Textverarbeitung und Wissensrepräsentation Oder: Das nur scheinbar unschuldige Instrument

Maximilian Herberger

A. Vorbemerkung

Seit einiger Zeit zitieren an Fragen der Wissensrepräsentation interessierte Informatiker gerne eine Abraham Maslow zugeschriebene[1] Feststellung: „To a person having a hammer every problem looks like a nail." Diese gelungene Formulierung macht schlagartig deutlich, daß die Wahl des Instruments die Art und Weise beeinflußt, in der ein Problem „als Problem" erlebt wird: Für denjenigen, der einen Hammer als Werkzeug gewählt hat, sehen plötzlich (denkt er nicht reflektiert genug) alle Probleme wie Nägel aus.

Die Beispiele mißlungener EDV-Einführung sind überwiegend (auch) dadurch gekennzeichnet, daß man die Instrumente nicht mit Rücksicht auf das zu lösende Problem (nach genauer methodischer Analyse desselben) auswählte, sondern lediglich Instrumente anbot in der leichtfertigen Annahme, diese müßten wegen einiger leicht ins Auge fallender Vorzüge zu einer Erleichterung und Unterstützung des Arbeitsablaufs führen.

Daß die „bloß" instrumentelle Sicht von EDV-Werkzeugen eine Konsequenz unzureichender Analyse des zu unterstützenden Vorgangs sein kann, läßt sich theoretisch schlüssig begründen und besonders gut an komplizierten Software-Gebilden (wie etwa „Expertensystemen") exemplifizieren. Eine solche Darlegung erfordert aber ein vertieftes Eingehen auf Software-Technologie und kann sich in den seltensten Fällen auf eigene Erfahrungen des juristischen Lesers verlassen. Es trifft sich deshalb gut, daß die angedeutete Problemlage auch schon bei scheinbar ganz unproblematischen Software-Produkten wie etwa der Textverarbeitung auftritt, und hier hat sie die juristische Praxis nicht zufälligerweise zuerst wahrgenommen. Deswegen soll im folgenden dieser der Praxis, der Anschaulichkeit (und dem Ort der Veröffentlichung) verpflichtete Ausgangspunkt gewählt werden, um in verhältnismäßig unaufwendiger Weise zu demonstrieren, daß sich schon bei einer Maßnahme wie der Einführung der Textverarbeitung bei Gericht oder am Richterarbeitsplatz Fragen stellen, die über eine kurzschlüssig maßnahmenorientierte Betrachtungsweise hinausführen.

B. Über Textverarbeitungs-Metaphern und die falsche Metapher des „Textbausteins"

EDV-gestützte Textverarbeitung bietet eine Fülle von Möglichkeiten, die unmittelbar (und zu Recht) als Vorteil erlebt werden. Es genügt auf die Leichtigkeit des Änderns, auf die Suchmöglichkeiten und die unproblematische Wiederverwendbarkeit des einmal Geschriebenen zu verweisen. Alle diese Vorteile hängen damit zusammen (man könnte auch sagen: werden dadurch erkauft), daß der Text während des Arbeitsvorgangs nicht mehr „in Papierform" verfügbar ist, sondern in eine elektronische Repräsentations-Gestalt überführt worden ist, die zu anderen Wahrnehmungsformen führt, als sie der geschriebene Text bietet. Das wird beispielsweise schon dann anschaulich deutlich, wenn einige Handbücher zur Textverarbeitung davon sprechen, man müsse sich den Text auf dem Bildschirm „als eine Rolle" vorstellen, die bei der Bewegung des Cursors nach unten schrittweise „aufgerollt" wird. Diese vom englischen Wort „to scroll" hergeleitete Sichtweise („scrolls" sind die Schriftrollen) läßt erkennen, daß man beim „Scrollen" eben nicht blättert, wie man es vom gedruckten Werk gewohnt ist, sondern sich den Text in anderer Weise veranschaulicht.

