Der Jurist als Programmierer

Carl-Theodor Olivet*

Wer einen Computer besitzt und mit professioneller Software arbeitet, wird mitunter feststellen, daß sie für seine Zwecke überdimensioniert ist. Dadurch kann sie für den erwünschten Zweck nicht nur unnötig teuer, sondern durch den mitgekauften Ballast auch schwer handhabbar sein. Die mitgelieferten Handbücher zeigen das zum Teil recht eindrucksvoll. Darüber hinaus enthält die professionelle Software oft auch nicht oder nur unvollständig das, was man sich im Sachzusammenhang der programmierten Materie auch noch wünscht und für wesentlich hält. Ergänzen kann man diese Programme nicht, weil sie zumeist nicht im Quellprogramm vertrieben werden, sondern in compilierter Form. Etliche wichtige Themenbereiche kann man als Programm gar nicht kaufen. Da stellt sich dann schon die Frage, ob es nicht sinnvoll ist, sich ergänzend eigene Programme zu erstellen, maßgeschneidert und im Geschmack der eigenen Diktion.

Hat man sich dazu durchgerungen und eigene — übrigens gänzlich unberechtigte — Minderwertigkeitskomplexe gegenüber dieser Materie überwunden, dann muß man sich darüber klar werden, ob auch die notwendige Zeit für ein eigenes Programmieren verfügbar ist. Man ist ja nicht von Haus aus Systemanalytiker und Programmierer, sondern Jurist mit voller Berufsbelastung. Ehefrau und Kinder sollen auch nicht zu Computerwitwe und -waisen werden, für andere Köstlichkeiten (Sport, Kunst, Hobby) muß dann letztlich auch noch Raum bleiben. Die nachfolgenden Erörterungen sollen einen Einblick dazu verschaffen, welchen Bereichen von Eigenprogrammen man sich zuwenden kann und was einen dabei erwartet. Die hier in den Beispielen verwendete Programmiersprache ist BASIC. Sie ist einfach und schnell erlernbar.

Wer sich einmal an das Programmieren herangewagt hat, wird sich der damit verbundenen Faszination nur noch schwer entziehen können. Man sollte also am Scheidewege „soll ich oder soll ich nicht" auch bedenken, daß sich einem damit ein ganz reizvolles Hobby eröffnen kann.

Meiner Erfahrung nach kann man folgenden „Machbarkeitskatalog" aufstellen:

1. Reine Rechenprogramme mathematischer/naturwissenschaftlicher Art: Dazu gehören Renten-/Diskont-/Zins-/Blutalkoholberechnung und dergleichen.

Die theoretische Aufbereitung als Grundlage für die Umsetzung in die Rechnersprache, ist bei solchen Programmen einfach und schnell gemacht. Studienbücher, etwa für Finanzmathematik oder Kraftfahrzeugdynamik, enthalten bereits die erforderlichen Formeln. Sie sind nur noch in die Programmsprache umzusetzen.

* Der Verfasser ist Richter am Landgericht Lübeck. Er hat das Buch „Computer-Programme für Juristen" veröffentlicht.

Die Systemanalyse wird durch die Formelvorgaben entbehrlich. Da man die Berechnungsbereiche seriell aneinanderhängen kann, stellen die Verknüpfungen zu einem Gesamtprogramm keine besonderen Anforderungen.

Beispiel: Der Barverkaufspreis eines Kleincomputers beträgt 800,— DM. Als Anzahlung sind 150,— DM zu leisten, sodann sind noch 12 Monatsraten zu je 60,— DM zu erbringen. Wie hoch ist der Effektivzins?

Ka = Barverkaufspreis
A = Anzahlungsbetrag
n = Anzahl der monatl. Raten
R = monatl. Rückzahlungsbetrag
p = effektiver Zinssatz

$$i = \frac{24*n}{n+1} * \left(\frac{R}{Ka-A} - \frac{1}{n}\right) \qquad i = \frac{p}{100}$$

$$i = \frac{24*12}{12+1} * \left(\frac{60}{800-150} - \frac{1}{12}\right) = 0.198815$$

$$p = 0.198815*100 = 19.88$$

Ergebnis: Der Effektivzins beträgt 19.88% p.a.

