Syntaxgesteuerte Dokumentenerstellung mit DocEdit

Alois Heinz* Karsten Meyer†

Zusammenfassung

Die Beobachtung, daß fast alle in verschiedenen Bereichen erstellte Dokumente nach formalisierbaren Regeln aufgebaut sind, führt recht bald zu der Suche nach geeigneten Werkzeugen zur Unterstützung der Dokumentererstellung. Wir beschreiben DocEdit, einen Editor zur interaktiven Erstellung strukturierter Dokumente, der durch eine Dokumenttypsyntax gesteuert dem Benutzer nur die Erstellung korrekter Dokumente erlaubt. DocEdit wurde im Rahmen des von der DFG gef rderten Projektes Dokumentenpublikation[1] implementiert, in dem ein System zur rechnergestützten Produktion und Publikation von Dokumenten aufgebaut wird. Wir haben Erfahrungen mit DocEdit gesammelt z. B. bei der Erstellung von Inputdateien für das Textsatz- und Formatiersystem TEX.

1 Einleitung

Solange, wie unsere (schriftliche) Überlieferung zurückreicht, interessieren sich Menschen für Techniken, ihre Gedanken, Ideen

*Institut für Informatik der Albert-Ludwigs-Universität Freiburg

†Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe (TH)

[1]DFG: (Sto 167/1-1)

und Erlebnisse dauerhaft festzuhalten und zu verbreiten. Wie wir heute wissen, war die Erfindung der (Laut-) Schrift eine langandauernde und schwierige Geburt. Und erst mit Gutenbergs Erfindung war es möglich, Schriften und Nachrichten in nennenswerten Auflagen zu verbreiten. Bis weit in die Mitte unseres Jahrhunderts war Handsatz die weitaus gebräuchlichste Methode, Schriftbilder in gleichförmigen Lettern für den Druck zu erzeugen; das Layout wurde durch Umbruch von Hand bestimmt. Zuerst der Maschinensatz in Blei, dann der Photosatz und Lichtsatz revolutionierten ein ganzes Gewerbe, veränderten Berufsbilder und führten zu sozialen Spannungen.

Die Revolution geht weiter: Die Fortentwicklung bei Personal-Computern stellt hohe Rechenleistung und Speichermöglichkeiten preiswert zur Verfügung. Neue Gerätegenerationen bei Laserdrukkern und Scannern ermöglichen qualifizierte Druckausgabe sowie schnelle Erfassung von Text- und Bildmaterial zur Weiterverarbeitung. Ein großes Angebot an Software zur Erstellung von Text und Graphik und Layoutgestaltung forciert den Einsatz der neuen Hardware. Die Druckerei auf dem Schreibtisch ist Realität geworden — und „Desktop Publishing ist das Thema der Saison" [15].

Zeiteinsparung und Kostensenkung sind die primären Argumente für den Einsatz von Desktop Publishing, doch gibt es noch eine Reihe wichtiger Gründe, die für den Einsatz sprechen, wie direkte Einflußnahme des Autors auf das Endprodukt oder Datenübernahme aus anderen Programmen, etwa Tabellenkalkulationen. Deshalb hat Desktop Publishing auch schnell Zugang gefunden in kleineren Verlagen, Werbeagenturen, Unternehmensabteilungen und Universitäten. Es werden Broschüren, Handbücher, Inhouse-Dokumentationen und Berichte aller Art schnell erstellt.

Doch die neue Welle hat auch Nachteile mit sich gebracht: Der Autor eines Dokumentes zeichnet nicht nur für den Inhalt, sondern auch für das Layout verantwortlich; dabei fühlt er sich oft alleingelassen. Wohl kann er aus sehr vielen Optionen (z. B. Schrifttypen und -graden) auswählen, doch fehlt ihm das Wissen, wie ein ansprechendes Layout erstellt werden kann. Es gibt kaum Möglichkeiten zur Konsistenzunterstützung. Z. B. sollten alle Überschriften eines Artikels gleich gesetzt sein, alle Berichte einer Reihe sollten gleiches Format haben. Manche Autoren lassen sich auch vom optischen Eindruck zu sehr gefangennehmen, sie formatieren ständig neu und um, wobei der Inhalt schon mal ins Hintertreffen gerät.