Die Überführung des Textes in eine andere Repräsentationsform bleibt nicht folgenlos, sie hat vielmehr Konsequenzen für den Umgang mit dem Text. Diese Konsequenzen können hilfreich oder schädlich sein, bisweilen hängt der Effekt auch von der Klarsicht des Benutzers ab. Sehr oft drückt sich die Haltung des Benutzers, seine besondere Sicht- und Erlebnisweise in z.T. vorgestanzten Metaphern aus. Das war eben schon am Beispiel des „rollenden Textes" zu sehen. Noch deutlicher wird es, wenn man (und darauf sollen sich die folgenden Überlegungen konzentrieren) das Beispiel der Rede vom „Textbaustein" betrachtet.

Die metaphorische Wendung vom „Textbaustein" erweckt den Eindruck, als habe man es hier mit „Bausteinen" zu tun, die – einmal zusammengefügt – ein Gebäude ergeben. „Stein auf Stein, das Häuschen wird bald fertig sein", wie es im Kinderreim heißt. Nun mag diese Metapher für manche Arten von Texten zutreffen, etwa für die Komposition von Werberundschreiben an heterogene Adressatengruppen, wenn jede der Zielgruppen an bestimmten Stellen gesondert angesprochen werden soll. Tatsächlich sind ja auch die ersten Textverarbeitungsanwendungen im PC-Bereich sogleich mit einer „Mailmerge"-Komponente verbunden worden, die auf derartige Anwendungen abzielt.

Für den richterlichen (und auch anwaltlichen) Arbeitsplatz indessen erweist sich die Metapher yom Textbaustein deswegen als unzutreffend (und damit – wie jede falsche Metapher – als gefährlich), weil juristische Texte bereits bei relativ unkomplizierten Standardanwendungen nicht als Aggregat (unmodifizierbarer, „fester") Einzel-„Bausteine" begriffen werden können. Vielmehr sind die „Bausteine" durch eine innere Komplexität gekennzeichnet, auf die man bei der Komposition des juristischen Dokuments Rücksicht nehmen muß. Tut man das bei der Organisation des Einsatzes von Text-Bausteinen nicht, sind Fehler unvermeidlich: Die falsche Metapher rächt sich in der Praxis. Zur Veranschaulichung genügt es, auf ein Beispiel wie das im Beschluß des Hessischen VGH vom 26. Juni 1984[2] behandelte zu verweisen: Dort waren u.a. im „maskulinen Singular" formulierte Textbausteine verwandt worden, obwohl es um mehrere Kläger ging.

C. Das „Innere" eines Textbausteins

Der eben zitierte Beschluß hatte sich mit so groben Fehlern beim Einsatz von Textbausteinen zu befassen, daß man geneigt sein könnte, jeden größeren theoretischen Aufwand zur ver-

1) Ich konnte die Original-Fundstelle bisher nicht verifizieren. Hinweise werden gerne entgegengenommen.
2) Az. 10 UE 1528/84, Informatik und Recht 1986, S. 69 – 71.

tieften Analyse der Problemlage als überdimensioniert abzulehnen: Daß dort, wo Plural geboten ist, kein Singular stehen darf, leuchtet sofort ein. Versucht man jedoch, die Konsequenz daraus für (sit venia verbo) die „Theorie des Textbausteins" zu ziehen, so löst sich schnell die scheinbar kohärente Einheit dieses „Steins" in eine sehr komplexe Struktur auf, dies allerdings nur dann, wenn sich ein vordergründiger Lösungsweg zur Rettung der Textbaustein-Kohärenz als nicht gangbar erweist.

Dieser Lösungsweg zielt darauf ab, „modifikationsfeste" Textbausteine herzustellen. Um das Singular-/Plural-Problem zu umgehen, könnte man beispielsweise die Wendung „die Klägerseite" einführen und dann durchgehend damit operieren. Man hätte dann auch zugleich das Problem der Differenzierung nach dem Geschlecht umgangen[3].

