingangsvariablen (A-34)

A	B	$A + B$	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

$\overline{A + B} = \overline{A} \cdot \overline{B}$

Beweis des 2. De Morganschen Gesetzes für zwei Eingangsvariablen (A-35)

Bei der Anwendung der Gesetze von De Morgan in einer Gleichung können deshalb Konjunktionen in Disjunktionen und umgekehrt umgewandelt werden. Beim Einfügen von Negationen ist darauf zu achten, daß stets *beide* Gleichungsseiten in der selben Weise behandelt werden. So gilt beispielsweise:

$Y = A \cdot B$	Konjunktion
$\overline{Y} = \overline{A \cdot B}$	Konjunktion auf beiden Seiten negiert!
$\overline{Y} = \overline{A} + \overline{B}$	Disjunktion nach 2. De Morganschen Gesetz

Soll die Ausgangsvariable (hier Y) nicht negiert werden, so kann durch die doppelte Negation (Boolesches Gesetz nach Gleichung (A-28)) der Wert einer Seite ebenfalls erhalten werden. Zur Anwendung der De Morganschen Gesetze kann diese nun aufgebrochen werden:

$Y = A \cdot B$	Konjunktion
$Y = \overline{\overline{A \cdot B}}$	doppelte Negation, nichts hat sich geändert
$Y = \overline{\overline{A} + \overline{B}}$	Disjunktion nach Aufbrechen einer Negation und Anwendung des 2. De Morganschen Gesetzes

Diese grundlegende Anwendung der De Morgansche Gesetze hat in der Praxis große Bedeutung. Damit kann ein Gleichungssystem an die gegebenen Voraussetzungen angepaßt werden. Diese Randbedingungen können sein

- Vorgabe der Bauelemente (Konjunktion oder Disjunktion),
- Vorgabe der Eingangsvariable (negiert oder nicht negiert) oder
- Vorgabe der Ausgangsvariable (negiert oder nicht negiert).

Bei der Berücksichtigung solcher Vorgaben wird man oft feststellen, daß nicht immer die Minimallösung realisierbar ist. Im nächsten Beispiel wird auf diese Randbedingungen eingegangen.

Beispiel:

A 4-6: Es soll eine bestehende Gleichung mit Hilfe der Gesetze von De Morgan in eine entsprechende Gleichung umgewandelt werden, die nur noch Konjunktionen enthält.

$Z = (E \cdot \overline{F}) + (A + \overline{B} + C)$

$Z = \overline{\overline{(E \cdot \overline{F}) + (A + \overline{B} + C)}}$ doppelte Negation

$Z = \overline{\overline{(E \cdot \overline{F})} \cdot \overline{(A + \overline{B} + C)}}$ 2. De Morgansche Gesetz

$Z = \overline{(E \cdot \overline{F}) \cdot (\overline{A} \cdot B \cdot \overline{C})}$ 2. De Morgansche Gesetz.

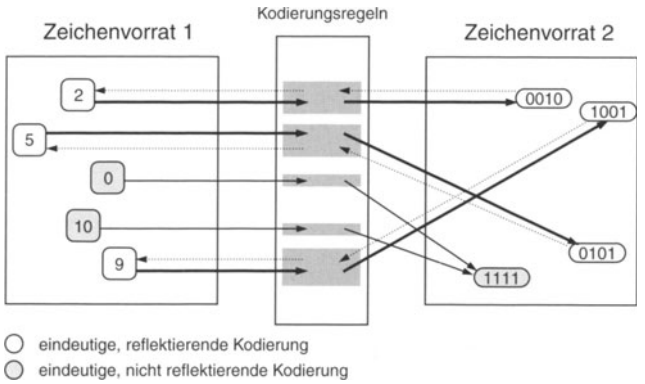


Bild A-25. Grundprinzip der Kodierung.

A 4.4 Kodes

Mit Kodes lassen sich aus einem Zeichenvorrat 1 mit Hilfe von Kodierungsregeln ein Zeichenvorrat 2 erzeugen (Bild A-25). Kodes haben eine *begrenzte Anzahl* von Elementen, die durch *Kodierung* aus einer vorhandenen Zahlenmenge entstehen. Die *Kodierungsregeln* legen dabei fest, wie die *Zielmenge* bei bekannten Ausgangsgrößen auszusehen hat.

Erfolgt die Zuweisung eines Elements aus dem Zeichenvorrat 1 einem Element des Zeichenvorrats 2, so spricht man von einer *eindeutigen, reflektierenden Kodierung,* da aus dem entstandenen Kodewort das Ausgangselement bestimmt werden kann. In Bild A-25 dunkel eingetragen ist auch ein Kodewort, das zwar *eindeutig* aber *nicht reflektierend* ist. Es kann durch zwei Ausgangselemente unabhängig von einander erzeugt werden.

Nicht reflektierende Kodes sind in der Regel eng mit ihrem Anwendungsgebiet verknüpft. Sie haben stets eine *Verkleinerung* des Zeichenvorrats zur Folge und werden deshalb zur *Optimierung* eines bestehenden Zeichenvorrats benutzt. Zur Verdeutlichung sei angenommen, daß in Bild A-25 das Kodewort „1 1 1 1" im Zeichenvorrat 2 beispielsweise eine Anzeigelampe steuert. Diese kann nun im Zeichenvorrat 1 durch die Elemente „0" und „10" aktiviert werden.

Die Mehrzahl der Kodes sind jedoch eindeutige, reflektierende Kodes. Die wichtigsten Vertreter sind in Bild A-26 zusammengestellt.

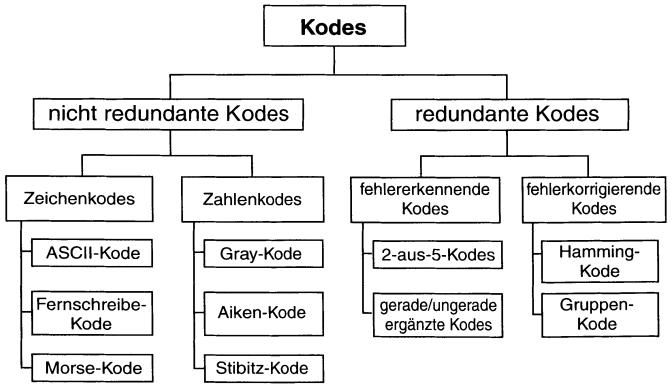


Bild A-26. Übersicht über die wichtigsten Kodes.

Tabelle A-21. ASCII-Tabelle nach CCITT-Kode Nr. 5

niederwertigeres Nibbel				höhenwertigeres Nibbel								
				D_7 = Parity Bit oder ungenutzt								
D_3	D_2	D_1	D_0	D_6	0	0	0	0	1	1	1	1
				D_5	0	0	1	1	0	0	1	1
				D_4	0	1	0	1	0	1	0	1
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE (TC7)	SP	0	@	P	‘	p
0	0	0	1	1	SOH (TC1)	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX (TC2)	DC2	„	2	B	R	b	r
0	0	1	1	3	ETX (TC3)	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT (TC4)	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ (TC5)	NAK (TC8)	%	5	E	U	e	u
0	1	1	0	6	ACK (TC6)	SYN (TC9)	&	6	F	V	f	v
0	1	1	1	7	BEL	ETI (TC10)	’	7	G	W	g	w
1	0	0	0	8	BS (FE0)	CAN	(	8	H	X	h	x
1	0	0	1	9	HT (FE1)	EM	)	9	I	Y	i	y
1	0	1	0	A	LF (FE2)	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT (FE3)	ESC	+	;	K	[	k	{
1	1	0	0	C	FF (FE4)	FS (IS4)	,	<	L	\	l	[
1	1	0	1	D	CR (FE5)	GS (IS3)	-	=	M	]	m	}
1	1	1	0	E	SO	RS (IS2)	.	>	N	^	n	~
1	1	1	1	F	SI	US (IS1)	/	?	O	–	o	DEL

Dabei unterscheidet man zwischen *redundanten* und *nicht redundanten* Kodes. Bei nicht redundanten Kodes wird der Darstellungsbereich des zugrunde liegenden Zahlensystems *maximal* ausgenutzt. Bei redundanten Kodes gibt es auch Kodewörter die *nicht* benutzt sind. Mit deren Hilfe lassen sich *Fehler,* die bei der Kodebildung oder Kodeübertragung entstanden sind, *erkennen* und *korrigieren.* (Abschn. A 4.4.3). Im weiteren werden die wichtigsten Kodes besprochen.

A 4.4.1 ASCII-Kode

Für die Datenübertragung und bei der Kopplung verschiedener digitaler Geräte hat man den *ASCII-Kode* (American Standard Code for Information Interchange) standardisiert. Er besteht aus einem 8-Bit Wort (Byte), bei dem das MSB (Most Significant Bit) das *Paritätsbit* (Prüfbit) darstellt, und die anderen sieben Bits ein Zeichen darstellen. So sind neben den Zahlen 0 bis 9 auch sämtliche Buchstaben (groß und klein) vorhanden sowie eine Reihe von Sonderzeichen. Aus diesem Grund wird der ASCII-Kode auch vorwiegend in der Textverarbeitung zur Erzeugung der Arbeitsdateien verwendet.

Wird das Paritätsbit benutzt, so können mit den sieben verbleibenden Bits 128 Zeichen dargestellt werden (s. Tabelle A-21). Dies ist der Standard ASCII-Zeichensatz nach CCITT Nr. 5. Bei gerader Paritätsprüfung wird das MSB auf „0“ gesetzt, wenn die Anzahl der „1en“ in den verbleibenden 7 Bits gerade ist, andernfalls auf „1“. Bei ungerader Paritätsprüfung ist dies gerade umgekehrt. Die Paritätsprüfung dient vor allem der Fehlererkennung bei der Übertragung von ASCII-Dateien (z.B. bei seriellen Druckerschnittstellen, s. Abschn. A 4.4.3.1). Verzichtet man auf eine Paritätsprüfung, so wird das Prüfbit auf „1“ gesetzt.