Für die meisten Layoutprogramme werden Text und Graphik zunächst getrennt erfaßt. Für den Text bedeutet das, daß er zunächst total strukturlos ist, obwohl der Autor eine Struktur erstellen will, etwa Kapitelüberschrift, Kapiteltext, Unterkapitelüberschrift etc. Diese Struktur wird erst nachträglich beim Layout übergestülpt, wobei aber keine Konsistenzprüfung stattfindet.

Ein Nachteil (vielleicht) minderer Art ist, daß man einigen Publikationen, ob Geschäftsbericht oder Werbebroschüre, schon ansieht, mit welchem Programm sie gestaltet wurden, weil sie alle irgendwie gleich aussehen, die gleichen Schrifttypen verwenden, die gleichen Trennfehler machen.

Um Autoren die Erstellung von Publikationen wesentlich zu erleichtern, wurde im Juli 1984 am Institut für Angewandte Informatik und Formale Beschreibungsverfahren der Universität Karlsruhe das Projekt „Dokumentenpublikation“ gestartet[2], dessen Ziel die Entwicklung eines Systems zur rechnergestützten Produktion und Publikation von (insbesondere technischen) Dokumenten ist. Die wichtigsten Komponenten des Systems sind ein Datenbanksystem zur Unterstützung der Dokumentenverwaltung, ein Formatier- und Druckdienst zur Dokumentausgabe (auf Laserdrucker, Lichtsatzanlagen), ein Dokumenteneditor und ein Telekommunikationsanschluß, der u. a. das elektronische Publizieren ermöglicht.

[2] Dieses Projekt wird nun am Institut für Informatik der Albert-Ludwigs-Universität Freiburg fortgesetzt.

Das Projekt und seine Komponenten sind in mehreren Veröffentlichungen vorgestellt worden [2, 3, 4, 5, 9]. Einige Teile wurden sowohl unter UNIX-Umgebungen als auch für DOS-Rechner entwickelt. Wir berichten hier über die Implementation eines Prototyps eines syntaxgesteuerten Dokumenteneditors mit ansprechender Benutzeroberfläche für DOS-Rechner.

Im nächsten Kapitel geben wir zunächst eine Gegenüberstellung der WYSIWYG- und Batch-Philosophie und begründen unsere Entscheidung für letztere. Wir erklären die Begriffe Dokumentstruktur, Dokumenttyp und Dokumentensyntax. Im 3. Kapitel werden dann Eigenschaften und Möglichkeiten des Dokumenteneditors DocEdit ausführlich erläutert. Im letzten Kapitel geben wir eine Zusammenfassung unserer Erfahrungen beim Einsatz von DocEdit.

2 Formatierung — WYSIWYG versus Batch

Wir wollen zunächst kurz definieren, was wir unter einem Dokument verstehen: In unserem Zusammenhang ist ein Dokument eine Menge von Informationen, die eine feste graphische Repräsentation auf einer geordneten Folge von Seiten besitzt. Je nach Ausgabemedium kann diese Repräsentation leicht (etwa in der Auflösung) variieren. Ein Dokument ist also immer formatiert.

Bei der Erstellung einer Publikation kann man drei Prozesse unterscheiden, das Editieren (Eingeben) von Text und Graphik, das Formatieren (Layoutgestaltung) und die Ausgabe. Es gibt sogenannte Batch-Systeme, bei denen diese Prozesse streng getrennt hintereinander ablaufen. Andere Systeme, bei denen zumindest das Formatieren und die Ausgabe am Bildschirm gleichzeitig (oder doch verzahnt) erfolgen, nannt man WYSIWYG-Systeme (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et).