Abgesehen davon, daß der eben skizzierte Lösungsweg zu sprachlich unschönen Formulierungen führt, gibt es eine prinzipielle Schranke methodischer Art für einen derartigen Ausweg. Diese Grenze wird deutlich, wenn man in dem zu erstellenden Text (von Rechts wegen) die Differenzierung thematisieren muß, die man zunächst in der „differenzierungs-neutralen" Fassung des Textbausteins ausgeklammert hatte. Man ist spätestens dann auf die Ausgangsfrage nach der inneren Struktur des Textbausteins zurückverwiesen.

Nähert man sich dieser Frage am Beispiel der Unterscheidung „Singular/Plural", so fällt auf, daß diese elementare Kategorisierung sich (ebenso wie die Unterscheidung nach dem Geschlecht) auf handelnde Personen bezieht. Es geht schon hier um Eigenschaftszuschreibungen an diese handelnden Personen (Auftreten in der Einzahl/Mehrzahl, Geschlecht männlich/weiblich). Wenn das so ist, müssen im Textbaustein „Platzhalter" (Variable) mindestens für die Bezeichnung von Personen mit der Möglichkeit der darauf bezogenen Quantifizierung und der Eigenschaftszuschreibung vorgesehen werden und - was noch wichtiger ist - das Textverarbeitungsprogramm muß (wovon sogleich noch die Rede sein soll) das „Belegen" dieser Variablen mit bestimmten Werten unterstützen.

D. Von der Strukturanalyse des Textbausteins zur Handlungstheorie

Das bisher zur inneren Struktur des Textbausteins Gesagte kann noch keinen hohen Neuigkeitswert für sich beabspruchen, obwohl vielleicht sogar derartige einfache Analysen schon zur Vermeidung bestimmter Fehler im justiziellen Textverarbeitungssektor beitragen könnten. Nicht ganz so selbstverständlich akzeptiert ist jedoch der folgende Schritt, mit dem - abstrakt formuliert - postuliert wird, daß die Formulierung „strukturierter" Textbausteine mit Variablen eine Handlungs- und Rollentheorie des Bereichs voraussetzt, in dem die Textbausteine Anwendung finden sollen.

Eben wurde gesagt, daß (und das ist der unbestrittene Kern einer geläuterten Auffassung vom Textbaustein) die Binnenstruktur solcher Textportionen durch den Einbau von „Platzhaltern" (mindestens) für die Bezeichnung von Personen gekennzeichnet sein muß. Setzt man an diesen Stellen dann nicht Individuennamen ein, sondern wählt man Bezeichnungen wie „Kläger" (Erblasser, Vermächtnisnehmer, Schuldner, Gläubiger etc. - die Reihe ist beliebig fortsetzbar), so stellt man auf die „Rolle" dieser Personen in einem rechtlich relevanten Handlungsumfeld ab. Die Herstellung so strukturierter Textbausteine setzt damit Klarheit über die möglichen „Rollen" voraus, die Personen in dem von den Textbausteinen zu erfassenden Wirklichkeitsausschnitt „spielen" können. Die gleiche umfassende Übersicht benötigt man auch bezüglich der für diese Personen möglichen Handlungen. Auch hier muß man (weil diesen Handlungen in variierender Weise - teils durch grammatische Kategorien - Eigenschaften zugeschrieben werden können) mit Variablen für Handlungen operieren und wissen, welche Belegungen dafür in Frage kommen: So kann der Schuldner beispielsweise kündigen, in Verzug geraten, erfüllen etc. Hat man den geforderten Überblick über die in Frage kommenden Personen, ihre denkbaren Rollen und Handlungsmöglichkeiten, so verfügt man über eine (Handlungs-) Theorie des zu analysierenden Bereichs. Daß man eine derartige Theorie für die logische Analyse juristischer Texte benötigt, war den auf diesem Sektor arbeitenden Wissenschaftstheoretikern der Jurisprudenz geläufig. Das Überraschende ist nun aber, daß die sinnvolle Konstruktion von Textbausteinen genau dieselbe Art von Analyse voraussetzt. Damit zeigt sich ein Phänomen, das beim EDV-Einsatz in juristischen Arbeitsumgebungen verhältnismäßig häufig zu beobachten ist: Im neuen Instrument verborgen „schlummert" ein Problempotential, das nur durch den Einsatz von Methodologie zu bewältigen ist. Das bedeutet nun nicht, daß diese Methodologie abstrakt und praxisfern daherkommen müßte. Im Gegenteil: Die experimentellen Möglichkeiten von EDV-Umgebungen laden geradezu dazu ein, Theorie praxisnah zu erproben. Und das zeigt sich schon beim Entwurf eines Systems von Textbausteinen.