Das entsprechende Computerprogramm in BASIC lautet:

```
10 cls
20 INPUT "Barverkaufspreis      ";KA
30 INPUT "Anzahlung             ";A
40 INPUT "Monatsraten (Anzahl)  ";N
50 INPUT "monatliche Rate (DM)  ";R
60 I=(24*N)/(N+1)
70 I=I*(R/(KA-A)-(1/N))
80 I=I*100
90 I=INT(I*100+0.5)/100
100 PRINT "Der Effektivzins beträgt ";I;" % p.a."
```

(Zeile 90 bewirkt eine Aufrundung des Ergebnisses.)

Man kann an der Entwicklung deutlich erkennen, daß ein solches Programm keine großen Analysen und Strukturierungen benötigt. Faßt man etliche solcher Programme zusammen, dann kann man sich eine nützliche Berechnungshilfe für finanzmathematische Fragen jeglicher Art schaffen. Bei den heutigen Ratenkreditgeschäften ist das eine wesentliche Argumentations- und Prüfungshilfe.

Beispiel: (aus dem Kraftfahrzeugbereich) Kraftfahrer K befährt nachts per Pkw eine Landstraße. Die Scheinwerfer scheinen 70 m weit. Wie schnell darf er höchstens fahren, um vor einem Gegenstand noch sicher halten zu können, der plötzlich in 70 m Entfernung sichtbar wird? Wir wollen davon ausgehen, daß K 60 m Anhalteweg benötigt. Die Bremsverzögerung soll 3 m/sec betragen.

VO = erfragte Geschwindigkeit
a = Bremsverzögerung
t = Schreck- und Bremsansprechzeit (üblich ist 1 sec.)

$$VO = \sqrt{(a*t)^2 + 2*a*Anhalteweg} - a*t$$
$$VO = \sqrt{(3*1)^2 + 2*3*60} - 3*1$$
$$VO = 16.21$$

Damit ist die Geschwindigkeit in m/sec errechnet. Die km/h errechnen sich aus 16.21*3.6 = 58.36.

Ergebnis: K darf also nur mit 58.36 km/h fahren.

Das Programm lautet:

```
10 CLS
20 PRINT "Bitte geben Sie ein:": PRINT
30 INPUT "die Länge des gewollten Anhaltewegs (in m)";S
40 INPUT "die Bremsverzögerung (in m/sec2)";A
50 rem------------------------------Berechnungsteil
60 V=SQR((A*1)^2+2*A*S)-A*1
70 V=V*3.6: V=INT(V*100+0.5)/100: PRINT
80 PRINT "Maximal erlaubte Geschwindigkeit: ";V;" km/h"
```

(In Zeile 70 ist der 2. Befehl eine Aufrundung von V)

2. Schlichte Verwaltungsprogramme: Dazu gehören Sortier-, Textblock-, Dateiprogramme usw.

Einer theoretischen Aufbereitung bedarf es nicht. Es sollen ja nur schlichte Handhabungen ablaufen. Die Systemanalyse für die Programmierung wird einem in Kernbereichen durch gängige Literatur erleichtert. Es gibt genügend Bücher auf dem Markt, die zum Beispiel die verschiedenen Arten von Sortierroutinen darstellen (Bubble-Sort, Quick-Sort usw.). Gleichwohl ist die Analyse nicht ganz leicht. Solche Programme haben die Eigenart an sich, nach immer mehr Verfeinerungen zu verlangen. Man will also alsbald z. B. nicht nur sortieren, das Programm soll dann auch noch Worte suchen und ersetzen, Daten austauschen, Zeittafeln erstellen und Listen erweitern und ausdrucken können. So wird dann an Programmierzeit statt geplanter Tage plötzlich ein Zeitraum von Wochen. So richtig schwierig ist die Analyse gleichwohl nicht, weil die gebrauchten Routinen in Modulen erstellt werden können, also nicht fürchterlich miteinander verwoben und von einander abhängig sind.

Beispiel: Ein Programm soll erstellt werden, das bis zu 30 unsortierte Zeitdaten (jeweils mit Kurztext) sortiert.