Der erweiterte ASCII-Zeichensatz (Tabelle A-22) verzichtet ebenfalls auf die Paritätsprüfung. Er verwendet das höchstwertigste Bit (MSB), um vom Standard ASCII-Zeichensatz auf die Erweiterung umzuschalten (Standard Zeichensatz: MSB = 0, erweiterter Zeichensatz: MSB = 1). Diese Variante kommt vor allem der Textverarbeitung zugute. Dadurch wird Platz geschaffen, um länderspezifischen Buchstaben und Zeichen darzustellen. Für Deutschland sind dies beispielsweise sämtliche *Umlaute* in groß und klein (Ä, ä, Ö, ö, Ü, ü) sowie das scharfe ß (z.B. ß = ASCII-Kode 225, steht auch für Beta). Aber auch

Tabelle A-22. Erweiterter ASCII Zeichensatz

niederwertiges Nibble	höherwertiges Nibble															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 000	DLE 016	SP 032	0 048	@ 064	P 080	` 096	p 112	Ç 128	É 144	á 160	▒ 176	└ 192	╨ 208	∝ 224	≡ 240
1	SOH 001	DC1 017	! 033	1 049	A 065	Q 081	a 097	q 113	ü 129	æ 145	í 161	▒ 177	┴ 193	╤ 209	ß 225	± 241
2	STX 002	DC2 018	" 034	2 050	B 066	R 082	b 098	r 114	é 130	Æ 146	ó 162	▓ 178	┬ 194	╥ 210	Γ 226	≥ 242
3	♥ 003	DC3 019	# 035	3 051	C 067	S 083	c 099	s 115	â 131	ô 147	ú 163	│ 179	├ 195	╙ 211	π 227	≤ 243
4	♦ 004	DC4 020	$ 036	4 052	D 068	T 084	d 100	t 116	ä 132	ö 148	ñ 164	┤ 180	─ 196	╘ 212	Σ 228	⌠ 244
5	♣ 005	§ 021	% 037	5 053	E 069	U 085	e 101	u 117	à 133	ò 149	Ñ 165	╡ 181	┼ 197	╒ 313	σ 229	⌡ 245
6	♠ 006	SYN 022	& 038	6 054	F 070	V 086	f 102	v 118	å 134	û 150	ª 166	╢ 182	╞ 198	╓ 314	µ 230	÷ 246
7	BEL 007	ETB 023	' 039	7 055	G 071	W 087	g 103	w 119	ç 135	ù 151	º 167	╖ 183	╟ 199	╫ 215	τ 231	≈ 247
8	BS 008	CAN 024	(040	8 056	H 072	X 088	h 104	x 120	ê 136	ÿ 152	¿ 168	╕ 184	╚ 200	╪ 216	Φ 232	° 248
9	HT 009	EM 025	) 041	9 057	I 073	Y 089	i 105	y 121	ë 137	Ö 153	⌐ 169	╣ 185	╔ 201	┘ 217	Θ 233	● 249
A	LF 010	SUB 026	* 042	: 058	J 074	Z 090	j 106	z 122	è 138	Ü 154	¬ 170	║ 186	╩ 202	┌ 218	Ω 234	• 250
B	VT 011	ESC 027	+ 043	; 059	K 075	[091	k 107	{ 123	ï 139	ø 155	½ 171	╗ 187	╦ 203	█ 219	δ 235	√ 251
C	FF 012	FS 028	, 044	< 060	L 076	\ 092	l 108	| 124	î 140	£ 156	¼ 172	╝ 188	╠ 204	▄ 220	∞ 236	n 252
D	CR 013	GS 029	- 045	= 061	M 077	] 093	m 109	} 125	ì 141	Ø 157	¡ 173	╜ 189	═ 205	▌ 221	Φ 237	2 253
E	SO 014	RS 030	. 046	> 062	N 078	^ 094	n 110	~ 126	Ä 142	¤ 158	« 174	╛ 190	╬ 206	▐ 222	∈ 238	■ 254
F	SI 015	US 031	/ 047	? 063	O 079	_ 095	o 111	DEL 127	Å 143	ƒ 159	» 175	┐ 191	╧ 207	▀ 223	∩ 239	SP 255

Tabelle A-23. Steuerzeichen im ASCII-Kode

ASCII-Zeichen	englische Bezeichnung	deutsche Bezeichnung
ACK	acknowledge	Rückmeldung
BEL	bell	Klingel
BS	backspace	Rückschritt
CAN	cancel	ungültig
CR	carriage return	Wagenrücklauf
DC	device control	Steuerzeichen für Gerätesteuerung
DEL	delete	löschen
DLE	data link escape	Datenübertragungsumschaltung
EM	end of medium	Ende der Aufzeichnung
ENQ	enquiry	Stationsaufforderung
EOT	end of transmision	Ende der Datenübertragung
ESC	escape	Umschaltung
ETB	end of transmission block	Ende des Datenübertragungsblocks
ETX	end of text	Textende
FE	format effector	Formatsteuerung
FF	format feed	Papiervorschub
FS	file seperator	Hauptgruppen-Trennung
GS	group seperator	Gruppen-Trennung
HT	horizontal tabulation	Horizontal-Tabulator
IS	information separator	Informationstrennung
LF	line feed	Zeilenvorschub
NAK	negativ acknowledge	negative Rückmeldung
NUL	null	Füllzeichen
RS	record seperator	Untergruppen-Trennung
SI	shift in	Rückschaltung
SO	shift out	Dauerumschaltung
SOH	start of heading	Kopfanfang
SP	space	Leerzeichen
STX	start of text	Textanfang
SUB	substitute character	Substitution
SYN	synchronous idle	Synchronisierung
TC	transmission control	Übertragungssteuerung
US	unit separator	Teilgruppen-Trennung
VT	vertical tabulation	Vertikal-Tabulator

spanische, griechische und viele andere Zeichen stehen zur Verfügung. Dies macht deutlich, daß die Erweiterung des ASCII-Satzes unterschiedlich sein kann. Die Steuerzeichen im ASCII-Kode und ihre Bedeutung sind in Tabelle A-23 zusammengestellt.

A 4.4.2 Gray-Kode

Das duale Zahlensystem (Abschn. A 4.1.1), besitzt einen Nachteil: Beim Übergang von einer Dualzahl zur nächsten können sich *mehrere* Bits ändern. Das folgende Beispiel zeigt, daß beim Übergang von der Zahl 7 auf die Zahl 8 vier Bits geändert werden.

```
 . . .
 . . . . .
7: 0 1 1 1
8: 1 0 0 0  Wechsel von 4 Bits!
 . . . . .
 . . .
```

Geschieht dieser Übergang nicht synchron, so können hier Fehler auftreten, die eine Verfälschung bis maximal des zu erkennenden Wertes ermöglichen (in diesem Beispiel, wenn der Übertrag auf das vierte Bit deutlich nach dem

Tabelle A-24. Übersicht über verschiedene Gray Kodes

dezimaler Wert	Gray Kodes: nicht zyklischer Gray Kode von 0 bis 9	zyklischer Gray Kode nach Glixon	zyklischer Gray Kode für die Zahlen 0 bis 15
0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1
2	0 0 1 1	0 0 1 1	0 0 1 1
3	0 0 1 0	0 0 1 0	0 0 1 0
4	0 1 1 0	0 1 1 0	0 1 1 0
5	0 1 1 1	0 1 1 1	0 1 1 1
6	0 1 0 1	0 1 0 1	0 1 0 1
7	0 1 0 0	0 1 0 0	0 1 0 0
8	1 1 0 0	1 1 0 0	1 1 0 0
9	1 1 0 1	1 0 0 0	1 1 0 1
10			1 1 1 1
11			1 1 1 0
12			1 0 1 0
13			1 0 1 1
14			1 0 0 1
15			1 0 0 0

zu null Setzen der ersten drei Bits kommt). Um den Fehler so klein wie möglich zu halten, sollte sich bei *jedem Übergang* nur *ein Bit* ändern. Man spricht dann auch von einem *einschrittigen* Kode, der sich nur in einer Stelle zu seinen benachbarten Zahlen unterscheidet. Realisiert wurde dies im *Gray-Kode* (E. Gray, 1835 bis 1901) nach Tabelle A-24.

Beim Gray-Kode, der die dezimalen Zahlen 0 bis 9 darstellt, ändert sich von einer Zahl zur nächsten stets nur ein Bit. In dieser Darstellung ist er die Basis für den erweiterten Gray-Kode, der alle 16 möglichen Kodeworte ausnutzt (Tabelle A-24, rechte Spalte). Nicht abgedeckt ist bei der Darstellung dezimaler Zahlen der Übergang von 9 auf 0: hier wechseln 3 Bits. Damit ist dieser Gray-Kode nicht zyklisch. Durch eine kleine Modifikation nach *Glixon* konnte jedoch auch dieser Übergang einschrittig gemacht werden, so daß dieser Gray-Kode nun auch für die Darstellung der dezimalen Zahlen 0 bis 9 zyklisch ist. In Tabelle A-24 ist diese Änderung grau unterlegt.

Der erweiterte Gray-Kode nutzt alle 16 Kodeworte aus. Er ist vom Basis-Kode (linke Spalte in Tabelle A-24) ausgehend grundsätzlich *zyklisch*.

Die Bildung des Gray-Kodes erfolgt aus den Dualzahlen (Abschn. A 4.1.1). Dabei wird die Ausgangszahl um eine Stelle nach links geschoben und anschließend mit sich selbst *modulo 2 addiert (Shift-Add-Verfahren)*. Durch Zurückschieben des Ergebnisses um eine Stelle nach rechts (Wegfall der letzten Stelle) erhält man den Gray-Kode. Bild A-27 zeigt dieses Prinzip an Hand der dualen Zahlen 1 bis 7.

Rechnertechnisch kann dies sowohl von einem Programm *(Software)* als auch durch ein Rechenwerk *(Hardware)* ausgeführt werden. Ein Beispiel für die wesentlich schnellere Hardware-Lösung sind Analog-Digitalwandler. Beide Vorgehensweisen sind in Bild A-28 gegenübergestellt.

Der Gray-Kode findet seine Anwendung sowohl bei linearen Wegmessungen, als auch bei der Bestimmung von Drehwinkeln. Dabei wird der Gray-Kode auf eine kreisförmige Kodescheibe von außen nach innen aufgetragen. Hier ist auf jeden Fall ein zyklischer Gray-Kode von Vorteil (Bild A-29).

A 4.4.3 Redundante Kodes

Redundante Kodes werden ebenfalls sehr häufig bei der Datenübertragung eingesetzt. Wie beim ASCII-Kode das Paritätsbit zur Fehlererkennung herangezogen werden kann (es ist ebenfalls redun-

Dezimalzahl	0	1	2	3	4	5	6	7
Dualzahl	000	001	010	011	100	101	110	111
Links-Shift	000←	001←	010←	011←	100←	101←	110←	111←
modulo 2 Addition	0000	0011	0110	0101	1100	1111	1010	1001
Rechts-Shift	→000	→001	→011	→010	→110	→111	→101	→100
Gray-Kode	**000**	**001**	**011**	**010**	**110**	**111**	**101**	**100**

Bild A-27. Gewinnung des Gray-Kodes aus Dualzahlen.