E. Von der prinzipiellen Ungeeignetheit „normaler" Textverarbeitung für die juristische Arbeit mit Textbausteinen

Wenn die These zutrifft, daß juristische Textbausteine adäquat nur dann konzipiert sind, wenn sie eine personen- und handlungsorientierte „Binnenstruktur" aufweisen, so fragt es sich, ob gängige Textverarbeitungs-Programme den Umgang mit derartigen „Textbausteinen" in geeigneter Weise unterstützen. Bezogen auf die Grundausstattung (also den Teil der Programme, der keine eigene Programmierarbeit des Anwenders voraussetzt) muß die Antwort verneinend ausfallen. Damit hat die kursorische Analyse ein relativ weitreichendes Ergebnis erbracht: Ein Organisationsmodell, das darauf abzielt, juristische Textbausteine ohne zusätzliche Programmunterstützung mit marktgängigen Textverarbeitungs-Programmen verwalten zu lassen, leidet von Anfang an unter einem gefährlichen Kunstfehler.

Eben wurde die Einschränkung gemacht, daß das die Standard-Textverarbeitungsprogramme betreffende Verdikt nur für den Teil dieser Programme gilt, der keine eigene Programmierarbeit des Anwenders voraussetzt. Berücksichtigt man die in Word (in der neuesten Version) oder WordPerfect enthaltenen Möglichkeiten, sog. „Makros", also Befehlsketten zu erstellen, sieht die Situation etwas anders aus. Denn diese „Makro-Programmierbarkeit" erlaubt es grundsätzlich, je nach Konstellation bestimmte Textpartien - unter Umständen mit Abfrage-

3) Falls man nicht eine Wendung wie „die Klägerseite" für einseitig maskulinorientiert erklärt. Will man auch diese Akzente beseitigen, gerät man in noch größere Kalamitäten und muß dann zu Formulierungen wie „die klagende Seite" etc. greifen.

möglichkeiten - so zusammenzufügen, daß größere Flexibilität der „Komposition" gegeben ist. Bei dem gegenwärtigen Stand der in Textverarbeitungsprogramme integrierten „Programmier"-Möglichkeiten gilt diese Feststellung aber nur prinzipiell und ist außerdem mit dem Vorbehalt zu versehen, daß die Erstellung derartiger Makros nicht ganz einfach ist, dies mindestens dann, wenn ein gewisser Komplexitätsgrad der Anwendung erreicht ist. Eine endgültige Schranke ist in dieser Umgebung gegenwärtig noch dann gegeben, wenn man - wovon sogleich noch als Postulat die Rede sein muß - bei der Dokumentenkomposition eine innere Kontrolle der Stimmigkeit für die Textbaustein-Zusammenfügungen verlangt. Das ist mit gegenwärtig in der Textverarbeitung vorgesehenen Makro-Programmiermöglichkeiten nicht zu realisieren.

Aus alledem folgt, daß gängige Textverarbeitungsprogramme, selbst wenn man die durch Makro-Programmierbarkeit gegebenen Möglichkeiten in Rechnung stellt, nicht das adäquate Instrument zur Erstellung, Verwaltung und Anwendung juristischer Textbausteine sind.