Zunächst muß man anerkennen, daß WYSIWYG-Systeme große Vorteile bieten: Während man das Ergebnis einer Formatierung am Bildschirm begutachtet, kann man Änderungen anbringen, deren Auswirkungen wieder (fast) sofort sichtbar sind. Der Zyklus Formatieren-Ausgeben-Inspizieren-Ändern-Neuformatieren kann sehr schnell interaktiv durchlaufen werden. Sicher sind diese Systeme dort richtig eingesetzt, wo (fast) jedes erstellte Dokument ein andersartiges Layout hat und wo der Benutzer Erfahrungen im Erstellen eines Layouts hat, wie etwa in der Werbebranche.

Aber die Dokumente dürfen nicht zu groß werden (einige Systeme erlauben nur bis zu 16 Seiten gleichzeitig), weil dann die Übersicht verlorengeht. Die Dokumente müssen also in handhabbare Teile zerlegt werden.

Weitere Nachteile werden schnell sichtbar: Die Intradokument-Konsistenz („sind alle Abschnittsüberschriften gleich gesetzt?") und Interdokument-Konsistenz („entspricht der Bericht dem Format der Berichtsreihe?") sind nicht automatisch überprüfbar; der Autor ist allein dafür verantwortlich. Eine Umformatierung (z. B. nötig bei der externen Veröffentlichung des zunächst intern erstellten Berichtes) kann sehr umständlich sein (man muß z. B. alle Abschnittsüberschriften neu formatieren).

Es ist klar, warum man mit WYSIWYG-Systemen diese Probleme hat: Sie kennen nur die Layout-Ebene, während das Dokument für den Autor doch Struktur hat. Sein Buch etwa besteht aus Inhaltsverzeichnis, Kapiteln und Literaturverzeichnis, jedes Kapitel hat eine Kapitelüberschrift, eventuell einen Kapiteltext und mehrere Abschnitte.

Bei Batch-orientierten Systemen, wie etwa **Troff** [16], **Scribe** [17] und TeX [11] werden beim Editieren des Textes Kommandos (Markup) eingefügt, die die spätere Formatierung steuern. Verzichtet man dabei auf prozedurale Kommandos (neue Zeile, Kursivschrift, 12 Punkt) und wählt statt dessen deskriptive Kommandos (der folgende Text ist eine Abschnittsüberschrift) so erhält man die sogenannte logische Auszeichnung, die auf C. F. Goldfarb, E. J. Mosher und R. A. Lorie [8, 6, 7] zurückgeht.

Diese Auszeichnung beschreibt die Dokumentenstruktur und enthält keine prozeduralen Teile. Sie kann verschiedenen Formen der Weiterverarbeitung dienen und sollte formalen Gesetzen entsprechen, so daß die Techniken der Weiterverarbeitung sich nicht von denen für andere formal definierte Objekte zu unterscheiden brauchen.

Die logische Auszeichnung liegt auch dem SGML-Vorschlag (**S**tandard **G**eneralized **M**arkup **L**anguage) der ISO [10] zur Standardisierung von Dokumenten zugrunde. Nach SGML haben Dokumente eine im wesentlichen hierarchische Struktur. Der Dokumenttyp des oben erwähnten Buches (seine *generisch-logische Struktur*) kann in SGML in der Art einer kontextfreien Grammatik beschrieben werden:

```
<!STRUC Buch    (Inhaltsverzeichnis, Kapitel+,
                 Literatur?)>
<!STRUC Kapitel (Ueberschrift, Eingangstext?,
                 Abschnitt*)>
```

Ein Autor, der ein Buch schreibt, identifiziert dessen Elemente und markiert Beginn und Ende jedes Elementes mit einem mnemonischen Namen (*generischen Bezeichner*). So kann ein spezielles Dokument vom Dokumenttyp Buch, also eine *spezifisch-logische Struktur* in SGML folgendes Aussehen haben:

```
<Buch>
<Inhaltsverzeichnis>
  ... Text zum Inhaltsverzeichnis ...
</Inhaltsverzeichnis>
<Kapitel>
<Ueberschrift>
Dieses ist die Ueberschrift des ersten Kapitels
```

```
</Ueberschrift>
<Eingangstext>
Hier beginnt das erste Kapitel ...
</Eingangstext>
<Abschnitt>
Und dies ist der erste Abschnitt ...
</Abschnitt>
</Kapitel>
<Kapitel>
</Kapitel>
</Buch>
```

Hier ist nur noch der Inhalt und die logische Struktur des Dokumentes wiedergegeben, die *inhaltslogische Struktur* (das „was?“). Die Verarbeitungsinformationen (Makrodefinitionen), die z. B. angeben, daß ein Kapitel auf einer neuen Seite beginnt, ein betontes Wort kursiv geschrieben wird (das „wie?“), sind völlig getrennt davon.

