SGML — Eine Einführung

Hans Holger Rath

Zentrum für Graphische Datenverarbeitung e.V.
Wilhelminenstraße 7
D 64283 Darmstadt
Tel.: 06151 / 155 152, FAX: 06151 / 155 199
EMail: rath@igd.fhg.de

Abstract

Die Inhalte einer Einführung in SGML sind immer vom übergeordneten Kontext abhängig. Dieser Kontext ist mit dem Workshop-Titel „SGML in der Praxis“ klar auf praktische Anwendungen ausgerichtet; entsprechend werden in diesem Artikel primär die praxisrelevanten Teile von SGML und seinem Umfeld erläutert. Es beginnt mit einer Motivation des standardisierten Dokumentformates SGML. Darauf folgt ein kurzer Ausflug in die Geschichte von SGML und der Einführung der Basisbestandteile von SGML. Da SGML ein Formalismus ist, müssen bei der Anwendung verschiedene Software-Werkzeuge eingesetzt werden; diese werden vorgestellt und ihre Zusammenarbeit im Gesamtprozeß wird beschrieben. Wirklich nur einführenden Charakter hat der Überblick über die Syntax und Sematik von SGML. Der Artikel endet mit einer Zusammenfassung der Vor- und Nachteile von SGML.

1 Motivation

Die Menge an Dokumenten, sei es ein Dokument auf Papier oder ein Dokument in elektronischer Form, nimmt ständig zu. Während die Handhabung von (aus)gedruckten Dokumenten auf Papier im wesentlichen auf die Anwendung der Werkzeuge Kopierer, FAX, Schere, Kleber und Tipp-Ex beschränkt ist, erlaubt das „elektronische“ Dokument weitaus komfortablere Bearbeitungsarten. Dies trifft aber nur solange zu, wie sich die Autoren immer in der gleichen „elektronischen Umgebung“ (Software, Hardware) aufhalten. Möchte oder muß ein Autor diese Umgebung (z. B. von MS-Word nach WordPerfect) wechseln, so ergeben sich sofort kleinere oder meist größere Probleme. Diese Probleme beruhen im wesentlichen auf den unterschiedlichen Speicherformaten, in denen die Dokumente vom jeweiligen System abgelegt werden.

Die Lösung eines solchen Problems liegt auf der Hand: die Verwendung eines standardisierten Dokumentformates wie SGML.

2 Historie und Grundlagen von SGML

SGML besteht seit 1986 als internationale ISO-Standard 8879 *Information Processing — Text and office systems — Standard Generalized Markup Language (SGML)* [1], dessen Potential erst Anfang der 90er Jahre von einer größeren Anwendergruppe erkannt wurde. Die Grundidee stammt von dem IBM-Mitarbeiter Dr. Ch. F. Goldfarb. Er entwickelte 1969 die *Document Composition Facility General Markup Language* (DCF GML [2]), eine Auszeichnungssprache zur Batch-Formatierung von Dokumenten. DCF GML ist eine Makrosprache für ein Formatiersystem und daher mit Formaten wie TeX [3] bzw. LaTeX [4] und NROFF/TROFF [5] vergleichbar.

DCF GML wurde dann in dem Standardisierungsgremium ISO/IEC JTC1/SC18/WG8 zu SGML verallgemeinert und um die Möglichkeit zur Festlegung einer Grammatik für die jeweilige Dokumentstruktur erweitert. Dadurch unterscheidet sich SGML von jeder anderen Auszeichnungssprache für Dokumente. Diese Gramma-

tik für eine Dokumentstruktur nennt sich *Document Type Definition* (engl. für Dokumenttypfestlegung[1)], DTD). In der DTD werden die Reihenfolge und die hierarchischen Abhängigkeiten der für das Dokument relevanten Textteile (*Elemente*) festgelegt und damit die Tags (engl. für Markierungen) für die später zu erstellenden Dokumente definiert. Den Elementen können in der DTD *Attribute* zugeordnet werden, die Eigenschaften der Elemente repräsentieren[2)] und die auch die Querverweise in SGML realisieren.