Das Programm legt einen 40-Zeilen-Bildschirm zugrunde, die Texteingabe erlaubt zu jedem Datum maximal 254 Zeichen.

```
10 cls
20 DIM A$(30):DIM A(30):DIM T(30):DIM M(30):DIM J(30)
30 REM------------------------------Eingabeteil
40 PRINT "Eingabebeispiel: 18,6,87 (Kommata beachten)"
50 PRINT "Eingabeende erfolgt über Datum: 0, 0, 0"
60 PRINT:PRINT
70 FOR I=1 TO 30
80 INPUT "Tag/Monat/Jahr";T(I),M(I),J(I)
90 A(I)=T(I)+M(I)*30+(1900+J(I))*360
100 IF T(I)=0 THEN 160
110 N=N+1
120 PRINT STRING$(39,"-")
130 LINE INPUT "Text: ";A$(I)
140 PRINT
150 NEXT I
160 PRINT "Bitte warten Sie"
170 REM------------------------------Sortierroutine
180 FOR I=1 TO N-1
190 FOR J=I+1 TO N
200 IF A(I)<=A(J) THEN 260
210 H=A(I):A(I)=A(J):A(J)=H
220 K=T(I):T(I)=T(J):T(J)=K
230 L=M(I):M(I)=M(J):M(J)=L
240 M=J(I):J(I)=J(J):J(J)=M
250 B$=A$(I):A$(I)=A$(J):A$(J)=B$
260 NEXT J
270 NEXT I
280 H=0:K=0:L=0:M=0:CLS:PRINT
290 REM------------------------------AUSGABE
300 FOR I=1 TO N
310 PRINT T(I);".";M(I);".";J(I);":"
320 PRINT string$(13,"-")
330 PRINT A$(I):PRINT
340 L=L+1:IF L <> 2 THEN 410
350 PRINT
360 PRINT "Weiter=Return-Taste, Durchlaufende=1"
370 INPUT ZL
380 PRINT
390 IF ZL=1 THEN 420
400 L=0
410 NEXT I
420 L=0
430 PRINT "Wiederholen = 1, Ende = 2"
440 INPUT ZL
450 ON ZL GOTO 300,460
460 PRINT "Ende ":end
```

Man sieht, daß dieses Programm schon erheblich aufwendiger ist als ein Rechenprogramm, zugleich läßt es aber auch erahnen, welche großen Erleichterungen man sich mit einem solchen (hier nur auszugsweise dargestellten) Datenverwaltungsprogramm schaffen kann.

3. Rechenprogramme mit normativen Eingaben: Hierunter fallen etwa Gebäudebewertung, Unternehmensbewertung.

Normative Eingaben sind Eingaben, bei denen der Benutzer des Programms die Weichen für das Ergebnis stellt, indem er die Fragen mit Eigenbewertungen beantwortet. Den Gegensatz bilden deskriptive Eingaben. Wird zum Beispiel in einem Erbrechtsprogramm erfragt „lebten die Eheleute in Gütertrennung?", dann kann die Antwort nur deskriptiv ausfallen: ja oder nein. Wird indes abgefragt „wieviel ist der Gegenstand wert?", dann enthält sie ein normatives Element, eben eine Vorbewertung durch den Benutzer. Die Problematik, normative Elemente korrekt zu erfassen und zu definieren, ist also auf den Benutzer verlagert und kein Problem des Programms. Insoweit ist die theoretische Aufbereitung des Sachgebietes wesentlich erleichtert. Allerdings gibt es im betriebswirtschaftlichen Fachbereich sehr viele unterschiedliche Meinungen dazu, mit welcher Bewertungsmethode und in welcher Gewich-

tung zu arbeiten ist und welches die maßgeblichen Bewertungskriterien sind. Das macht die theoretische Aufbereitung etwas schwierig; man muß sich doch beträchtlich in die unterschiedlichen Meinungsstände einlesen, bevor man eine Linie für die Systemanalyse des Programms entwickeln kann. Die Systemanalyse ist indes wiederum recht einfach. Um das Programm flexibel zu halten und unterschiedlichen Ansichten und Argumenten dienlich zu machen, ist es sinnvoll, die vertretenen Hauptmeinungen zu programmieren und eventuell noch eine Saldierung der unterschiedlichen Ergebnisse nach den verschiedenen Bewertungsmethoden vorzusehen, so daß man ein Durchschnittsergebnis erhält. Die Bewertungsmethoden sind im Kern auch nur Rechenschritte und in Formeln erfaßbar. Ich habe ein Programm zur Unternehmensbewertung erstellt, in dem die Eingaben nach den neun (!) wesentlichen Bewertungsmethoden durchgerechnet sind.
Hier soll indes nur ein Auszug dargestellt werden.