Die Vorteile der Batch-orientierten Formatierung sind nun deutlich: Inhaltsstruktur und Layoutprozeduren sind getrennt; wurden die Layoutprozeduren einmal (von einem Experten) erstellt, kann der Autor sich immer wieder darauf beziehen. Andererseits kann das Layout geändert werden ohne die Inhaltsstruktur zu verändern. Beim Editieren kann der Autor in Strukturen denken und Strukturen erstellen und ändern. Er oder sie ist nicht mehr mit Layoutfragen beschäftigt.

Manche sehen den zeitlichen Abstand zwischen Editieren und Prüfen des formatierten Ergebnisses als Nachteil an, doch mit den heute verfügbaren schnellen Formatierern und Preview-Programmen (das formatierte Dokument kann am Bildschirm betrachtet aber nicht direkt geändert werden) dürfte das kaum noch ein Problem darstellen.

Ein Vorteil der Batch-Formatierer ist, daß sie (immer noch)

Dokumente höherer typographischer Qualität erzeugen. Als Beweis mag vielleicht die Aufzählung einiger Eigenschaften des von D. Knuth entwickelten TEX dienen, das bei uns im Formatierdienst eingesetzt wird: Trennungsalgorithmus an jede Sprache anpaßbar, flexibler Zeilenumbruch, geräteunabhängige Ausgabe, Erkennung von Ligaturen und Unterschnitt, beliebig viele Schriftarten, Blocksatz, variabler Flattersatz, Formsatz, Zentrierung, Tabellensatz, Mehrspaltensatz, Satz komplizierter mathematischer Formeln, automatische Kapitel- und Abschnittsnumerierung, laufende Kopfzeilen (dabei Unterscheidung zwischen geraden und ungeraden Seiten), Numerierung und Plazierung von Fußnoten, integriertes Referenzierungskonzept, automatisches Aufbereiten und Setzen von Inhaltsverzeichnis und Index.

Aufgrund der großen Flexibilität haben wir uns also für Batch-Formatierer entschieden. Doch wir wollten den Autor nicht mit einer Dokumenttypdefinition und einem Texteditor alleinlassen. Denn die Vermischung von Information und Metainformation auf Textebene erschwert inhaltliche wie strukturelle Bearbeitung eines Dokumentes, selbst wenn man von der Gefahr von Syntax- und Tippfehlern in den strukturbezogenen Teilen absieht.

Im folgenden Kapitel beschreiben wir Konzeption und Implementation des durch eine Dokumenttypsyntax gesteuerten Editors DocEdit, mit dessen Hilfe ein Autor nur noch korrekt ausgezeichnete Dokumente (Inhaltsstrukturen) erstellen kann.

3 Konzept und Implementation von DocEdit

DocEdit ist ein syntaxgesteuerter bildschirmorientierter Editor für Inhaltsstrukturen, der unter MS-DOS oder PC-DOS ab Version 3.0 lauffähig ist (auf beliebigen Bildschirmkarten). Während der Benutzer am Bildschirm nur die baumartige Struktur oder den Textinhalt der Knoten editieren kann, wird bei der Ablage ein Textfile erzeugt, das zusätzliche Auszeichnungsinformation (Markup) enthält und für einen Formatierer als Kommentar gekennzeichnete Strukturinformation. Dieses Textfile kann später wieder gelesen und neu editiert werden.

Der Begriff Formatierer darf hier nicht zu eng verstanden werden. Denn mit geeigneter Syntaxbeschreibung ist man mit DocEdit ohne weiteres in der Lage, z. B. Programme, etwa in PASCAL, syntaktisch korrekt zu erzeugen.