F. Über die „Textverarbeitung" hinaus: Die Idee des „Document Drafting Processor"

Gedanken wie die bisher skizzierten waren innerhalb der „American Bar Foundation" (ABF) der Anlaß, über ein „alternatives" Modell der Verarbeitung von Texten zum Zwecke der Erstellung juristischer Dokumente nachzudenken. Das Ergebnis war die sog. „ABF Language", der Vorschlag eines von den Programmen zu realisierenden Sprachstandards, die über „bloße" Textverarbeitung hinaus die strukturorientierte Dokumentenerstellung unterstützen wollen. Eine Untermenge dieser Sprache ist in dem von James A. Sprowl (Chicago) konzipierten „Document Drafting Processor"[4] realisiert, von dem im folgenden die Rede sein soll, um das bisher Gesagte zu veranschaulichen.

Der „Document Drafting Processor" ist ein Programm, das mit Hilfe eines sog. „Modelldokuments" zusammen mit Benutzereingaben einen individuellen juristischen Text erstellt. Das Modelldokument enthält Markierungen für die Variablen-Positionen und Texte für die Fragen, die dem Benutzer an der jeweiligen Stelle vorgelegt werden sollen. Die Benutzereingaben werden nicht nur zur Erstellung des gerade anstehenden Dokuments verwendet, sondern auch in einer Mandantendatei abgelegt, in der sie datenbankmäßig zugänglich sind. Im folgenden sollen diese Komponenten an Hand eines (einfachen) Beispiels näher erläutert werden. Dabei finden Komponenten Verwendung, die auf der Programm-Diskette als Beispiel für die Redaktion eines notariellen Testaments bereitgestellt sind.

G. Die Arbeit mit dem „Document Drafting Processor"

1. Der Hintergrund: Das „Modell"-Dokument

Das eben erwähnte „Modell"-Dokument für ein ganz einfaches notarielles Testament hat folgendes Aussehen:

```
                Last Will And Testament
                        Of
                [name: the testator's name]

I, [name: the testator's name], of [address: the city and state where the
testator lives], revoking all prior wills and codicils, do hereby make
this my last will and testament.

 [ IF married: the testator is married INSERT ]
I leave all of my property, both real and personal, to my [h or w:
husband or wife, the proper term for the testator's spouse], [sp name:
the name of the testator's spouse], if [he or she: he or she, the proper
pronoun for the spouse] survives me by more than 30 days.  Otherwise I
leave all of my property, both real and personal, to my children per
stirpes.
 [ OTHERWISE INSERT ]
I leave all of my property, both real and personal, to my children pre
stirpes.
 [ END IF ]

I hereby appoint [exec: the name of my executor] executor of my estate.
 [IF no bond: the executor need not post bond INSERT ]
My executor need not post security on the probate bond.
 [ END IF ]
 [ IF no accts: the executor need not file any accounts INSERT]
My executor shall not have to file any inventories or accounts.
 [ END IF ]

Signed by mo before witnesses this [long date].

                                    [name ]
Witnessed:
```

„Modell"-Dokument „WILL.MDL"

2. Erläuterungen zu dem „Modell"-Dokument

a. Die Variablen-Markierungen mit zugehörigen Fragetexten

In dem „Modell"-Dokument fallen zunächst die in eckige Klammern gesetzten Partien auf. Gleich zu Beginn findet sich die erste derartige Passage in folgender Form:

```
[name: the testator's name]
```

Dabei ist „name:" der Variablenname, hier „name" für den Namen des Erblassers. Der Text nach dem Doppelpunkt ist der Fragetext, den das Programm dem Benutzer vorlegt, wenn er mit diesem „Modell"-Testament ein individuelles Testament erstellen will. Das Programm würde sich dementsprechend wie folgt melden:

```
VARIABLE: name
          The testator's name:
>
```

An der durch „<" markierten Stelle wird die Benutzereingabe für die Belegung der Variablen „Name des Erblassers" erwartet.