Beispiel: Welchen Wert hat Unternehmen X nach der „Jahrkaufmethode" bei nachhaltig erzielbarem jährlichem Ertrag 100000,— DM und einem Landeszinsfuß von 8% p.a. Es soll davon ausgegangen werden, daß der Substanzwert des Unternehmens 1 Mio DM beträgt.

Mit dieser Methode wird für eine Anzahl von Jahren der Übergewinn vergütet. Ist er subjektgebunden, dann legt man 3–5 Jahre, ist er vornehmlich objektgebunden, dann legt man 5–8 Jahre zugrunde (hier sieht man deutlich das normative Element der erforderlichen Eingabe). Wir wollen von 5 Jahren ausgehen.

S = Substanzwert
n = Jahre
p = Landeszinsfuß

$$i = \frac{p}{100}$$

E = nachhaltig erzielbarer jährl. Ertrag

Unternehmenswert = S + n*(E − i*S)
UnternWert = 1.000.000 + 5*(100.000 − 0.08*1.000.000)
UnternWert = 1.100.000

Ergebnis: Das Unternehmen ist also 1100000,— DM wert.

Programmiert sieht das so aus:

```
 10 CLS
 20 PRINT "Bitte geben Sie ein:":PRINT
 30 INPUT "den Substanzwert :";S: PRINT
 40 INPUT "den nachhaltig erzielbaren jährl. Ertrag ";E
 50 INPUT "auf wieviele Jahre bezogen ";N: PRINT
 60 PRINT "den Landeszinsfuß für langfristige risiko-";
 70 INPUT "freie Kapitalanlagen ";P
 80 REM-------------------------------Berechnungsteil
 90 I=p/100
100 UNT=S+N*(E-I*S)
110 PRINT
120 PRINT "Der Unternehmenswert beträgt ";UNT; " DM."
```

So kann man die verschiedensten Berechnungen (Stuttgarter Verfahren, Methode der laufenden Goodwill-Abschreibung, Methode Schnettler, befristete Goodwillabschreibung, Mittelwertverfahren, Methode der Übergewinnverrentung usw.) sehr schnell anstellen und ist damit gegenüber Sachverständigen, dem Gericht oder Streitparteien im Vergleichsgespräch bestens präpariert.

4. Rechenprogramme mit verwobener Struktur: Hierunter fallen Pflichtteilsberechnungen, Quotenvorrechtsberechnungen in Versicherungsfragen usw.