3.1 Die Benutzeroberfläche

DocEdits Benutzeroberfläche ist sehr komfortabel und ermöglicht auch absoluten Laien das Erstellen von Dokumenten. Beim Neuanlegen eines Dokumentes kann man in einem Menü die Auswahl des gewünschten Dokumenttyps treffen, beim Editieren schon angelegter Dokumente wird der Dokumenttyp automatisch erkannt.

Im *Editiermodus* zeigt die erste Bildschirmzeile die Überschriften der Pull-down-Menüs, eine Information zum Hilfe-Einstieg und die aktuelle Uhrzeit. Die zweitletzte Zeile ist die Statuszeile mit der Anzeige des Dokumentnamens, des Namens des Funktionstastenuntermenüs, der Spalten- und Zeilenposition des Cursors und der Größe des freien Speicherplatzes. Die letzte Zeile zeigt die aktuelle Funktionstastenbelegung (F1 bis F10) und einige Statusflags (z. B. Insert ein oder aus) an. Hier werden auch eventuelle Fehlermeldungen eingeblendet.

Auf der übrigen Bildschirmfläche werden die Knotentexte dargestellt, der Reihenfolge im Syntaxbaum (siehe Abschnitt 3.2) entsprechend, jeweils durch eine spezielle Anzeigezeile getrennt. In diesem Modus kann der Benutzer wie in einem gewöhnlichen Editor Texte editieren, wobei alle üblichen Editorbefehle angeboten werden und über die hierarchisch organisierten Funktionstastenmenüs erreichbar sind (z. B. Text suchen und ersetzen, Text aus externen

Files einlesen). Spezielle Funktionen, z. B. Auswahl eines Schrifttyps, können über den Einstieg in die Pull-down-Menüs angewählt werden. Außerdem können alle von WordStar[3] her bekannten Tastenkombinationen (Controlsequenzen) benutzt werden und haben die von dort bekannte Wirkung mit wenigen Einschränkungen. Z. B. ist das Markieren eines Blocks über Knotengrenzen hinweg nicht möglich.

Mit einer Funktionstaste schaltet der Benutzer in den *Konzeptmodus* um. Ein Fenster legt sich über einen Teil des Dokumentes und es erscheint die Liste der Knotennamen in Präfixordnung (erst Wurzel, dann die Subknoten) und entsprechend der Hierarchiestufe eingerückt. Der aktuelle Knotenname ist unterlegt, zunächst der, in dem sich der Cursor zuletzt befand. Mit Hilfe der üblichen Cursor- und Page-Tasten kann ein neuer Knoten aktuell gemacht werden; dies kann auch ein noch nicht angelegter Knoten sein. Noch nicht angelegte Knoten sind wieder speziell unterlegt und werden dort angezeigt, wo sie erzeugt werden können. Über die Tasten „-“ und „+“ können Knoten geschlossen (die Subknoten werden im Konzeptmodus nicht mehr angezeigt) und geöffnet werden. Auch die Return-Taste öffnet Knoten, legt noch nicht angelegte Knoten neu an und schaltet bei Knoten, die Text enthalten können, in den Editiermodus zurück. Dort steht der Cursor dann in der ersten Zeile des entsprechenden Knotentextes.

Dies mag für eine grobe Übersicht über die Benutzerschnittstelle genügen, für weitere Details verweisen wir auf [13, 14].

3.2 Die Dokumenttypbeschreibung

Jeder Dokumenttyp wird in einer Dokumenttypbeschreibung spezifiziert. Das Erstellen von Dokumenttypbeschreibungen ist eigentlich eine Aufgabe für Spezialisten, die u. a. über gute Kenntnisse

[3] WordStar ist ein Warenzeichen der MicroPro International Corporation.

des Formatierers verfügen müssen. Um das Editieren von Dokumenttypbeschreibungen wirkungsvoll zu unterstützen, werden sie selbst als Dokumente mit DocEdit erstellt und sind vom Dokumenttyp `DOCEDIT.DTB` (alle Dokumenttypbeschreibungen haben die Extension `.DTB`). Zu diesem Verfahren vergleiche man auch [5].