b. Das Schleifen-Konzept: Bedingte Verzweigungen

Neben den unbedingten Variablen-Abfragen fallen Partien auf, die Bedingungen ins Auge fassen und je nach Antwort zu unterschiedlichen Alternativen verzweigen:

```
[ IF married: the testator is married INSERT ]
... Text ... [...],
[ OTHERWISE INSERT ]
... Text ...
[ END IF ]
```

4) Die „American Bar Association" verteilt im Rahmen von „Softlaw - As Is Software Exchange" für einen nominellen Unkostenbeitrag von $ 10 die vorläufige Testversion dieses Programms. In dieser Version ist das Programm auf die Verarbeitung von maximal 50 Variablen beschränkt. Ich bin bemüht, für die Mailbox die Erlaubnis zu erhalten, das Programm dort anzubieten.

An dieser Stelle läuft folgende Sequenz ab:

```
VARIABLE: married
          The testator is married
          YES OR NO? (Type Y, N, or ?)
>
```

Je nach Antwort wird entweder der für die Antwort „Ja" vorgesehene Text eingesetzt oder der für den anderen Fall („otherwise") bereitgestellte Text. Dabei können innerhalb der einzelnen Varianten erneut Variable abgefragt werden, wie dies in dem Beispiel der Fall ist.

Mit den vorgestellten Konstrukten können die in einem linearen Ablauf nötigen Programmierelemente zur Dokumentenerstellung verwandt werden. Dies geschieht so, daß man das Modell-Dokument unter Beachtung der beschriebenen syntaktischen Regeln in einem Editor erstellt, der eine reine ASCII-Datei erzeugen kann. Danach greift dann der „Document Drafting Processor" auf das Modell-Dokument zu, das der Benutzer beim Programmstart benennt.

3. Ergebnis Nr. 1: Das individuelle Dokument

Als erstes Ergebnis produziert ein Programmlauf des „Document Drafting Processor" ein individuelles Dokument, das auf Grund der Benutzereingaben vor dem Hintergrund des Modells erstellt worden ist. Im Falle unseres einfachen Testaments sähe ein beispielhaftes Ergebnis wie folgt aus:

Das Beispiel bedarf keiner weiteren Erläuterung; sein Inhalt erschließt sich problemlos durch einen Vergleich mit dem zuvor behandelten Modell-Dokument.

4. Ergebnis Nr. 2: Die Mandanten-Datei

In der Mandanten-Datei werden die Ergebnisse eines Programmlaufs protokolliert. In dem Beispiel des eben abgedruckten Testaments stehen in dieser vorgangsbezogenen Mandanten-Datei die folgenden Eintragungen:

```
Die Mandanten-Datei "TESTC"

name: the testator's name =
<Test A. Client>

address: the city and state where the testator lives =
<Chicago, Illinois>

married: the testator is married  =
<Y>

h or w: husband or wife, the proper term for the testator's spouse =
<wife>

sp name: the name of the testator's spouse =
<Mrs. Test A. Client>

he or she: he or she, the proper pronoun for the spouse  =
<she>

exec: the name of my executor  =
<Mr. Any Executor>

no bond: the executor need not post bond  =
<Y>

no accts: the executor need not file any accounts  =
<N>
```

```
Das individuelle Dokument "TWILL"

                Last Will And Testament
                        Of
                   Test A. Client

I, Test A. Client, of Chicago, Illinois, revoking all prior wills and
 codicils, do hereby make this my last will and testament.

I leave all of my property, both real and personal, to my wife, Mrs. Test
 A. Client, if she survives me by more than 30 days.  Otherwise I leave
 all of my property, both real and personal, to my children per stirpes.

I hereby appoint Mr. Any Executor executor of my estate.  My executor
 need not post security on the probate bond.

Signed by me before witnesses this 21st day of January, 1986.

                                  _____________________________
                                  Test A. Client

Witnessed:

                                  _____________________________

                                  _____________________________
```

Wiederum sind besondere Erläuterungen nicht notwendig: Verfolgt man den in der Modell-Datei programmierten Dialog-Verlauf und liest man die hier protokollierten Angaben mit, so sieht man, wie Schritt für Schritt das als Ergebnis ins Auge gefaßte juristische Dokument entsteht.