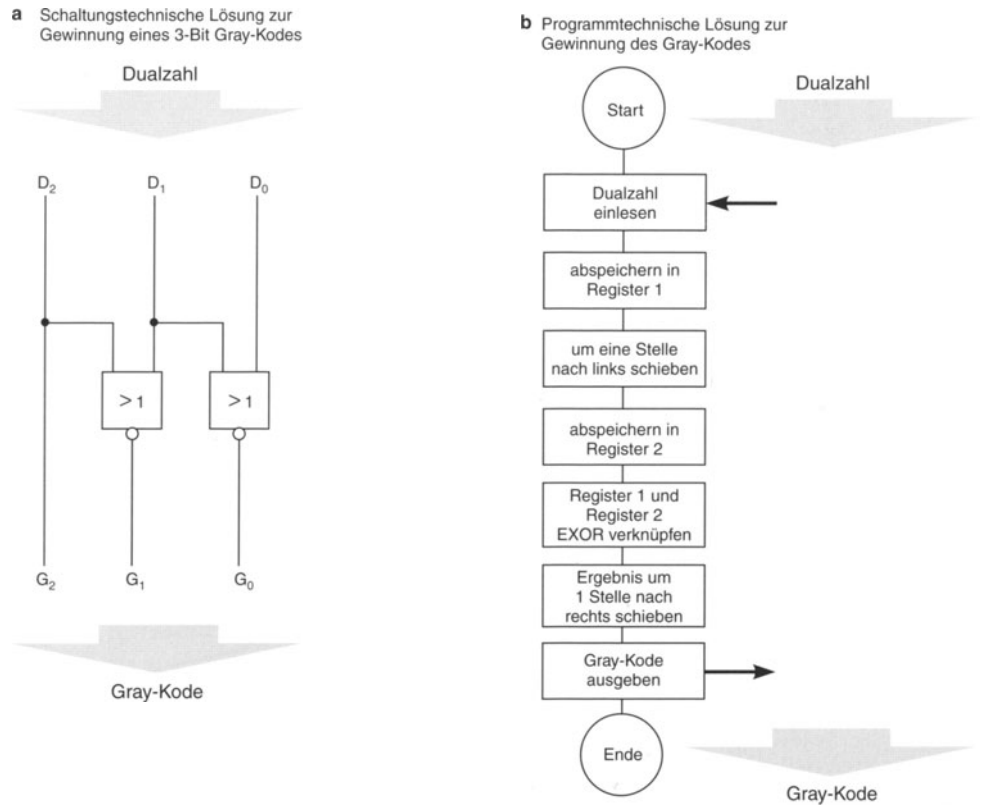


Bild A-28. Realisierung des Shift-Add-Verfahrens.

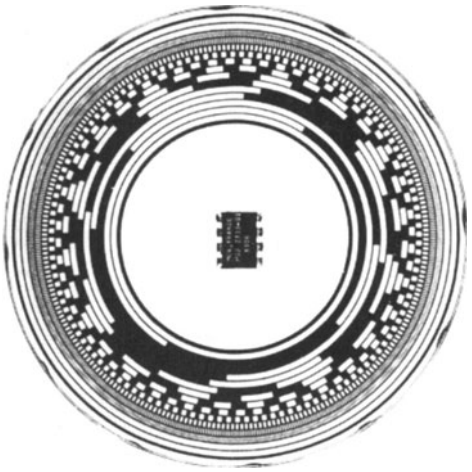


Bild A-29. Kreisteilung einer Winkelkodierscheibe im Gray Kode.

dant, da es zum *Informationsinhalt* nicht beiträgt), so sind diese redundanten Kodes speziell dazu ausgelegt, Fehler zu *erkennen* und gegebenenfalls zu *korrigieren.* Letzteres ist nur dann möglich, wenn die Redundanz auf die Fehlerstelle aufmerksam macht.

> Wird in einem Kode mehr als nur die Information übertragen, so ist dieser *redundant.* Diese Redundanz kann dazu verwendet werden, Fehler zu erkennen und gegebenenfalls zu korrigieren.

Die Redundanz sollte in einem sinnvollen Verhältnis zur übertragenen Information stehen. Dies hat für die Erkennung und Korrektur von Fehlern zur Entwicklung bestimmter Kodes geführt.

Grundsätzliche Verfahren zur Kodesicherung

Zur *Erkennung* oder *Korrektur* eines Fehlers ist Redundanz notwendig. Will man einen Fehler nur erkennen, so besteht eine einfache Möglichkeit darin, die übertragene Information zu wiederholen (50%ige Redundanz). Durch einfachen Vergleich ergibt sich bei richtiger Übertragung Übereinstimmung, im anderen Fall eine Fehlermeldung. Eine Korrektur ist damit nicht möglich.

Bei fehlerkorrigierenden Kodes muß die Redundanz noch weiter erhöht werden. Eine Möglichkeit besteht darin, die Information insgesamt dreimal zu senden (67%ige Redundanz). Dabei entstehen mit sehr hoher Wahrscheinlichkeit mindestens zwei gleiche Datenworte, die übereinstimmen und die richtige Information beinhalten.

Die oben aufgeführten Verfahren zur Fehlererkennung und Fehlerkorrektur lassen sich auf jegliche Art der Übertragung anwenden, sind aber nicht besonders effizient. Die Theorie der fehlererkennenden und korrigierenden Kodes geht von der Tatsache aus, daß bei einem voll ausgenutzten Kode *ein Fehler* in einem Kodewort ein *neues Kodewort* erzeugt. Also muß sich ein Kode, bei dem ein Fehler erkannt werden soll, mindestens in *zwei* Stellen des Kodewortes unterscheiden. Zwischen den benutzten Kodeworten liegen also *unbenutzte,* die auf einen Fehler hinweisen. Dieser Abstand wird auch als *Hammingdistanz* $d_{\min}$ bezeichnet, die auf den Grad der erkennbaren und korrigierbaren Fehler zurückschließen läßt. Eine Hammingdistanz von beispielsweise $d_{\min} = 2$ liegt dann vor, wenn sich das nächste Kodewort in zwei Stellen unterscheidet.

A 4.4.3.1 Fehlererkennende Kodes

Zur einfachen Fehlererkennung muß wenigstens *ein* Bit spendiert werden. Am Beispiel des ASCII-Kodes ist dies das Paritäts-Bit D7. Durch ein solches Paritäts-Bit läßt sich jede Kodierung zur Fehlererkennung ergänzen. Am Beispiel der Dualzahlen von 0 bis 15 soll dies gezeigt werden (Tabelle A-25).

Das Paritäts-Bit D4 (auch Prüfbit genannt) ist die *Quersumme* der Bits D0 bis D3. Bei einer ungeraden Anzahl von Einsen wird das Paritäts-Bit „1", bei einer gerader Anzahl „0". So spricht man auch von einer *geraden Ergänzung* durch das Paritäts-Bit (engl.: even parity), im anderen Fall von einer *ungeraden Ergänzung* (odd parity).

Auf der Empfangsseite wird die Quersumme über alle fünf Bits gebildet, D0 bis D3 und Paritäts-Bit D4. Wurde der Kode richtig übertragen, so ergibt die *Quersumme* stets *null.*

> Ein Kode mit Paritätsprüfung wurde dann richtig übertragen, wenn seine Quersumme am Empfangsort bei gerader Paritätsprüfung null ergibt.

In obigem Beispiel (Tabelle A-25) wurden die Dualzahlen 0 bis 15 durch ein Prüfbit ergänzt. Es entstand so ein *dualergänzter Kode,* der statt *vier* nunmehr *fünf* Stellen besitzt.

Tabelle A-25. Dualzahlen mit Paritäts-Bit

Paritäts-Bit	Dualzahlen				Quersumme
	D4	D3	D2	D1	D0
0	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
0	0	0	1	1	0
1	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
1	0	1	1	1	0
1	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
1	1	0	1	1	0
0	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
0	1	1	1	1	0

Es gibt noch eine ganze Reihe fünfstelliger Kodes, wobei die *2-aus-5-Kodes* eine besondere Bedeutung haben. Wie sich auch bereits aus der Bezeichnung ablesen läßt, handelt es sich dabei um fünfstellige Kodes, bei denen stets *zwei Stellen* auf „1", die restlichen auf „0" sind. Die Fehlererkennung beruht bei diesen Kodes ebenfalls auf der *Geradzahligkeitsprüfung:* bei richtigem Empfang der Datenworte muß die Quersumme stets null ergeben, da stets zwei Bits gesetzt sind. Wird während der Übertragung ein Bit verfälscht, so entsteht in jedem Fall eine ungerade Anzahl von Einsen, die erkannt wird.

Beispiele für 2-aus-5-Kode sind der *Walking-Kode* und der *7-4-2-1-0-Kode.* Beide Kodes sind in der Tabelle A-26 gegenübergestellt. Beim Walking-Kode werden zwei Bit-Paare (in Tabelle A-26 eingekreist) beim Übergang auf die nächste Zahl um zwei Stellen weitergeschoben. Es entsteht so der Eindruck, daß diese Paare (grau hinterlegt) durch die Zahlen 0 bis 9 durchlaufen (engl.: walking).

Der 7-4-2-1-0-Kode (Tabelle A-26, rechte Hälfte) soll an dieser Stelle als Vertreter weiterer 2-aus-5-Kodes stehen, deren Kodierung sich aus der *Wertigkeit* der benutzten Stellen ergibt. In diesem Fall besitzen die einzelnen Bits die Wertigkeit 7, 4, 2, 1 und 0. Durch Setzen von zwei Bits lassen sich alle Zahlen von 1 bis 9 darstellen. Das Kodewort für null stellt eine Ausnahme dar und ergibt sich aus dem von den Zahlen 1 bis 9 nicht genutzten Kodewort.

A 4.4.3.2 Fehlerkorrigierende Kodes

Sollen Fehler nicht nur *erkannt,* sondern auch *korrigiert* werden, so muß die Redundanz weiter erhöht werden. Ein Zusammenhang zwischen der Redundanz und der möglichen Zahl der erkennbaren und korrigierbaren Fehler hat *Hamming* (R. Hamming) in seinen Gleichungen festgelegt. Der Abstand zweier benachbarter Kodewörter im Koderaum wird auch als *Hammingdistanz* $d_{\min}$ bezeichnet.

Für $d_{\min} = 1$ bedeutet dies, daß sich die Kodewörter nur in einer Stelle unterscheiden, wie beispielsweise der *Gray-Kode.* Bei $d_{\min} = 2$ unterscheiden sich die Kodewörter in zwei Stellen, wie dies bei den *2-aus-5-Kodes* der Fall ist.

Bei $d_{\min} = 1$ kann ein Fehler weder erkannt noch korrigiert werden, da eine Verfälschung des Kodewortes immer zu einem *neuen gültigen Kodewort* führt. Wird hingegen ein Kode mit $d_{\min} = 2$ in einer Stelle gestört (man spricht hier auch von einem Fehler mit dem *Gewicht 1),* so führt dies stets zu einem *ungültigen* Kodewort, so daß dieser Fehler erkannt wird. Deshalb gilt:

> Zur Erkennung eines einfachen Fehlers ist mindestens eine Hammingdistanz von $d_{\min} = 2$ erforderlich.