Zunächst wird in einer Art BNF die Dokumentensyntax spezifiziert. Jedes Dokument hat genau einen Wurzelknoten. Ein Knoten kann weitere Subknoten als Unterstruktur enthalten oder aber ein Textknoten (Blatt des Baumes) sein. Die Anzahl des Vorkommens von Subknoten kann über Präfixe gesteuert werden (kein Präfix: genau einmal, „?“: null- oder einmal, „*“: beliebig oft, „+“: mindestens einmal). Die Definition kann Alternativen und Rekursionen enthalten. Für Textknoten kann spezifiziert werden, welche sogenannten Inline- bzw. Intextknoten an beliebiger Stelle im Text vorkommen dürfen (wichtig z. B. für Schriftenauswahl, Abbildungen oder Aufzählungen). Z. B. können folgende Zeilen in einer Dokumenttypspezifikation stehen:

```
Unterkapitel ::= Ueberschrift ?Text *Unterkapitel.
Abbildung    ::= [Tabelle | Graphik].
Text         ::= #text.
```

Dann folgt für jeden spezifizierten Knoten eine Folge von Anweisungen, die das Editieren und das Auszeichnen des entsprechenden Knotens steuern. Zum Editieren können z. B. linker und rechter Rand, das Insert-Flag und die Anzeigezeile im Editiermodus gesetzt werden. Es kann ein Default-Knotentext angegeben werden, der automatisch in einem neu erzeugten Knoten eingetragen wird.

Zum Auszeichnen werden die Knotenbeginn- und -endemarkierung angegeben. Diese können auch einzeln entfallen. Es ist möglich, ein Dokument neu zu editieren, nachdem die Auszeichnungsinformation des Dokumenttyps geändert wurde. Beim erneuten Abspeichern enthält es dann die neue Auszeichnung. So kann man ohne Probleme eine Inhaltsstruktur auch für verschiedene Formatierer auszeichnen, indem man Dokumenttypdefinitionen bereithält, die im Syntaxteil identisch, jedoch im Auszeichnungsteil unterschiedlich sind.

Weiter enthält die Dokumenttypbeschreibung eine Translationstabelle für einzelne Zeichen und Zeichengruppen. Dies ist wichtig, weil einige Sonderzeichen und nationale Zeichen (z. B. Umlaute), die über die Tastatur erreichbar und auf dem Bildschirm darstellbar sind, für manche Formatierer in spezielle Kommandosequenzen umgesetzt werden müssen. Mit Hilfe dieser Tabelle geschieht das, ohne daß der Benutzer etwas davon merkt.

Ein zwar nicht notwendiger aber doch sehr wichtiger Teil ist die Liste von Hilfeeinträgen. Hier kann für jeden Knoten eine spezielle hierarchisch geschachtelte Hilfe angegeben werden, die im Editor sowohl im Editier- als auch im Konzeptmodus aufgerufen werden kann. Ein Benutzer kann sich also spezifische Informationen zu einem Knoten geben lassen, die u. U. im Detailierungsgrad gestaffelt sind. Ein Hilfeeintrag enthält Parameter zur Knotenzugehörigkeit, Position und Größe des Hilfefensters auf dem Bildschirm, eventuellen Untereinträgen und natürlich den Hilfetext. Dokumenttypdesigner sollten ein spezielles Augenmerk auf die gute Dokumentation des Dokumenttyps in diesem Hilfesystem richten.

3.3 Hilfesysteme

DocEdit verfügt über drei Hilfesysteme, die jeweils hierarchisch geschachtelt sind, d. h. man kann von einigen Hilfefenstern zu Subfenstern verzweigen und wieder zurückkehren und so einen Hilfebaum durchwandern.

Das knotenbezogene und damit dokumenttypabhängige Hilfesystem wurde bereits in Abschnitt 3.2 erklärt. In der Kopfzeile des Editors findet der Benutzer einen Hinweis über den Einstieg in ein allgemeines Hilfesystem, auch DocEdit-Tutor genannt. Hier werden Philosophie, Funktions- und Arbeitsweise von DocEdit ganz genau vom groben Konzept bis detailliert zu einzelnen Befehlen erklärt. Der Tutor ist insbesondere für neue Benutzer gedacht und kann als Online-Benutzerhandbuch angesehen werden.