Im SGML-Standard ISO 8879 sind die Sprachmittel und Regeln zum Aufbau von DTDs und von den nach den DTDs erstellten Dokumenten (sog. *Instanzen* der DTD) festgeschrieben. Zudem werden in einer *Reference Concrete Syntax* (engl. für Bezugslexikalik) die Zeichen der SGML-Syntax im Standard vorgegeben. Diese lassen sich aber, wenn notwendig, in einer *Variant Concrete Syntax* (engl. für variante Lexikalik) anwendungsspezifisch abändern. Daraus ergeben sich Abhängigkeiten wie in Abbildung 2–1 gezeigt.

Eine SGML-Anwendung besteht also aus einer Concrete Syntax (das ist in vielen Fällen die Reference Concrete Syntax), aus einer DTD und aus den Instanzen (den Dokumenten) der DTD. Ob alle diese Teile dem SGML-Standard entsprechen und auch zusammen passen (insbesondere die Instanzen und die DTD), kann von einem *SGML-Parser* überprüft werden.

3 SGML-Werkzeuge

Um mit SGML komfortabel arbeiten zu können, ist die Verwendung von SGML-Werkzeugen unerläßlich. Auf jeden Fall benötigt man einen *SGML-Parser*, der, wie bereits erwähnt, Concrete Syntax, DTD und Instanz