Erhöht man die Hammingdistanz, so können entsprechend des erweiterten Koderaums auch Fehler mit einem höheren Gewicht erkannt werden. Für die maximale Anzahl $F_{E\max}$ der erkennbaren Fehler gilt:

$$F_{Emax} = d_{\min} - 1. \qquad \text{(A-36)}$$

Die Korrektur eines Kodes ist möglich, wenn die fehlerhafte Kodezahl eindeutig einer gültigen Zahl im Koderaum zugeordnet werden kann. Der notwendige *Korrekturradius* r_k des *Korrekturraumes* ergibt sich nach Gl. (A-37) zu:

$$r_k < \frac{d_{\min}}{2}. \qquad \text{(A-37)}$$

Zur Korrektur eines Fehlers ist also mindestens eine Hammingdistanz von $d_{\min} = 3$ notwendig, da sonst der Korrekturradius kleiner als 1 wird.

Tabelle A-26. 2-aus-5-Kodes

	Walking-Kode					7-4-2-1-0-Kode				
dezimaler Wert	D4	D3	D2	D1	D0	D4	D3	D2	D1	D0
0	0	0	0	1	1	1	1	0	0	0
1	0	0	1	0	1	0	0	0	1	1
2	0	0	1	1	0	0	0	1	0	1
3	0	1	0	1	0	0	0	1	1	0
4	0	1	1	0	0	0	1	0	0	1
5	1	0	1	0	0	0	1	0	1	0
6	1	1	0	0	0	0	1	1	0	0
7	0	1	0	0	1	1	0	0	0	1
8	1	0	0	0	1	1	0	0	1	0
9	1	0	0	1	0	1	0	1	0	0
						7	4	2	1	0
						Wertigkeit der Stellen				

Bild A-30 zeigt zwei Kodewörter mit einer Hammingdistanz von $d_{min} = 3$ und den dazugehörigen Korrekturraum.

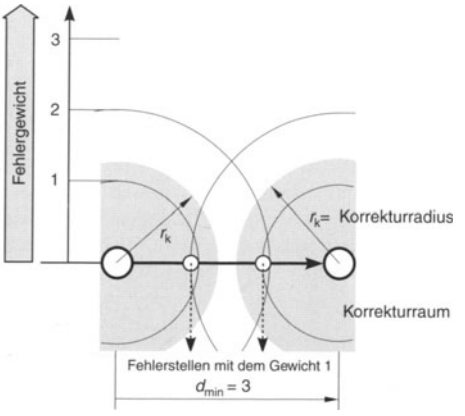


Bild A-30. Korrekturraum und Korrekturradius bei einer Hammingdistanz von $d_{min} = 3$.

Tritt bei dem Beispiel in Bild A-30 ein Fehler mit dem Gewicht 1 auf (Verfälschung des Kodes in einer Stelle), so wird er richtig zum nächsten Kodewort hin korrigiert. Er liegt innerhalb des durch den Korrekturradius beschriebenen Korrekturraums. Ein Doppelfehler (Gewicht = 2) führt hingegen stets zu einer falschen Korrektur, da der Fehler nicht im gültigen Korrekturraum liegt und in den Einzugsbereich eines anderen gültigen Kodewortes fällt.

Bild A-31 zeigt den Korrekturraum für die Hammingdistanz $d_{min} = 4$. Hier werden in einem Kodewort maximal bis zu drei Fehler erkannt.

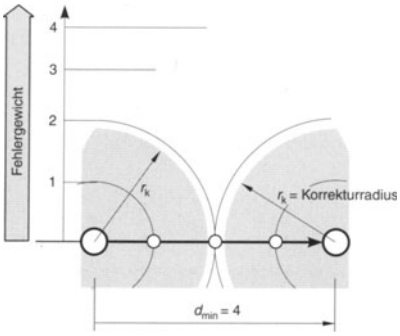


Bild A-31. Korrekturraum und Korrekturradius bei einer Hammingdistanz von $d_{min} = 4$.

Richtig korrigiert werden können jedoch ebenfalls nur einfache Fehler, da ein Doppelfehler auf der Schnittlinie beider Korrekturräume liegt und deshalb nicht mehr eindeutig zugeordnet werden

kann. Der zum Korrekturraum gehörende Korrekturradius r_k ist demnach stets kleiner als die halbe Hammingdistanz. Bild A-31 veranschaulicht die Aussage der Gl. (A-37).

Die maximale Anzahl der *korrigierbaren Fehler* $F_{K\max}$ wird durch den Korrekturradius bestimmt und läßt sich aus Gl. (A-37) und Bild A-31 direkt entnehmen:

$$F_{K\max} < \frac{d_{\min}}{2}. \qquad (A-38)$$

Da $F_{K\max}$ nur ganze Zahlen annehmen kann, läßt sich aus obiger Ungleichung für $d_{\min}$ die Gl. (A-39) ableiten.

$$d_{\min} = 2 \cdot F_{K\max} + 1. \qquad (A-39)$$

Die Anzahl der erkennbaren Fehler (F_E), wenn nicht alle korrigierbar sind oder wenn nicht die maximale Anzahl von Korrekturen (also nur F_K) durchgeführt werden soll, ergibt sich nach:

$$F_E = d_{\min} - 2 \cdot F_K - 1. \qquad (A-40)$$

Dabei gilt:

$$F_E \leq F_{E\max}, \qquad (A-41)$$
$$F_K \leq F_{K\max}. \qquad (A-42)$$

Beispiel:

A 4-7: Zur Veranschaulichung der Zusammenhänge der Gleichungen (A-36) bis (A-42) soll ein Kode mit einer Hammingdistanz von $d_{\min} = 5$ angenommen werden. Nach Gl. (A-36) errechnet sich die maximale Anzahl der erkennbaren Fehler (wenn keine korrigiert werden) zu $F_{E\max} = d_{\min} - 1 = 4$. Das bedeutet, daß alle Kodewörter, die zwischen zwei gültigen Kodewörtern liegen, als Fehler erkannt werden. Es können also Fehler mit einem Gewicht von *4* noch erkannt werden. Der Korrekturradius ist dabei gleich null.

Bei Korrektur erhält man nach Umstellen von Gl. (A-39) die maximale Anzahl der korrigierbaren Fehler: $F_{K\max} = (d_{\min} - 1)/2 = 2$. Darüber hinaus können nach Gl. (A-40) keine weiteren Fehler F_E mehr erkannt werden, da $F_E = d_{\min} - 2 \cdot F_K - 1 = 0, bei F_K = F_{K\max}$. Bei einer Hammingdistanz von $d_{\min} = 5$ können also maximal Fehler mit einem Gewicht von 2 richtig korrigiert werden. Fehler mit einem Gewicht von beispielsweise 3 würden in einen anderen Korrekturraum fallen und deshalb falsch korrigiert werden (Bild A-31). Soll die Korrektur nur bei einem Fehlergewicht von 1 erfolgen (Einschränkung des Korrekturraums), so können dafür weitere Fehler erkannt werden: $F_E = d_{\min} - 2 \cdot F_K - 1 = 2, bei F_K = 1$. Bei diesen erkannten Fehlern handelt es sich um Fehler mit dem Gewicht 2 und 3. Durch die Einschränkung des Korrekturraums wird also Platz geschaffen, um höherwertigere Fehler zu erkennen.

Um diese Anforderungen an die Fehlererkennung und -korrektur bei den bereits bekannten Kodes anzuwenden, müssen entsprechend *Kontrollstellen k* zu den vorhandenen *Nutzbits m* hinzugefügt werden. Man erhält so ein *Kodewort N,* das aus

$$N = m + k \qquad (A-43)$$

Stellen besteht. Der so entstandene Hamming-Kode gehört damit zu den *Gruppenkodes,* da er sich aus einer *Informationsgruppe* (m) und einer *Kontrollgruppe* (k) zusammensetzt.

Wieviele Kontrollstellen an einen Kode angefügt werden müssen, hängt von der Hammingdistanz $d_{\min}$ ab, und damit von dem *Gewicht* der *korrigierbaren* Fehler. Sollen beispielsweise alle einfachen Fehler korrigiert werden, so ist $d_{\min} = 3$ (Bild A-30). Das bedeutet, daß sich ein Kodewort beim Übergang auf das nächste in drei Stellen unterscheiden muß. Für einen *Ein-Bit-Kode* (m = 1) müssen demnach 2 Kontrollbits hinzugefügt werden, um diese Bedingung zu erfüllen. Der Hammingkode besteht dann aus N = 3 Stellen. Aber bereits bei einem Kode mit m = 2 reichen die beiden Korrekturstellen nicht mehr aus: k muß hier 3 sein (Tabelle A-27). Der Zusammenhang ergibt sich allgemein für eine Hammingdistanz von $d_{\min} = 3$ zu:

$$m = 2^k - k - 1 \qquad (A-44)$$

Für eine Hammingdistanz von $d_{\min} = 4$ gilt:

$$m = 2^{k-1} - k. \qquad (A-45)$$

In Tabelle A-27 sind die Nutzbits und die notwendige Anzahl der Korrekturstellen bei den Hammingdistanzen $d_{\min} = 3$ und $d_{\min} = 4$ gegenübergestellt (nach Gl. (A-44) und Gl. (A-45)), ebenso die daraus resultierende Gesamtwortbreite.

Tabelle A-27 zeigt deutlich, daß die Kodesicherung bei großen Wortbreiten durch verhältnismäßig wenige Kontrollstellen erreicht werden kann. Bei 57 Nutzbits sind lediglich 6 Kontrollstellen notwendig, was eine Redundanz von weniger als 10% bedeutet. Zur Sicherung eines Halbbytes (4 Bit) ist dagegen eine Redundanz von annähernd 50% notwendig.