Ein drittes Hilfesystem ermöglicht einen situationsbezogenen Einstieg in den DocEdit-Tutor und erläutert die genauen Funktionen der im Moment aktivierbaren Funktionstasten oder Menüs. Der fortgeschrittene Benutzer wird diese Hilfe nur anstoßen, um sich seltener benutzte Funktionen erklären zu lassen.

Wir glauben mit diesem dreigeteilten, abgestuften Hilfesystem allen Benutzern gezielte Hilfe geben und dabei auf ein Handbuch gänzlich verzichten zu können. Nochmals verweisen wir auf die Verantwortung des Dokumenttypdesigners für die Erstellung der knotenbezogenen Hilfe.

3.4 Benutzerspezifische Anpassungen

Jeder Benutzer eines Editors weiß es zu schätzen, wenn dieser auf seine speziellen Bedürfnisse angepaßt werden kann. Oft benötigt man immer wiederkehrende Kommandosequenzen oder schreibt ein Dokument, in dem lange oder schwierige Worte sehr häufig vorkommen.

DocEdit wird beim Aufruf durch Angabe eines Profiles benutzerspezifisch umkonfiguriert. Profiles (mit der Extension `.PRF`) sind selbst Inhaltsstrukturen des Typs `PROFILE.DTB`. Sie werden mit DocEdit erstellt und können auch während einer Sitzung verändert werden.

Durch das Profile werden einigen Tasten (Alt-Tasten) Makros zugeordnet. Ein Makro ist in diesem Zusammenhang eine Folge beliebiger Tasten, also auch Funktionstasten, mit Ausnahme der Alt-Tasten. Rekursive Makros sind also nicht gestattet.

Eine weitere Möglichkeit sind Textabkürzungen. In einer beliebig langen Tabelle können Abkürzungen und ihre Expansionen definiert werden, wobei der Punkt (.) als aktive Taste eine Abkürzung beendet. Beim Schreiben dieses Textes z. B. wurde die Abkürzung „dt.“ verwendet, die sofort in „Dokumenttyp“ expandiert wurde.

3.5 Implementation

DocEdit wurde im Rahmen von Studien- und Diplomarbeiten implementiert [1, 13, 14], wobei die aktuelle Implementation auf der Turbo Editor Toolbox von Borland International aufbaut. Die Toolbox beinhaltet alle Datenstrukturen und Prozeduren zur Erstellung von Texteditoren im Quellcode. Die Datenstruktur zur internen Verwaltung des Textes ist dabei eine doppelt verkette Liste von Zeilen.

Wir haben den gesamten Quellcode allerdings in Microsoft Pascal übertragen zur Umgehung von Speicherplatzbeschränkungen und Erreichung besserer Modularität (Unit-Konzept). Die Implementation der DocEdit-spezifischen Teile, insbesondere der Strukturverwaltung, erforderte ein Höchstmaß an Aufmerksamkeit. Durch den Einsatz der Toolbox brauchten wir uns jedoch um die rein textbezogenen Funktionen kaum noch zu kümmern, was unserer Kreativität bei den Erweiterungen sehr zugute kam.

4 Anwendererfahrungen

Die Mitglieder des Instituts für Informatik und des Institutes für Angewandte Informatik und Formale Beschreibungsverfahren haben weit zurückreichende Erfahrung im Bereich der Dokumentenerstellung. Dabei kommt heute meist TEX oder das auf TEX aufbauende Makropaket LATEX [12] als Formatierer zum Einsatz.