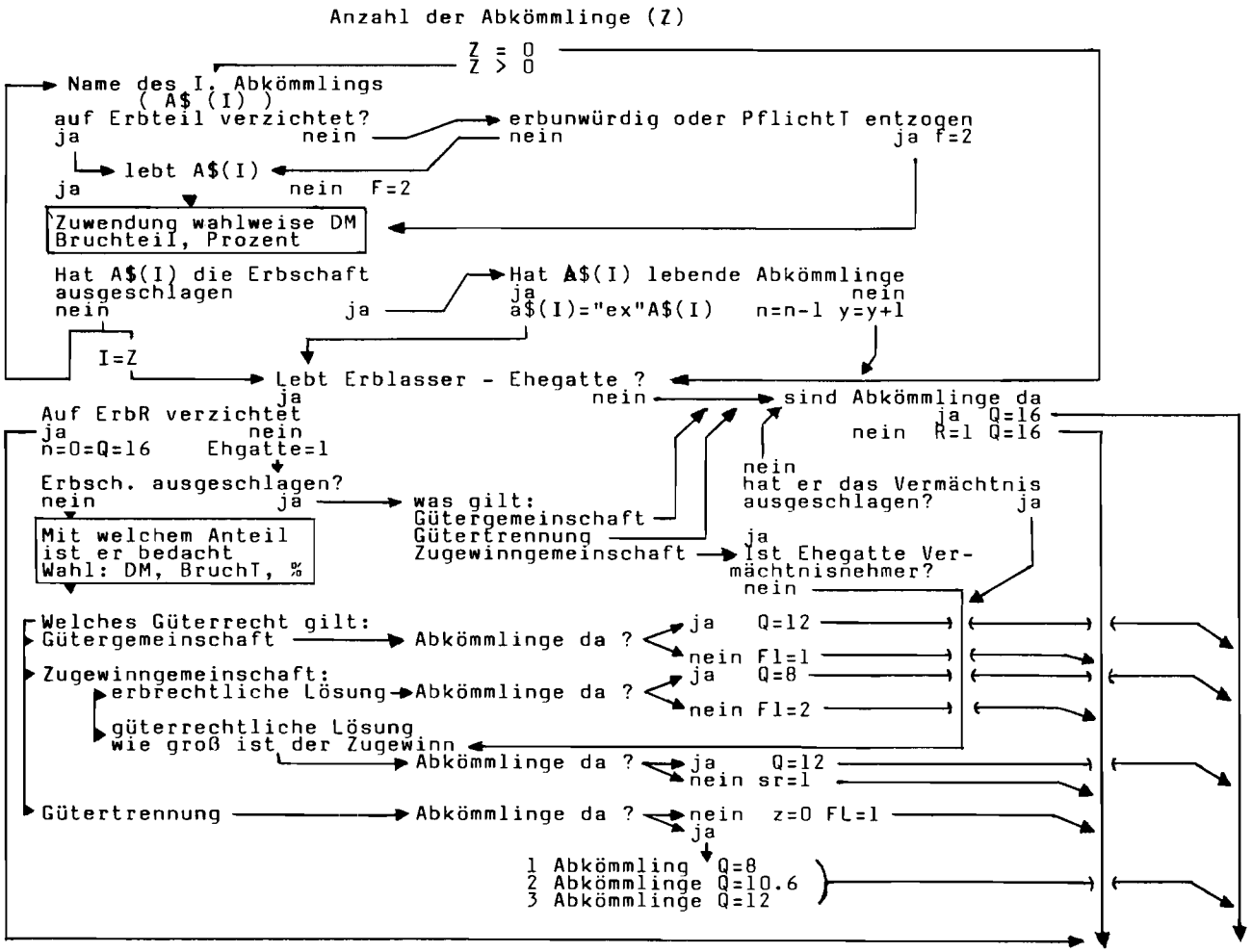
Die theoretische Aufbereitung ist überwiegend mühselig, weil solche Gesetzesmaterien nach vielen Richtungen stark wechselwirksame Beziehungen und Verquikkungen enthalten. Hier die Gesetzessystematik auch in den Feinheiten zu erfassen, ist nicht leicht. Als Grundlage sollte man nur erstklassige Großkommentare zu Rate ziehen, nachdem man sich über Kurzlehrbücher die Grundstrukturen noch einmal vergegenwärtigt hat. Ein solches Vorgehen hat sich bewährt. Man muß sich zugleich von dem Gedanken lösen, alle denkbaren Problemstellungen mit dem Computerprogramm erfassen zu könnem Das ist — wenn nicht gar unmöglich, so doch — zu zeitaufwendig.

Zweckmäßig ist es also, zum einen eine Bereichsbegrenzung vorzunehmen (etwa aus dem Erbrecht: Bereich Pflichtteilsrecht) und zum anderen die Analyse — Ebene des Programms zu beschränken (z.B. wird die Stammeserbfolge, § 1924 Abs. 3 BGB, nur noch en bloc analysiert). Das schwächt das Programm keineswegs. Wichtig ist zudem, daß man in der Programmvorbereitung nicht jeder kleineren Lehrmeinung Raum gewährt, sondern für den Benutzer des Programms nur dann Alternativen vorsieht, wenn sich gewichtige Lehrmeinungen gegenüberstehen.

Mit der theoretischen Aufbereitung ist bereits einen Großteil der Systemanalyse miterledigt. Recht schwierig gestaltet es sich nun, diese genannten Wechselwirkungen technisch in ein Programm einzubinden. Um sich nicht zu verzetteln, sollte man vor dem Programmieren unbedingt ein Fluß-Diagramm erstellen. Das bedeutet nun nicht, sich erst einmal einige Stunden hinzusetzen und ein solches Diagramm in allen Einzelheiten zu entwerfen. Das ist viel zu langweilig.