Tabelle A-27. Zusammenhang zwischen Nutzbits, Kontrollbits und Wortbreite nach Hamming

$d_{min} = 3$			$d_{min} = 4$		
Nutzbits m	Kontrollbits k	Wortbreite N	Nutzbits m	Kontrollbits k	Wortbreite N
1	2	3	1	3	4
2	3	5	2	4	6
3	3	6	3	4	7
4	3	7	4	4	8
5	4	9	5	5	10
6	4	10	6	5	11
7	4	11	7	5	12
8	4	12	8	5	13
9	4	13	9	5	14
10	4	14	10	5	15
11	4	15	11	5	16
26	5	31	26	6	32
57	6	63	57	7	64
120	7	127	120	8	128

Wird in die Gleichungen (A-44) und (A-45) die Hammingdistanz eingearbeitet, so ergibt sich Gl. (A-46) zu:

$$m = 2^{k-(d_{\min}-3)} - [k - (d_{\min} - 3)] - 1 \quad \text{(A-46)}$$

A 4.4.3.3 Redundanzoptimierte Kodes

Redundanzoptimierte Kodes stellen eine weitere wichtige Gruppe von Kodes dar. Ihre *Wortlänge* wird mit Hilfe unterschiedlicher Wahrscheinlichkeiten für das Auftreten eines Kodewortes optimiert. Die Kodes in den Abschn. A 4.4.1 bis A 4.4.3 weisen für jedes Kodeelement *Zeichenketten gleicher Länge* auf. *Statistisch optimierte* Kode hingegen haben *variable Zeichenketten,* die von der Auftrittswahrscheinlichkeit der Kodeworte abhängig sind. Bei gleichverteilter Wahrscheinlichkeit kann damit natürlich nichts gewonnen werden.

Grundsätzlich wird bei redundanzmindernden Kodes dem am *häufigsten* auftretenden Kodewort die *kürzeste Zeichenkette* zugewiesen und den am *seltensten* auftretenden Quellwörtern die *längsten* Kodewörter. Am Beispiel unseres Alphabetes würde man den Buchstaben E und D (häufiges Auftreten) kurze Zeichenketten und den Buchstaben X und Y (sehr seltenes Auftreten) lange Zeichenketten zuweisen. Damit wird bei den allermeisten Übertragungen eines solchen Kodes eine deutliche Verringerung der Nachrichtenlänge erreicht, ohne daß der Nachrichteninhalt vermindert wird. Anwendung hat dieses Prinzip vor allem in der *Kommunikationstechnik* wie *Bildtelefon* und *Faxgeräte* gefunden.

Bekanntestes Beispiel für einen redundanzoptimierten Kode ist der *Morse-Kode.* Dabei handelt es sich um einen Kode, dessen statistische Verteilung nach obigen Kriterien empirisch ermittelt wurde. Bild A-32 zeigt den zum Morse-Kode gehörenden Kodebaum.

Beim Morse-Kode handelt es sich um einen *ternären Kode,* also einem Kode, der aus *3 Elementen* besteht *(Punkt, Strich* und *Zwischenraum).* Der Zwischenraum muß dabei als selbständiges Element betrachtet werden, da aus ihm beispielsweise auch der Wortzwischenraum erzeugt werden muß. Dementsprechend besitzt der Kodebaum in Bild A-32 auch 3 Richtungen, die miteinander verknüpft den entsprechenden Kode ergeben. Der Wahrscheinlichkeit im Alphabet entsprechend sind die am häufigsten auftretenden Buchstaben E und T mit dem jeweilig kürzesten Kodewort kodiert: 1 Punkt bzw. 1 Strich. Der Zwischenraum zwischen den Buchstaben wird als häufigstes Element überhaupt auch einstellig behandelt und automatisch nach jedem Buchstaben angehängt. Der Wortzwischenraum wird aus drei Zwischenräumen gebildet (hier wird bereits deutlich, daß der Morsekode an manchen Stellen eben-

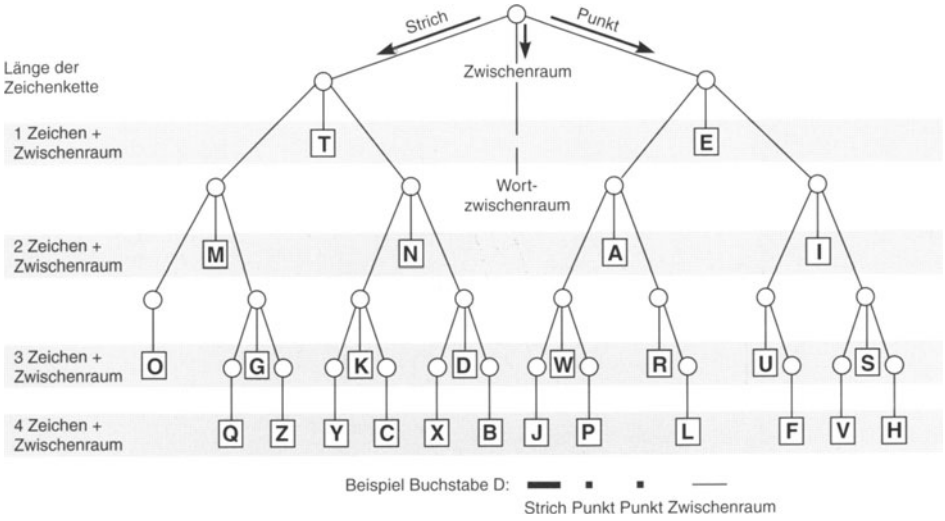


Bild A-32. Kodebaum für den Morsekode.

falls *redundant* ist, da ein Kodewort mit 2 Zwischenräumen nicht existiert).

Eine vollständige Ausnutzung des Kodebaumes ist dann gegeben, wenn aus *jedem* Knotenpunkt des Baumes *drei Zweige* hervorgehen, entsprechend dem ternären Kode. Bei allen Knoten vor den Endpunkten ist dies nicht der Fall, was auf die nicht vollständige Ausnutzung des Kodebaumes hinweist.

Die günstigste Kodewortlänge kann bei gegebenen *Wahrscheinlichkeiten* p_k durch ein *analytisches Verfahren* nach Shannon ermittelt werden. Unvollständige Kodebäume können dabei in einem zweiten Schritt so korrigiert werden, so daß sich ein vollständiger Kodebaum ergibt.

Für binäre Kodierungen eignet sich eine *halbgrafische Methode* nach *Shannon-Fano.* Sie liefert stets einen *vollständigen* Kodebaum. Voraussetzung ist, daß die Wahrscheinlichkeiten p_n der Quellensymbole bekannt sein muß. Sie werden in einer Tabelle mit fallender Wahrscheinlichkeit eingetragen. In einer zweiten Spalte werden die einzelnen Wahrscheinlichkeiten in umgekehrter Reihenfolge, also von unten nach oben, zur Summe sk addiert. Da die Kodeworte in der Umgebung von sk = 0,5 die höchste Wahrscheinlichkeit besitzen, werden ihnen die einstelligen Kodeworte „1" und „0" zugeordnet. Die entstandenen Untermengen links und rechts von sk = 0,5 werden in gleicher Weise behandelt: Sie gilt es in zwei möglichst gleiche Teile zu zerlegen, denen anschließend wieder „1" und „0" zugeordnet werden. Dies ist die zweite Stelle des Kodewortes. Die Aufteilung der *Reststämme* erfolgt solange, bis schließlich jede Untermenge gerade noch aus der maximalen Anzahl zuordenbarer Elemente (bei binärer Kodierung wie oben beschrieben sind das zwei) oder weniger besteht. Bild A-33 zeigt einen nach dieser Methode entwickelten Kodebaum.

Redundanzoptimierte Kodes nach Shannon-Fano bieten einen *höchstmöglichen Informationsgehalt* bezogen auf die Kodelänge. Sehr umfangreiche Telegramme können damit schnell und richtig übertragen werden. Solche Telegramme sind jedoch aufgrund der fehlenden oder sehr geringen Redundanz *besonders störanfällig.* Eine Verfälschung führt immer zu einem anderen gültigen Kodewort, Störungen der Kodelänge kann sogar die gesamte Information zerstören, da die *synchronen* Dekodiergeräte auf der Empfangsseite nicht mehr oder nicht gültig einrasten. Für die sichere Übertragung von Daten werden deshalb auch redundanzoptimierte Kodes mit *zusätzlichen Kontrollbits* ausgestattet, um so eine *Fehlererkennung* bzw. *-korrektur* zu ermöglichen. Der Vorteil einer schnellen Datenübertragung wird dabei nur geringfügig verkleinert.

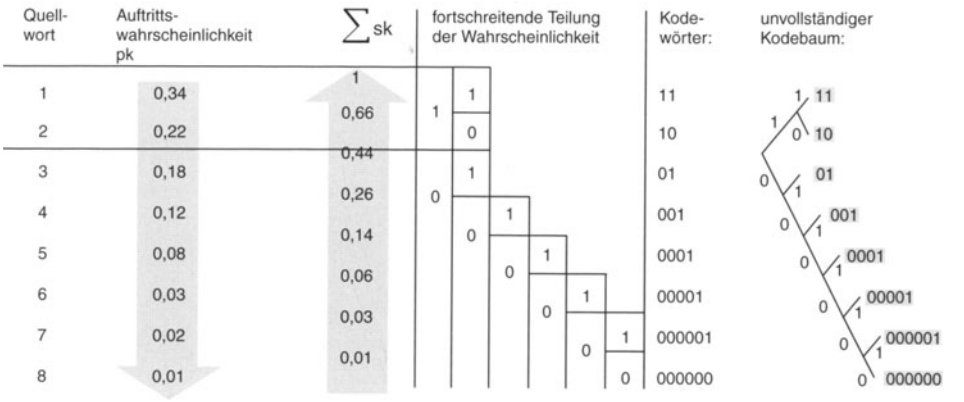


Bild A-33. Vollständiger Kodebaum nach Shannon-Fano.

Zur Übung:

ÜA 4-1: Zu welcher Basis werden folgende Zahlensysteme gerechnet: Oktalsystem, Hexadezimalsystem, Dualsystem und binäres Zahlensystem?

ÜA 4-2: Es soll ein *nonales* Zahlensystem eingeführt werden.

a) welche Elemente besitzt die Basis des Nonalsystems?
b) welche sinnvolle Kennzeichnung des *Nonalsystems* ist möglich?
c) bei welcher Zahl erfolgt der Übertrag?
d) wie werden die dezimalen Zahlen 9, 10, 23 und 100 dargestellt?

ÜA 4-3: Auf einem Prozessordatenbus werden die Werte 0100.0011.1001.1111 (D15 - D0) gemessen.

a) welchem dezimalen Wert entspricht dies?
b) welcher hexadezimale Wert wird dargestellt?
c) wenn das MSB ein Vorzeichenbit ist, handelt es sich um eine positive oder negative Zahl?

ÜA 4-4: Folgende Zahlen in der VBD und ZK Darstellung sollen negiert werden:

a) 00000
b) 01111
c) 0 0100 0000 1100 1010
d) 0 0110 0111 1000 0101
e) $02C_H$

ÜA 4-5: Im Zweierkomplement werden negative Zahlen dargestellt. Es soll das Zweierkomplement folgender Zahlen gebildet und der dezimale Wert angegeben werden:

a) 0000 0000
b) 0111 1111
c) 0101 0101
d) 0011 0011

ÜA 4-6: Warum kann eine normalisierte Mantisse einer positiven Zahl nicht kleiner als 0.5_D werden?