Bereits recht früh haben wir Vorläuferversionen von DocEdit mit eingeschränktem Funktionsumfang allen interessierten Institutsmitgliedern zur Verfügung gestellt. Dies erwies sich als großer

Vorteil, denn obwohl anfangs noch ein Engpaß an geeigneten Dokumenttypbeschreibungen bestand, wurde DocEdit angenommen. Und aus Rückmeldungen und Diskussionen mit den Benutzern ergab sich mancher Verbesserungsvorschlag. Die Verfügbarkeit von DocEdit führte auch zur Erstellung von Dokumenttypbeschreibungen durch einzelne Anwender, deren Dokumente dadurch, sofern sie zum gleichen Typ gehören, in der Gestaltung einheitlicher wurden.

Dieser Artikel wurde mit DocEdit auf einem IBM AT geschrieben (Dokumenttyp `ARTIKEL.DTB`) und mit LaTeX formatiert[4]. Mit einem Preview-Programm[5] konnte das Ergebnis sofort am Bildschirm begutachtet werden. Die Ausgabe erfolgte auf einem Laserdrucker (Cordata LP300) in einer Auflösung von 300 Punkten pro inch.

Danksagung:

DocEdit ist das Ergebnis von Teamarbeit. Institutsmitglieder und Studenten hatten großen Anteil an Spezifikation und Implementation. Besonderen Dank schulden wir Herrn Peter Dolland.

Literatur

[1] Jürgen Bischoff: *DOCEDIT, Ein syntaxgesteuerter Dokumenteneditor*, Studienarbeit, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, 1987.

[2] Anne Brüggemann-Klein: *TeX-Treiber für Lichtsetzanlagen*, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, Bericht 171, 1986.

[3] Anne Brüggemann-Klein, Peter Dolland, Alois Heinz: *How to please Authors and Publishers: A Versatile Document Prepa-*

[4] Die PC-Portierung des TeX-Formatierers stammt von MicroTeX.

[5] TeXview vom RRZN Universität Hannover

ration System at Karlsruhe, in: Jacques Désarménien (ed): TEX for Scientific Documentation, Second European Conference, Straßburg, Juni 1986, Lecture Notes in Computer Science 236, 1986.

[4] Anne Brüggemann-Klein, Peter Dolland, Alois Heinz, Ursula Schmidt: *Werkzeuge zur Erzeugung und Ausgabe von TEX-Files*, 2. SAVE-Tagung in Heidelberg, 1987.

[5] Peter Dolland: *Konzeption eines Dokumentenarbeitsplatzsystems*, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, Forschungsbericht Nr. 170, 1986.

[6] C. F. Goldfarb: *Document Composition Facility Generalized Markup Language: Concepts and Design Guide*, Form No. SH20-9188-0, IBM Corporation, White Plains, 1980.

[7] C. F. Goldfarb: *A Generalized Approach to Document Markup*, SIGPLAN Notices of the ACM, 1981.

[8] C. F. Goldfarb, E. J. Mosher, T. L. Peterson: *An Online System for Integrated Text Processing*, Proceedings of the American Society for Information Science, 7, 147–150 (1970).

[9] A. Heinz, K. Meyer: *DocEdit — Ein Editor zur syntaxgesteuerten Erstellung strukturierter Dokumente*, Proc. 17. GI-Jahrestagung, Technische Universität München, Informatik-Fachbericht 156, Springer Verlag, Berlin, 278–287 (1987).

[10] ISO/DIS 8879.

[11] Donald E. Knuth: The TEX-book, Addison-Wesley Publishing Co., 1986.

[12] Leslie Lamport: LATEX: *A Document Preparation System*, Addison-Wesley Publishing Co., 1985.

[13] Karsten Meyer: *DocEdit, Implementation eines strukturgesteuerten Dokumenteneditors*, Studienarbeit, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, 1987.

[14] Karsten Meyer: DocEdit — *Ein syntaxgesteuerter Dokumenteneditor*, Diplomarbeit, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, 1987.

[15] Dieter Nenner: *Druckerei der Zukunft*, CHIP, Mai 1987.

[16] J. F. Ossanna: *NROFF/TROFF User's Manual*, in: UNIX Programmer's Manual, Volume 2A.

[17] B. K. Reid: *The Scribe Document Specification Language and its Compiler*, Proceedings of the International Conference on Research and Trends in Document Preparation Systems, 59–62, 1981.