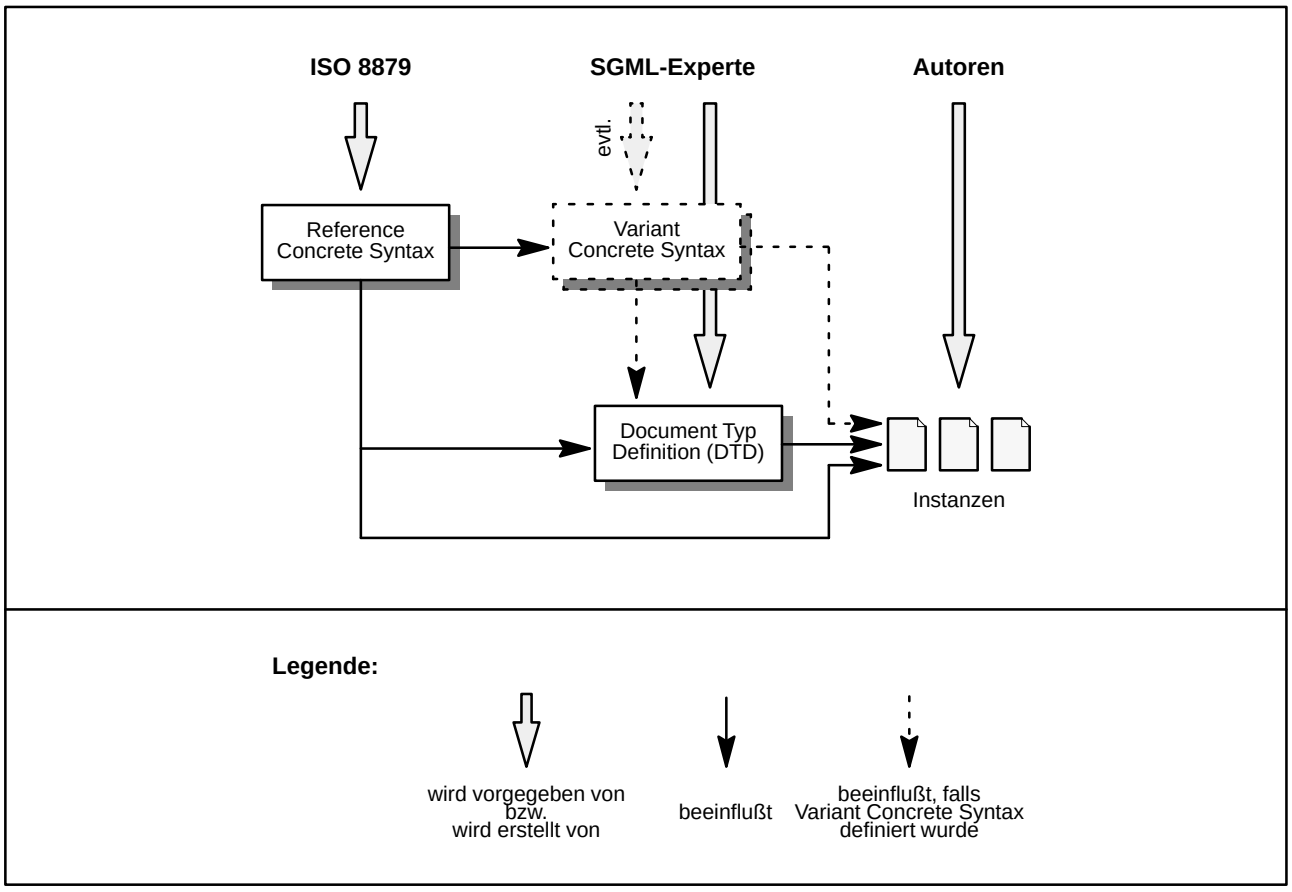


Abbildung 2–1
Formale Teile einer SGML-Anwendung

1) Alle SGML Fachbegriffe sind in DIN 28879 [6] ins Deutsche übersetzt. Da diese Übersetzungen zumeist etwas unglücklich sind und in Fachkreisen auch ungebräuchlich sind, werden sie hier nur aufgeführt, aber nicht durchgehend verwendet.

2) Z. B. ob die Unterpunkte einer Aufzählung mit einem Punkt oder einem Spiegelstrich beginnen.

überprüfen kann, und einen *SGML-Formatierer*, der die SGML-Tags der Dokumente in das gewünschte „schöne“ Format (Layout) umwandelt und z. B. auf Papier ausgibt.

Zum Erstellen der Instanzen können, da es sich um reine ASCII-Daten handelt, *ASCII-Editoren*, wie sie bei jedem Betriebssystem vorhanden sind, verwendet werden. Diese haben aber drei eklatante Nachteile: (i) jedes einzelne Zeichen der SGML-Tags muß eingetippt werden, (ii) es muß mit dem Parser immer wieder überprüft werden, ob die eingegebene Instanz auch wirklich der DTD entspricht und (iii) erst nach Aktivierung des SGML-Formatierers (als Batch-Aufruf) kann überprüft werden, ob das Dokument auch das gewünschte Aussehen hat.

Arbeitet man mit einem WYSIWYG-System[3)], das die Generierung der SGML-Tags übernimmt und den Autor damit erheblich entlasten kann, so ist dessen internes Speicherformat für die erstellten Dokumente meistens nicht SGML. Das bedeutet, daß man zur Bearbeitung von SGML-Instanzen Import- und Exportprogramme benötigt, die zwischen SGML und dem internen Format konvertieren. Wird vor dem Import bzw. nach dem Export der SGML-Parser gestartet, so spricht man von einem *Pre-* und *Postprocessing System* (siehe Abbildung 3–1).

Bei beiden bisher vorgestellten Systemen war der Parser nicht direkter Bestandteil des Werkzeuges. Er wurde dazu verwendet, die bereits gemachten Fehler zu finden. Ist der Parser aber in das Werkzeug integriert und werden vom ihm ständig alle Benutzeraktionen bzgl. der Konformität mit der jeweiligen DTD überwacht, so