ÜA 4-7: Man vereinfache mit Hilfe der Booleschen Algebra folgenden Ausdruck: $(A+B)\cdot(A+C)$. Welche Regeln wurden angewandt?

ÜA 4-8: Welche Gesetze beweisen den Zusammenhang zwischen Disjunktion und Konjunktion?

ÜA 4-9: Wann wird ein Zeichenvorrat als Kode bezeichnet?

ÜA 4-10:

a) Was versteht man unter einem einschrittigen Kode?
b) Nennen Sie ein Beispiel
c) Welche Vorteile werden ausgenutzt?
d) Welche Hammingdistanz hat ein einschrittiger Kode?

ÜA 4-11:

a) welche Hammingdistanz ist zur Erkennung von Fehlern notwendig?

b) welche Hammingdistanz ist zur Korrektur von Fehlern notwendig?

ÜA 4-12: Ein Kode besitzt die Hammingdistanz 6

a) Wieviele Fehler können maximal erkannt werden?
b) Wieviele Fehler können maximal korrigiert werden?
c) Wieviele können dann unter b) noch erkannt werden?

Weiterführende Literatur

Beuth, K.: Elektronik 4, Digitaltechnik. Würzburg: Vogel Verlag.
Philippow, E.: Taschenbuch der Elektrotechnik, Bd. 1 u. 2. München: Hanser Verlag.
Philippow,E.: Grundlagen der Elektrotechnik. Leipzig: Akadem. Verlagsges. Geest & Portig.

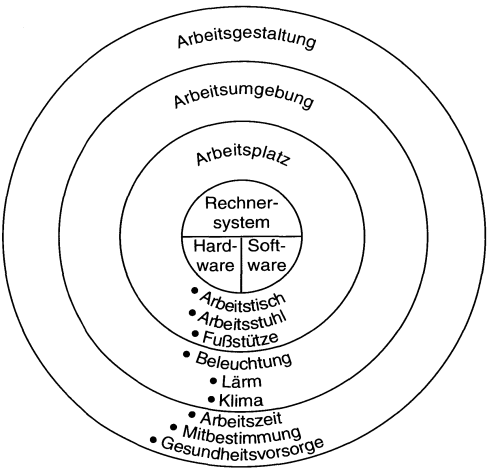


Bild A-34. Überblick über Software-Ergonomie.

A 5 Ergonomie

Der Computer am Arbeitsplatz ist heute bereits eine Selbstverständlichkeit. In einer Zeit starken Wettbewerbsdrucks und eines schnellen Marktwandels werden nur diese Unternehmen eine Überlebenschance haben, die leistungsfähige und motivierte Mitarbeiter besitzen, die diese modernen Technologien einsetzen. Deshalb erhält eine gesundheitsgerechte Auswahl, Anordnung und Betrieb von Bildschirmarbeitsplätzen eine besondere Bedeutung.

A 5.1 Gesamtkonzeption

Wie Bild A-34 zeigt, darf sich die *Ergonomie* nicht auf den Rechner (Hardware) und den Arbeitsplatz beschränken. Es geht letztlich darum, alle Prozesse der Datenverarbeitung *dem Menschen anzupassen (kognitive* Ergonomie). Dazu gehört unter anderem auch eine sinnvolle Zeiteinteilung für die Dauer der Arbeit an Bildschirmarbeitsplätzen, eine benutzergerechte Software, die Möglichkeit der Mitarbeiter, ihre Arbeitsplätze mitgestalten zu lassen und eine gesundheitliche Aufklärung. In Bild A-34 sind diese *vier Kreise* der Ergonomie zu erkennen:

1. Kreis: Rechnersystem (Hard- und Software)

Bei der Hardware spielen für den Bildschirm beispielsweise Fragen der Zeichengröße und -schärfe oder die Flimmerfreiheit eine Rolle. Für die Software ist beispielsweise auf eine gute Benutzerführung zu achten.

2. Kreis: Arbeitsplatz

Besondere Anforderungen sind an den Arbeitsplatz zu richten: an den Arbeitstisch, den Arbeitsstuhl und die Fußstütze.

3. Kreis: Arbeitsumgebung

Hierbei handelt es sich um die *visuelle* (Beleuchtung), die *akustische* (Lärm) und die *klimatische* Umgebung.

4. Kreis: Arbeitsgestaltung

In diesen Kreis fallen unterschiedliche Bereiche: beispielsweise die Art der Aufgaben, die Zeiteinteilung, die Einbeziehung der Mitarbeiter bei der Raumgestaltung oder die Gesundheitsvorsorge.

A 5.2 Rechnersystem

Ein Rechnersystem besteht aus dem eigentlichen Gerät *(Hardware)* und den Programmen *(Software).*

A 5.2.1 Hardware

Die Hardware setzt sich aus folgenden Teilen zusammen:

- Bildschirm,
- Eingabe von Information,
- Rechner und
- Ausgabegeräte.

Bildschirm

Bildschirme werden beurteilt nach ihrer

- Anzeigenart,
- Zeichengröße,
- Zeichenschärfe,
- Zeichenkontrast,
- Zeichengestalt,
- Zeilenabstand,
- Flimmern und
- Reflexe.

a) Anzeigenart

Von der Technik her unterscheidet man Bildschirmanzeigen mit der *Kathodenstrahlröhre* (CRT: Cathode Ray Tube), die vornehmlich im Bürobereich eingesetzt werden, mit *Flüssigkristallanzeige* (LCD: Liquid Crystal Device), die im mobilen Bereich Verwendung finden, oder *Plasmabildschirme* für besonders feingliedrige und scharfe Zeichnungen. Angezeigt werden können entweder *helle Zeichen* auf dunklem Grund oder *dunkle Zeichen* auf hellem Grund. Ferner sind noch *einfarbige* (monochrome) und *farbige* Bildschirme zu unterscheiden. Bei Farbbildschirmen müssen drei Farbpunkte (rot, grün, blau) gemischt werden.

b) Zeichengröße

Die Zeichengröße muß nach der Verteilung der *Sehschärfe* im Gesichtsfeld ausgesucht werden. In Abhängigkeit vom Sehabstand e wird folgende Gleichung verwendet:

$$0{,}052\ \mathrm{e} < \text{Zeichengröße in mm} > 0{,}07\ \mathrm{e} \qquad \text{(A-47)}$$

Das bedeutet: Bei einer üblichen Sehentfernung von 50 cm soll die Zeichenhöhe zwischen 2,6 mm und 3,6 mm liegen. Die Zeichengröße sollte 2,6 mm nicht unterschreiten.

c) Zeichenschärfe

Die Zeichen auf dem Bildschirm sollten möglichst so scharf wie die *gedruckten* Zeichen sein. *Monochrome* Bildschirme sind im allgemeinen besser, weil sie schärfere Zeichen liefern.

d) Zeichenkontrast

Die unterste Grenze für den Zeichenkontrast liegt bei 1:3 (äußere Leuchtdichte zu Leuchtdichte der Bildschirmzeichen). Das heißt, ein Zeichen sollte mindestens 3-fach so intensiv sein wie seine Umgebung. Die optimalen Zeichenkontraste liegen zwischen 1:6 und 1:10. Für helle Zeichen auf dunklem Grund ist eine *obere Grenze* von 1:20 einzuhalten.

e) Zeichengestalt

Es ist zu raten, einen gewohnten Zeichensatz zu verwenden. Buchstaben mit ausgeprägten *Ober- und Unterlängen* sowie *Groß-* und *Kleinbuchstaben* dienen dazu, Informationen schnell zu erfassen.

f) Zeilenabstand

Texte müssen vom Zeilenende zum nächsten Zeilenanfang verfolgt werden können. In Abhängigkeit von der Zeilenlänge l, der Zeichengröße z kann der optimale Zeilenabstand d gefunden werden (alle Angaben in mm):

$$d > 0{,}05l + z \qquad \text{(A-48)}$$

Dabei ist d der Zeilenabstand in mm, z die Zeichengröße in mm und l die Zeilenlänge in mm. Für Korrekturen am Bildschirm sind Zeilenabstände von 1,5 mm bzw. 2 mm sinnvoll (größere Zeilen erfordern auch größere Zeilenabstände).

g) Flimmern

Der Bildschirm sollte absolut flimmerfrei sein.

h) Reflexe

Auf den Bildschirm auffallendes Licht sollte möglichst nicht in die Augen des Bedieners reflektiert werden. Dazu ist es ratsam, Bildschirme mit *optischer Entspiegelung* anzuschaffen.

Eingabe von Information

Informationen können in den Rechner eingegeben werden durch:

- Tastatur,
- Joystick,
- Maus,

- Tablett,
- Touchscreen und
- Sprache.

a) Tastatur

Es ist eine *seitliche Fingerführung* wichtig, die *konkave* Tasten ermöglichen. Die Tastengröße müssen *menschengerecht* sein, d. h. eine *Kantenlänge* (Durchmesser) von 12 mm bis 20 mm aufweisen und einen *Tastenabstand* zwischen 18 mm und 20 mm besitzen. Der *optimale Tastenweg* liegt zwischen 1 mm und 5 mm. Die Tasten sollten einen *spürbaren Auslösedruck* besitzen (für alle Tasten gleich: zwischen 0,25 N und 1 N) und eine schwarze Beschriftung aufweisen. Akustische Meldungen können die Arbeit am Bildschirm erleichtern.

Die Tastatur besteht aus *Buchstaben-, Zahlen-* und aus *Funktionstasten.* Je nach Aufgaben ist eine spezielle Tastenanordnung zu empfehlen. Größere Tastenfelder erfordern die Bewegung von *Hand und Arm.* Deshalb sollten häufig gebrauchte Tasten in Blöcken zu 3x4 Tasten zusammengefaßt werden, weil ein solcher 3x4-Block für das Auge gut zu überschauen und mit den Fingern noch leicht zu bedienen ist. Gefährliche Tasten, die das Programm zum Absturz bringen können, sollten eine Mehrfinger-Bedienung erfordern oder außerhalb des direkten Griffbereiches liegen.

b) Joystick

Mit dem Joystick wird der *Cursor* (Markierung der aktuellen Stelle) mit der Hand so bewegt, wie es dem Auge entspricht.

c) Maus

Damit wird der Cursor auf dem Bildschirm bewegt. Die motorische Beanspruchung ist besonders für ältere Menschen groß. Sehr anstrengend ist der *ständige Wechsel* zwischen Tastatur (z. B. Texteingabe) und Maus (z. B. Auswahl einer Funktion).

d) Tablett

Die einzelnen Funktionen sind auf Tabletts angeordnet, wie sie vor allem im CAD-Bereich (rechnergestützte Konstruktion) üblich sind. Es ist zu empfehlen, zusammengehörige Funktionen entsprechend zusammenhängend zu gruppieren. Es kann auf Dauer anstrengend sein, die Arbeit mit dem Auge (Bildschirm) und der Hand (Tablett) zu koordinieren.

e) Touchscreen

Bei diesem Eingabegerät wird mit einem Finger direkt auf die gewünschte Funktion am Bildschirm gezeigt. Die Koordination zwischen Hand und Auge ist optimal. Einschränkungen müssen hingenommen werden hinsichtlich der Auswahl der Funktionen (Bildschirmgröße) und der Breite des Zeigefingers.

f) Spracheingabe

Die Spracheingabe wird ein wichtiges Eingabemedium der Zukunft sein. Dazu ist eine starke *Sprachdisziplin* (bestimmter Wortschatz) und eine *Sprechdisziplin* (gleicher Tonfall) erforderlich.