Ich skizziere mir bei meiner Arbeit zunächst nur gedanklich die Hauptlinien des Programms, auf einen Teilbereich beschränkt. Dann tippe ich aus freier Hand das entsprechende Programm ein. Nun teste ich dieses kleine Programm dahin durch, ob es in meinem Sinne funktioniert. Tut es das, dann skizziere ich mir gedanklich den Flußplan des nächsten Teilbereichs und ergänze das Programm entsprechend. Nun teste ich das bisherige Programm erneut durch und beseitige auftretende Fehler. So wird das Programm Stück um Stück erweitert. Irgendwann kommt dann natürlich der Zeitpunkt, in dem ich den Überblick zu verlieren drohe oder gar verliere. Ich drucke das Programm dann aus und erstelle nach diesem dann das Fluß-Diagramm, ge-

Diagramm zum Pflichtteilsrecht



rade so, wie es sich nach dem bisherigen Programm darstellt. Dadurch erhalte ich wieder den Überblick, kann das Diagramm ergänzen und der Ergänzung entsprechend das Programm erweitern. Auch dieses geschieht — wohl gemerkt — immer nur Stück um Stück.

Daß diese Vorgehensweise nicht der Schultheorie des Programmierens entspricht, ist mir klar. Ich meine jedoch, daß jeder eben so vorgehen sollte, wie es ihm am meisten Spaß macht.

Der folgende Diagramm-Auszug zum Pflichtteilsrecht stellt ein Beispiel eines Flußdiagramms dar. Es ist nicht in einem großen Wurf hingeschrieben, sondern — wie geschildert — allmählich gewachsen. Die aufgeführte Variable Q stellt die Erbteilsquote der Abkömmlinge dar. Von den beiden aus dem Bild führenden Pfeilen führt der linke zur Ordnung Erblasser — Eltern, der rechte zu den Fragen nach dem Nachlaßwert, den Anrechnungs- und Ausgleichspflichten, Schenkungen usw. F, FL, sr, n usw. sind Flaggen für später erforderliche Verzweigungen.

Hat man das Programm fertig gestellt, dann ist es umfassend auszutesten. Dazu sollte man die in den Kommentaren zu findenden Beispiele sämtlich mit dem Programm durchrechnen, nicht etwa nur eigen gebildete Berechnungsbeispiele. Der Vorteil der Fremdbeispiele ist der, zugleich zu sehen, ob man nicht etwas gesetzessystematisch oder -interpretatorisch mißverstanden hatte. Natürlich kann es vorkommen, daß ein Berechnungsbeispiel in einem Großkommentar falsch ist. Das ist aber recht schnell zu merken, weil man bei unterschiedlichen Ergebnissen von Kommentar/Computer auf andere Kommentare zugreift und die dort aufgeführten Beispiele zum Vergleich heranzieht. Bei der Programmierung des Pflichtteilsrechts zeigte mein Programm häufiger abweichende Ergebnisse. Die Ursachenforschung ergab, daß die Kommentatoren die Begriffe Ausgleichspflicht und Anrechnungspflicht nicht sauber auseinanderhielten. Fehler sollte man also nicht immer nur bei sich selbst suchen.

Zu dem Programm sollte schließlich ein genaues Protokoll (eine Kommentierung, Dokumentation) erstellt werden. Es kommt immer mal vor, daß man ein Programm nach Jahr und Tag gerne erweitern oder ändern will. Ist dann keine Erläuterung zur Programmstruktur sowie zur Bedeutung der einzelnen Flaggen und Variablen zur Hand, dann ist es sehr mühselig, sich darin wieder zurecht zu finden.

5. Programme zur Analyse von Gesetzesstrukturen

Analysen der Verzweigungen zum Eigentümer-/Besitzerverhältnis, der Anspruchsvoraussetzungen nach der VOB (Verdingungsordnung für Bauleistungen) usw.

Solche Programme herzustellen stellt meiner Meinung nach eine sehr reizvolle Herausforderung dar. Wer die Gerichtspraxis kennt, weiß den Nutzen solcher Programme zu schätzen. Meines Erachtens besteht für sie ein echtes Bedürfnis. Ich habe derartige Programme noch nicht erstellt, entgegen der hochtrabenden Charakterbezeichnung solcher Programme wird es jedoch nicht allzu schwierig sein. Vorstellbar ist — prinzipiell — folgendes Struktogramm:

Man sucht die verschiedenen Anspruchsnormen und deren Voraussetzungen zusammen. Dann ordnet man sie pyramidenartig

Anspruchsgrundlage	**1 = Tatbestandsmerkmale**	**a + b**
	2 =	**a + b + c**
	3 =	**a + b + c + d**
	4 =	**a + b + c + d + e**
usw.		