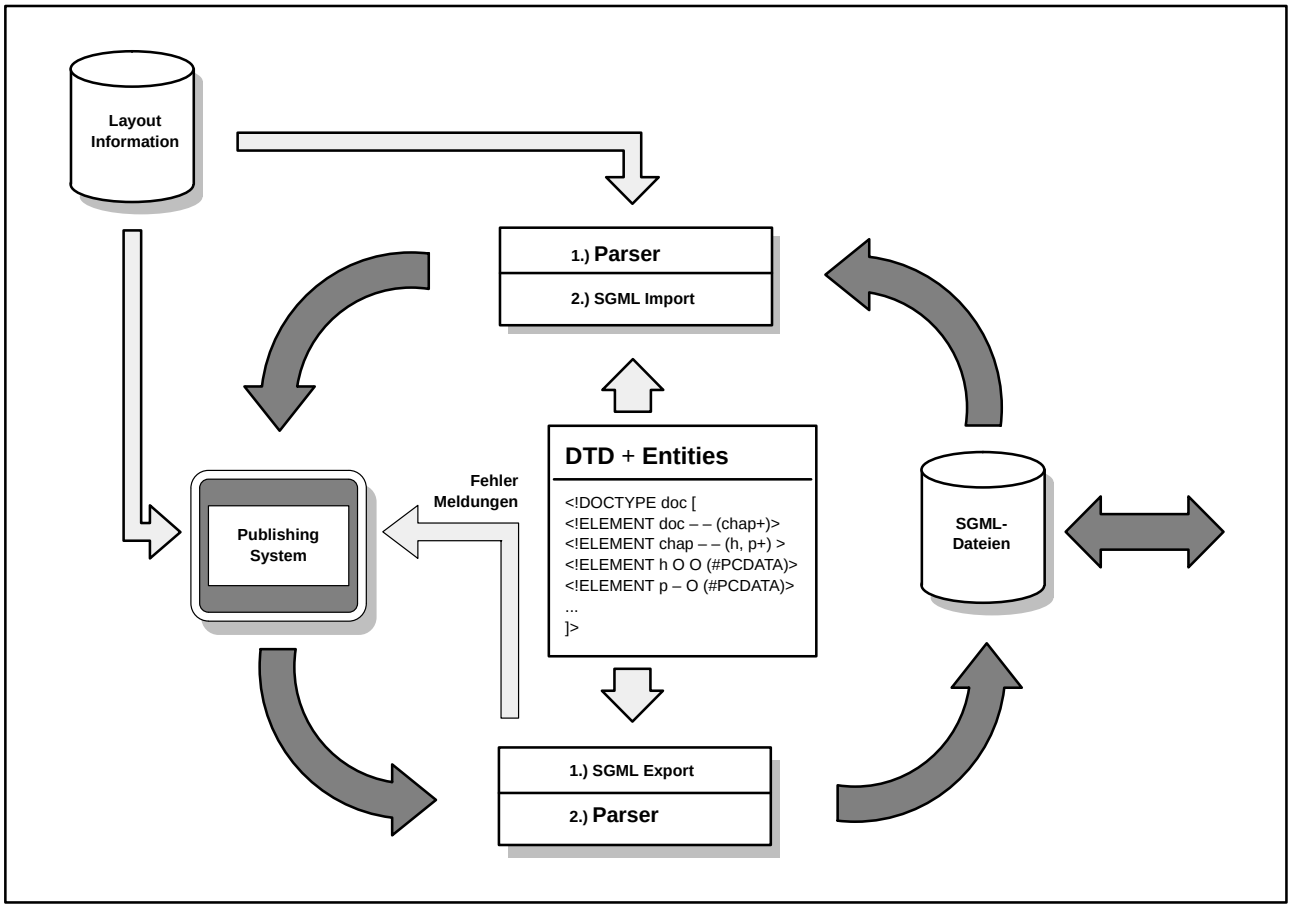


Abbildung 3–1
SGML-System mit Pre-/Postprocessing

3) WYSIWYG: What You See Is What You Get. Ausdruck für Textverarbeitungssysteme deren Bildschirmausgabe der Dokumente größtenteils mit dem späteren Papierausdruck übereinstimmt.

kann man von Fehlervermeidung statt von Fehlerbehebung sprechen (siehe Abbildung 3–2). Ein in die Software integrierter Parser sollte folgende Auswirkungen auf die Benutzerschnittstelle des Programms haben:

- Das Erzeugen von neuen SGML-Tags, die den Elementen der DTD entsprechen, ist abhängig von der aktuellen Position der Schreibmarke (Cursor). Der Benutzer *kann* nur die Elemente erzeugen, die in dem aktuellen Kontext von der DTD erlaubt sind. Daraus ergibt sich der Begriff „Kontext Sensitiver Editor“.
- Die Eingabe von Text ist nur innerhalb der Tags erlaubt, für die die Texteingabe in der DTD festgelegt wurde.
- Die Funktionen Cut, Copy und Paste (Ausschneiden, Kopieren und Einfügen) von Tags und Text laufen kontrolliert ab, so daß einerseits die vorgegebene Struktur nicht zerstört wird und andererseits immer das richtige Layout erzeugt wird.

Natürlich sind verschiedene Mischformen bei den SGML-Werkzeugen möglich. Je nach Anforderung und Preisvorstellung sind die einzelnen Systeme interessant. Ausschlaggebend für die Auswahl eines Systems ist sicherlich, ob

- der Editor kontext sensitiv arbeitet,
- er WYSIWYG unterstützt und in welcher Qualität,
- das WYSIWYG allen Layoutanforderungen der Dokumente gerecht wird,
- das Layout kontextabhängig generiert wird (z. B. ist ein Absatz in einer Aufzählung eingerückt, ein „normaler“ Absatz dagegen nicht),
- die SGML-Attribute der DTD editiert werden können,
- die Attribute bei einem WYSIWYG-System direkte Auswirkungen auf die Formatierung haben und ob
- der SGML-Editor beliebige DTDs mit jeweils eigenem Layout verarbeiten kann.