Rechner

Hier ist zu prüfen, ob der Rechner getrennt vom Bildschirm untergebracht werden kann (z. B. als Tower-Ausführung), ferner ob die Ventilatorgeräusche gering zu halten sind. Der Rechner sollte wegen eventueller Reparaturen nicht zu aufwendig eingebaut sein.

Ausgabegeräte

Es sind zu unterscheiden:

- optische Ausgabegeräte und
- akustische Ausgabegeräte.

Zu den *optischen* Ausgabegeräten gehören die Drucker und Plotter. An die Schriftqualität und an die Zeichen sind dieselben Anforderungen zu stellen wie an den Bildschirm. Es ist zu empfehlen, einen möglichst *leisen* Drucker zu wählen, weil nachträgliche Schallschutzmaßnahmen nicht optimal sind.

Akustische Ausgabegeräte haben den Vorteil, daß diese Information *zusätzlich* zur bildlichen wahrgenommen werden kann. Die Lautstärke sollte mindestens 10 dB(A) über dem Geräuschpegel des Arbeitsplatzes liegen. Bei unbekannter Information wird ein Wert von 20 dB(A) vorgeschlagen. Die akustischen Ausgabegeräte ermöglichen *Sehbehinderten* die Arbeit mit dem Computer. Für Normalsichtige kann die akustische Ausgabe zur *Warnung* dienen oder zur Ausgabe, wenn das Auge überbeansprucht ist.

A 5.2.2 Software

Das Gebiet der *Software-Ergonomie* ist sehr differenziert und komplex. Software umfaßt eine aufgabenorientierte *(semantische)* Komponente und eine funktionsorientierte *(syntaktische)*. Das heißt, ein Software-Benutzer hat eine klare Aufgabe zu erledigen (semantisch) und benötigt dazu die Kenntnis der Wirkung von Funktionstasten oder Sprachbefehle (syntaktisch). Der Benutzer kann je nach Bedeutung der syntaktischen oder semantischen Komponente in folgende Gruppen eingeteilt werden:

- Anfänger
  (besitzt weder syntaktische noch semantische Kenntnisse);
- gelegentlicher Benutzer
  (besitzt semantische Kenntnisse, aber keine syntaktischen);
- Profi
  (besitzt sowohl semantische als auch syntaktische Kenntnisse).

Besondere Ansprüche an die Software-Ergonomie werden insbesondere erwartet an

- Benutzeroberfläche und die
- Kommunikation.

Benutzeroberfläche

Die Informationen auf dem Bildschirm sollten so gruppiert sein, daß sie *inhaltlich* zusammengehören und mit *einem Blick* erfaßt werden können.

Die Fülle der Informationen müssen für den Anwender *verdichtet* (kodiert) werden. Dies geschieht durch bestimmte Worte eines Menüs, durch spezielle Zeichen (Piktogramme) oder durch Abkürzungen. Die einzelnen Verdichtungsformen können durch *Größe, Helligkeit, Farbe* und *Ort* benutzergerechter angeboten werden.

Kommunikation

Dieses Feld beschreibt folgende Möglichkeiten, eine *Mensch-Maschine-Kommunikation* möglichst optimal zu gestalten:

- Befehle
  (Lernen von Kommandos, Funktionen und Anweisungen);
- Menüs
  (Vorgabe von Möglichkeiten; Auswahl durch Buchstaben oder Cursorbewegung durch Maus);
- Direktmanipulation
  (entweder eine *grafische Oberfläche* wie beim Macintosh oder in Windows mit der Auswahl durch die Maus oder mit *Pull-Down-Menüs,* die mehrfach gestaffelt sein können).

Qualitativ hochstehende Software muß nach DIN 66234 Teil 8 folgende Eigenschaften aufweisen:

- einfach und überschaubar,
- selbsterklärende Funktionen,
- steuerbar durch den Benutzer,
- erwartungsgerecht und
- fehlerrobust

(geringe Fehlermöglichkeit, klare Fehlerkennung und eindeutige Fehlerkorrektur).

Die *Qualität* der Kommunikation wird darüber hinaus durch folgende drei Merkmale bestimmt:

- Antwortzeit,
- Fehlermeldung und
- Hilfefunktion.

a) Antwortzeit

Die Antwortzeiten sollten erwartungsgemäß *kurz* und *konstant,* sowie auf den Arbeitsrhythmus des Benutzers eingestellt sein.

b) Fehlermeldung

Es ist eine allgemeine Weisheit, daß der Mensch aus *Fehlern lernt.* Deshalb sollten alle Fehlermeldungen auf den Verursacher positiv wirken. Folgende Empfehlungen lassen sich geben:

- immer an derselben Stelle ausgeben,
- konstant in Formulierung und Abkürzung sein,
- Wortwahl muß positiv und motivierend sein,
- Wortwahl muß konstruktiv sein (Maßnahmen zur Fehlerbehebung),
- genau den Fehler beschreiben und nur für diesen Hilfen anbieten (eventuell in mehreren Stufen).

c) Hilfefunktion

Mit der Hilfefunktion, die bereits fast einheitlich bei PC's mit der <F1>-Taste aufgerufen werden

kann, werden dem Benutzer folgende Hilfestellungen gewährt:

- Darstellung der Möglichkeiten, um die Aufgaben zu erfüllen,
- richtige (syntaktische) Verwendung von Funktionen,
- Fehlerkorrektur der Syntax (auch über 4 bis 5 Anweisungen hinaus) und
- Orientierung, an welcher Stelle man sich befindet.

A 5.3 Arbeitsplatz

Die Gestaltung des Arbeitsplatzes betrifft den zweiten Kreis der Ergonomie in Bild A-34. Die ergonomische Gestaltung des Arbeitsplatzes ist zwar abhängig von der Art der Bildschirmarbeit (Bild A-35); dennoch lassen sich einige allgemeine Regeln angeben.

Im allgemeinen verursachen Bildschirmarbeitsplätze keine gesundheitlichen Gefahren. Allerdings sollten folgende Vorschriften beachtet werden:

- richtige Körperhaltung,
- Anordnung von Bildschirm, Beleghalter und Tastatur,
- Arbeitstisch, sowie
- Bürostuhl und Fußstütze.

a) Richtige Körperhaltung

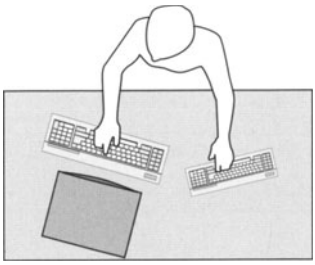
In Bild A-36 wird gezeigt, daß auf eine richtige Körperhaltung geachtet werden muß, bei der das Gesäß vollständig auf der Sitzfläche ist und die Bandscheiben *gleichmäßig* belastet sind (Bild A-36 a).

Schiefe Körperbelastungen erhöhen einseitig den Druck auf den Rand der Bandscheiben und führen so zu vorzeitigem Verschleiß der Bandscheiben und zu Rückenschmerzen. Bild A-36 c zeigt die Körperstellen, an denen sich Schmerzen bemerkbar machen können. Deshalb ist dringend anzuraten, daß man sich öfters kontrolliert und vor allem längere und schiefe Körperhaltungen (z. B. bei Blenden des Bildschirms) vermeidet.

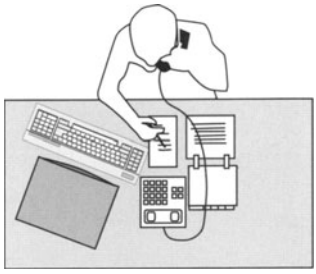
b) Anordnung von Bildschirm, Beleghalter und Tastatur

Bei der Aufstellung eines Bildschirms müssen folgende Fehler vermieden werden:

a Datenerfassung



b Dialogarbeitsplatz



c Textverarbeitung

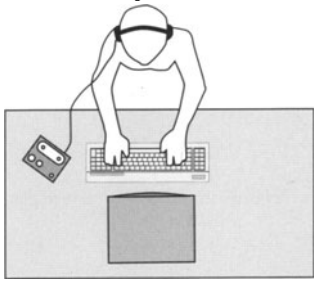


Bild A-35. Typische Tätigkeiten bei Bildschirmarbeitsplätzen (Quelle: BAD).

- zu große Hell-Dunkel-Kontraste,
- Spiegelungen und
- Blendungen.

Für die Aufstellung bedeutet dies: Die Blickrichtung vor dem Bildschirm ist *parallel* zur Fensterfront (Bild A-37). Bereiche direkt am Fenster sind zu meiden (Verwendung als Sozial- oder Besprechungsecke). Falls sich eine solche räumliche Anordnung nicht verwirklichen läßt, können

Bild A-36. Körperhaltung am Bildschirmarbeitsplatz (Quelle: BAD).

störende Blendungen und Reflexionen durch spezielle Lamellenstores verhindert werden.

c) Arbeitstisch

Am besten wäre ein *höhenverstellbarer* Tisch, der im Idealfall 72 cm hoch ist. Die sonstigen geeigneten Abmessungen für die Anordnung der Gegenstände und das Arbeiten am Bildschirm sind aus Bild A-38 zu entnehmen.

d) Bürostuhl und Fußstütze

Für die Sitzhöhe eines Bürostuhls gelten folgende Empfehlungen:

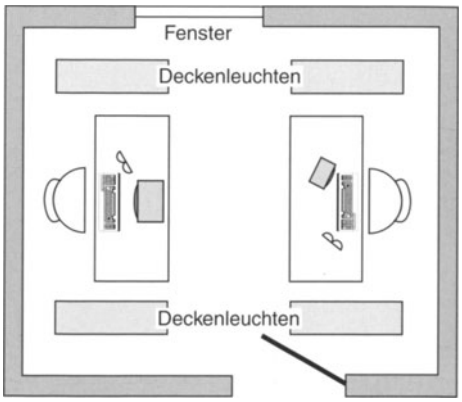


Bild A-37. Anordnung des Bildschirms (Quelle: BAD).