Diese Aufstellung wird in einer Datei abgelegt.

Im Programm baut man die Fragen zu den Tatbestandsmerkmalen entsprechend auf. Sobald eine Frage mit „nein" beantwortet wird, unterbleiben die noch folgenden Fragen. Sind etwa drei Fragen richtig beantwortet, dann gilt die Anspruchsgrundlage Nr. 2 und wird entsprechend angezeigt. Machbar ist das also mit einem Schablonen-Prinzip. Sind Verzweigungen erforderlich, dann muß man entsprechende weitere Schablonen in Dateien anlegen und entsprechend abrufbar machen.

Aus den Erörterungen ist zu ersehen, daß es sich schon lohnt, eigene Programme herzustellen. Mit ein wenig Phantasie lassen sich viele Möglichkeiten finden, die eigene täglliehe Arbeit mit kleineren Hilfsprogrammen zu erleichtern. Auf diese Weise sind unschätzbare Argumentationsvorteile gegenüber Prozeßgegnern oder Verhandlungspartnern zu gewinnen, die sich ein Jurist nicht entgehen lassen sollte. Spätestens dann, wenn der Herr Gegner mit solchen Argumentshilfen auftritt, wird man erkennen, daß tatsächlich die Floskel, Stagnation sei Rückschritt, ihre Berechtigung hat.

Mit den folgenden beiden Beiträgen wird im Rahmen der JuR-Serie „Textverarbeitungsprogramme" die Berichterstattung zu Word Perfect vorläufig abgeschlossen. Nöcker und Schultze besprechen die Version 4.2. Röder beschreibt ein Zusatzprogramm, das zur Behebung von Mängeln des automatischen Trennsystems (vgl. bei Nöcker/Schultze 3c) ausgeliefert wird. Juristisch interessant ist dabei die Frage, ob ein diesbezüglicher Nachbesserungsanspruch besteht.

Word Perfect 4.2 — auf dem Weg zur Perfektion

Thomas Nöcker und Jörg-Martin Schultze*

I. Einleitung

Das Textverarbeitungssystem WP 4.1 erfreut sich seit einiger Zeit steigender Beliebtheit.[1] Vor kurzem ist auch auf dem deutschen Markt die Nachfolgeversion WP 4.2 erschienen. In dieser Version sind neben der Einfügung neuer Funktionen (siehe unten II 2 und 3) einige der von uns[2] angeführten Fehler nnd Unzulänglichkeiten von WP 4.1 beseitigt worden (siehe unten II 1).

II. Änderungen in WP 4.2

1. Behobene Mängel

a. Linien über den rechten Rand

Anders als bei WP 4.1 verursacht der Versuch Linien über den rechten Rand zu ziehen keinen Absturz mehr. Die Linie wird „ordnungsgemäß" zur Trennung vorgeschlagen.

b. Sicherungsdateien

Hier hat die neue Version einen großen Mangel beseitigt: sowohl Text 1 als auch Text 2 werden gesichert, wenn sie sich insgesamt — wenn auch mehrfach unterbrochen — die eingestellte Zeitspanne lang auf dem Bildschirm befunden haben. Es kann also nicht mehr vorkommen, daß durch „ungünstiges" Hin- und Herschalten eine automatische Sicherung der erstellten Texte völlig unterbleibt. WP 4.2 legt im Gegensatz zu der Vorgängerversion Sicherungsdateien nach Ablauf der eingegebenen Zeitspanne aber nur noch dann an, wenn eine Textänderung stattgefunden hat. Das schont, wenn sich ein gleichbleibender Text längere Zeit auf dem Bildschirm befindet, die Laufwerke.[3]

* Wiss. Mitarbeiter am Institut für Ausländisches und Internationales Privat- und Wirtschaftsrecht der Westfälischen Wilhelms-Universität Münster.

[1] Allgemein zu WP 4.1 Scheidt IuR 1987, S. 196; speziell zu den Mängeln Nöcker/Schultze IuR 1987, S. 236.

[2] in IuR 1987, S. 236.

[3] ... und die Nerven des Bearbeiters!