Eine Datei wie diese Protokoll-Datei ist eine geeignete Grundlage für die oben als wichtiges Desiderat bezeichnete Konsistenzprüfung der Variablenbelegung. Geeignete Organisation der Variablen vorausgesetzt sind nicht alle Antworten kompatibel. Beispielsweise wäre das Pronomen „he" nicht kompatibel mit „wife" als Bezeichnung für die Ehefrau. Der „Document Drafting Processor" sieht solche Konsistenzkontrollen nicht vor. Er bietet aber Organisationsmöglichkeiten für das in Dokumenten zu verarbeitende Wissen, die das „Aufsetzen" einer Konsistenzkontrolle ohne allzu großen Aufwand möglich machen.

H. Der „Design"-Hintergrund des „Document Drafting Processor"

Die Arbeiten von James A. Sprowl standen in Verbindung mit den Projekten am „Center for Computer Assisted Learning and Instruction" (CCALI) der Universität von Minnesota. Auch dort schrieb man (gleichfalls in PASCAL) für Lernzwecke Programme, die innerhalb einer bestimmten (hier:) Lern-Sequenz an markierten Stellen Variablen-Belegungen abfragen und dann entsprechend zu bestimmten Aktionen weiterverzweigen. Die strukturelle Nähe beider Aufgabenstellungen ist deutlich. Zugleich bildet die Verbindung der Entwicklung juristischer Lernprogramme mit anwendungsorientierten Dokumentenerstellungs-Programmen in einer einheitlichen Entwicklungslandschaft ein Paradigma, das Lernen mit Berufspraxis verbindet. Was die EDV-Umgebung angeht (und vielleicht nicht nur diese Umgebung) sollte der Jurist in einer Umgebung lernen, die der Umgebung vergleichbar ist, in der er später berufspraktisch arbeitet.

I. Ausblick

Inzwischen ist die Software-Entwicklung auf dem Gebiet der computerunterstützten Dokumentenerstellung (und mit ihr auch der Entwicklungsstand des „Document Drafting Processor") weiter fortgeschritten. Festgehalten wurde aber die oben vorgestellte Grundidee des „Document Drafting Processor", der deswegen auch in seiner Grundversion (gerade wegen der Einfachheit, mit der dort die Grundidee umgesetzt wurde) seinen bleibenden pädagogischen Wert behält. Programmsysteme wie der in der kanadischen Kanzlei Lang und Mitchener (Toronto/Ontario) von Bruce Lewis konzipierte „Document Modeler[5] gehen ebenfalls von dem „Wissensmodell" aus, das juristische Texte als Aggregat differenzierter Untereinheiten sieht, deren Zusammensetzung von Variablenbelegungen gesteuert wird. Hinzu kommen dann Verfeinerungen wie eine script-basierte Ablaufsteuerung mit Regelkomponenten, die Einblendung von Hilfs- und Kommentartexten, numerische Felder mit Berechnungsmöglichkeiten und vieles mehr. In Richtung derartiger Verfeinerungen sind noch viele Schritte denkbar. Das zugrundeliegende Wissensmodell dürfte aber für juristische Dokumente derart auskonturiert sein, das es sich als dauerhaft erweisen könnte. Die Entwicklungsarbeit verlagert sich damit für die absehbare Zukunft auf die Herauspräparierung der einschlägigen Dokumenten-Architektur und die Integration derartiger „Architekturen" in Programmsysteme der vorgestellten Art. Wenn man stattdessen undifferenziert „Textverarbeitung" an den juristischen Arbeitsplatz bringt, riskiert man (vergleiche das Eingangszitat), dort ein Arsenal von Hämmern anzulegen, wo vielleicht Schraubenzieher angebrachter wären.

5) Dieses Programmsystem ist für „Windows" programmiert. Es wird unter dem Warenzeichen „Legalware" vermarktet. jur-PC wird es in einem gesonderten Beitrag ausführlich vorstellen.