- Winkel zwischen Ober- und Unterschenkel bei aufgestellten Füßen 90° oder etwas mehr;
- Oberschenkel nicht zwischen Sitzfläche und Tischplatte einklemmen;
- Winkel zwischen Ober- und Unterarmen 90° oder etwas mehr;
- nur leicht geneigte Kopfhaltung (Bildschirmoberkante in Augenhöhe oder etwas darunter).

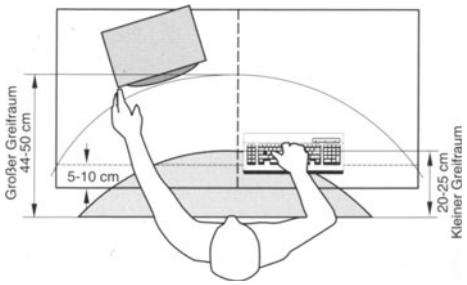


Bild A-38. Optimale Geometrie des Arbeitstisches (Quelle: BAD).

Bei der Sitzfläche ist zu empfehlen:

- Zwischen Kniekehle und Vorderkante der Sitzfläche sollte eine Handbreit Luft bleiben;
- gesamte Sitzfläche nutzen.

Für die Rückenstütze ist sinnvoll:

- Oberkante der Rückenstütze bis zur Mitte des Schulterblattes;
- Abstützung der Wirbelsäule an deren tiefster Einbuchtung.

Einen in dieser Weise ergonomisch gestalteten Arbeitsplatz zeigt Bild A-39.

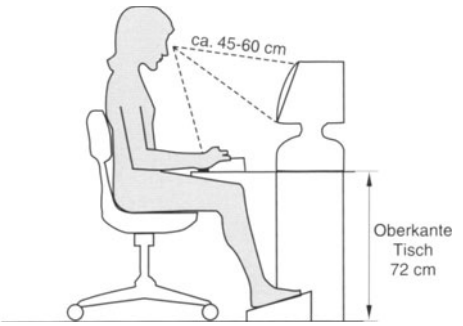


Bild A-39. Ergonomisch gestalteter Bildschirmarbeitsplatz.

A 5.4 Arbeitsumgebung

Die Arbeitsumgebung betrifft folgende Punkte:

- Beleuchtung,
- Lärm und
- Klima.

a) Beleuchtung

Nach DIN 66234 (Teil 7) und DIN 5035 soll die Beleuchtungsstärke zwischen 300 lx und 500 lx liegen. Die Leuchten sollten in Reihen brennen, die zum Fenster parallel liegen. Entscheidend ist auch die Verteilung der Beleuchtungsstärke, die vom Reflexionsgrad der umgebenden Gegenstände abhängig ist. So soll der Reflexionsgrad der Decke bei etwa 70% liegen und die der Wände zwischen 30% und 50%.

b) Lärm

Bildschirmarbeit erfordert hohe Anforderungen an das Auge, an die Feinmotorik und die Konzentration. Aus diesem Grund sollten zusätzliche akustische Lärmquellen vermieden werden. Hierzu zählen insbesondere Raumlüfter und Geräteventilatoren, aber auch Nadeldrucker und Plotter. Deshalb ist zu empfehlen, Tintenstrahl- oder Laserdrucker einzusetzen.

c) Klima

Dafür gelten die Richtlinien für die Büroräume. Es muß allerdings bedacht werden, daß die Geräte zur Erwärmung des Raumes beitragen. Nach den Vorschriften sollte die *relative Luftfeuchtigkeit* 40% bis 60% betragen, die *Luftgeschwindigkeit* Werte zwischen 0,1 m/s und 0,2 m/s aufweisen und die Raumtemperaturen zwischen 21°C und 23°C liegen.

A 5.5 Arbeitsgestaltung

Bei der Gestaltung der Arbeit mit Bildschirmarbeitsplätzen sind neben einer gründlichen *Einweisung* und *Schulung* folgende Gesichtspunkte von Bedeutung:

- Arbeitszeit,
- Mitbestimmung und
- Gesundheitsvorsorge.

a) Arbeitszeit

Das Gefühl der *Überbeanspruchung* muß den Mitarbeitern genommen werden. Dazu dient eine sinnvolle Mischung unterschiedlicher Tätigkeiten, wobei nach arbeitswissenschaftlichen Untersuchungen maximal 4 Stunden am Tag direkt am Bildschirm verbracht werden sollen. Bei *sich wiederholenden Arbeiten* und *ständigem Blickkontakt* mit dem Bildschirm ist eine Pause von 5 bis 10 Minuten je Stunde bzw. 15 bis 20 Minuten alle 2 Stunden sinnvoll. Pause bedeutet dabei nicht Ruhe, sondern auch das Beschäftigen mit anderen Aufgaben.

b) Mitbestimmung

Bei der Einführung von Computertechniken ist die Unternehmensleitung gut beraten, wenn sie versucht, die Angst vor einem befürchteten Arbeitsplatzabbau zu nehmen und klarzumachen,

daß mit dem Werkzeug Computer die vorhandenen Mitarbeiter ihre Arbeit *besser* erledigen können. Deshalb sollten die Mitarbeiter vor der Einführung informiert und nach ihren Wünschen befragt werden. Nur wenn die Mitarbeiter an ihrem Bildschirmarbeitsplatz gerne arbeiten, weil sie ihn auch mitgestalten konnten (z. B. eigene Bilder oder Farben der Stühle und Tische), werden sie produktiver werden und weniger anfällig für Krankheiten.

c) Gesundheitsvorsorge

Alle Mitarbeiter am Bildschirm sollten gesundheitlich aufgeklärt werden; insbesondere über ihre Belastungen in folgenden Bereichen:

- Hand und Arm,
- Kopf, Nacken und Rücken,
- Augen und
- Haut.

Es ist anzuraten, ab und zu Bewegungsübungen zu machen, um vor allem die Brust- und Lendenwirbelsäule zu stärken.

A 5.6 Checkliste zur Ergonomie

In Tabelle A-28 werden die in den vorigen Abschnitten aufgezeigten Erkenntnisse als Checkliste zusammengefaßt, um einen schnellen Überblick zu erhalten. Werden die Fragen mit „Nein" beantwortet, dann sollte Abhilfe geschaffen werden.

Tabelle A-28. Checkliste zur Ergonomie

Fragen	Antwort	
Zeichen	ja	nein
Ist die Zeichengröße ausreichend? (Minimum: 2,6 mm bzw. 0,052*Sehweite in cm)	○	○
Ist die Schärfe der Zeichen ausreichend?	○	○
Kann die Helligkeit verstellt werden?	○	○
Genügt der Zeichenkontrast? (1:6 bis 1:1)	○	○
Sind Ober- und Unterlängen vorhanden?	○	○
Ist der Zeilenabstand ausreichend? (Minimum: 0,05*Zeilenlänge)	○	○
Sind die Zeichen flimmerfrei?	○	○
Sind die Zeichen reflexfrei?	○	○
Tastatur		
Sind die Tasten zwischen 10 und 20 mm breit	○	○
Besitzen die Tasten einen spürbaren Druckpunkt?	○	○
Sind die Tastenoberflächen konkav?	○	
Liegt der Auslösedruck zwischen 0,25 und 1 N?	○	○
Ist die Tastenbeschriftung schwarz?	○	○
Entsprechen die Buchstabentasten den Erwartungen?	○	○
Sind die Funktionstasten leicht find- und bedienbar?	○	○
Können Ziffern mit einer Hand eingegeben werden?	○	○
Können gefährliche Funktionen durch weit abliegende Tasten betätigt oder durch Mehrfachtasten ausgelöst werden?	○	○
Rechner		
Kann der Rechner leicht repariert werden?	○	○
Ist der Rechner leicht verschiebbar?	○	○
Drucker		
Sind die Zeichen von Druckqualität?	○	○
Sind die Drucker leise?	○	○
Software		
Wird der Ausbildungsstand der Benutzer berücksichtigt?	○	○
Sind die Informationen gut in Gruppen strukturiert?	○	○
Sind ähnliche Informationen immer am gleichen Ort?	○	○
Sind die Bezeichnungen und Abkürzungen verständlich?	○	○
Sind die Symbole gut verständlich?	○	○
Kommunikation		
Sind die Befehle einfach und überschaubar?	○	○
Sind die Funktionen selbsterklärend?	○	○
Kann der Benutzer die Funktionen steuern?	○	○
Sind die Eingaben fehlerrobust?	○	○
Ist die Antwortzeit kurz genug?	○	○
Sind die Fehlermeldungen positiv formuliert?	○	○
Ist die Wortwahl motivierend?	○	○
Ist klar, wie der Fehler zu beheben ist?	○	○
Ist die Hilfefunktion ausreichend?	○	○
Gibt es ein lesbares elektronisches Handbuch?	○	○

Tabelle A-28 (Fortsetzung)

Fragen	Antwort	
Arbeitstisch	ja	nein
Beträgt die Höhe 72 cm?	○	○
Kann die Tischhöhe verstellt werden?	○	○
Ist die Größe ausreichend?	○	○
Ist unter dem Tisch ausreichend Platz für die Füße?	○	○
Arbeitsstuhl		
Ist der Stuhl höhenverstellbar? (40 bis 51 cm Sitzhöhe)	○	○
Hat der Stuhl eine Rückenlehne?	○	○
Ist keine elektrische Aufladung vorhanden?	○	○
Ermöglicht der Stuhl ein dynamisches Sitzen? (Drehachsen in der Nähe der Kniegelenke)	○	○
Ist der Winkel zwischen Ober- und Unterschenkel >90°?	○	○
Ist zwischen Kniekehle und Vorderkante der Sitzfläche eine Handbreit Luft?	○	○
Beleuchtung		
Ist die Beleuchtungsstärke zwischen 300 und 500 lx?	○	○
Liegen die Leuchten parallel zum Fenster?	○	○
Ist die Beleuchtung flimmerfrei?	○	○
Lärm		
Sind die Drucker geräuschlos?	○	○
Sind die Arbeitsplätze schallgedämpft?	○	○
Sind die Geräte mit Ventilatoren nicht auf dem Tisch?	○	○
Klima		
Beträgt die relative Luftfeuchtigkeit 40 bis 60%?	○	○
Arbeitszeit		
Arbeiten Sie nicht mehr als 4 Stunden pro Tag am Bildschirm?	○	○
Machen sie jede Stunde 5 bis 10 Minuten oder alle zwei Stunden 15 bis 20 Minuten eine andere Arbeit?	○	○
Mitbestimmung und Ausbildung		
Werden Sie bei der Gestaltung der Arbeitsplätze gefragt?	○	○
Werden Sie ausreichend geschult?	○	○
Gesundheitsvorsorge		
Werden Sie über Ihre gesundheitlichen Belastungen informiert?	○	○
Machen Sie von Zeit zu Zeit Entspannungsübungen?	○	○