Die Erfüllung des letzten Punktes ist unablässig, wenn man eigene DTDs mit eigenem Layout in ein SGML-System integrieren will. Hier spielt neben dem „Ob es geht?“ aber auch die Frage nach dem „Wie komfortabel geht es?“ eine nicht unerhebliche Rolle. Schließlich entstehen bei der Einführung von SGML in einem Unter-

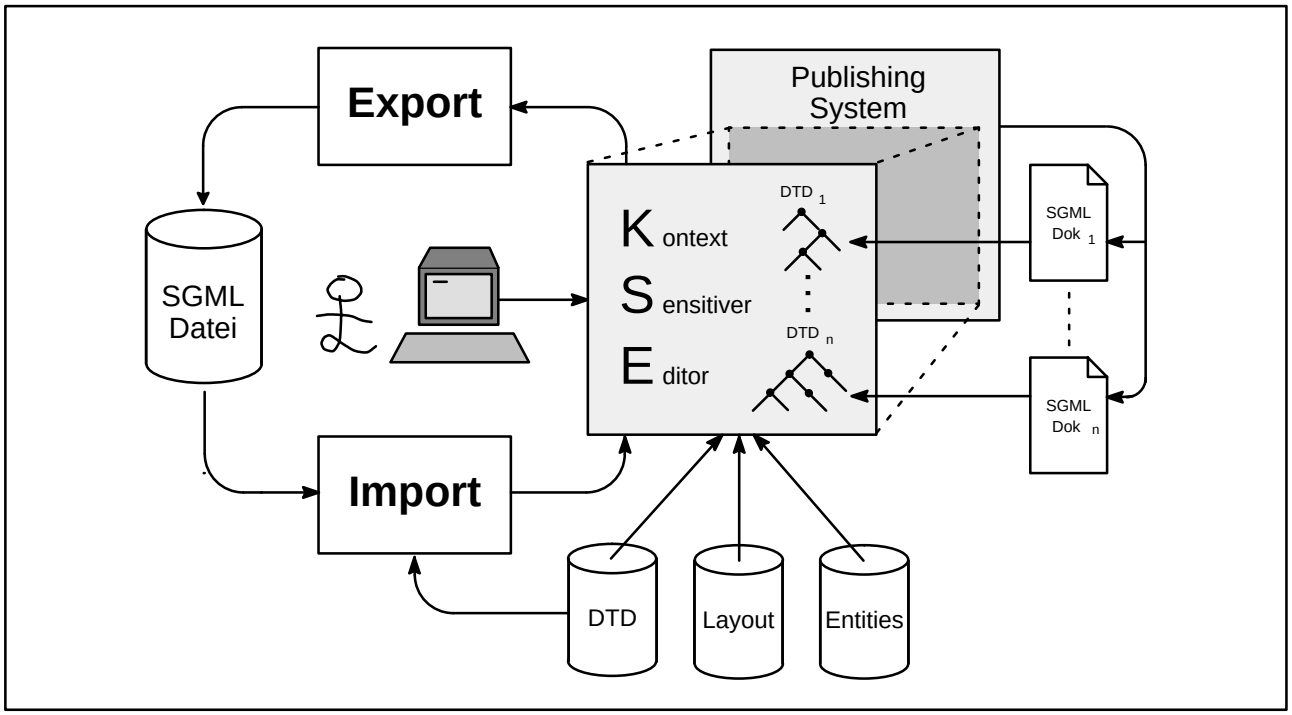


Abbildung 3–2
Kontext sensitiver WYSIWYG SGML Editor

nehmen gerade bei der Erstellung der Anwendungen nicht unerhebliche Kosten, die sich bei einem leicht zu bedienenden und komfortablen System sicherlich gering halten lassen.

4 Aufbau eines SGML-Systems

Mit denen im Abschnitt 3 vorgestellten Werkzeugen *Parser*, *Editor* und *Formatierer* sind bereits die wichtigsten Software-Komponenten zur Bearbeitung von SGML-Dokumenten genannt. In der Abbildung 4–1 wird dies auf einem allgemeineren Niveau dargestellt.

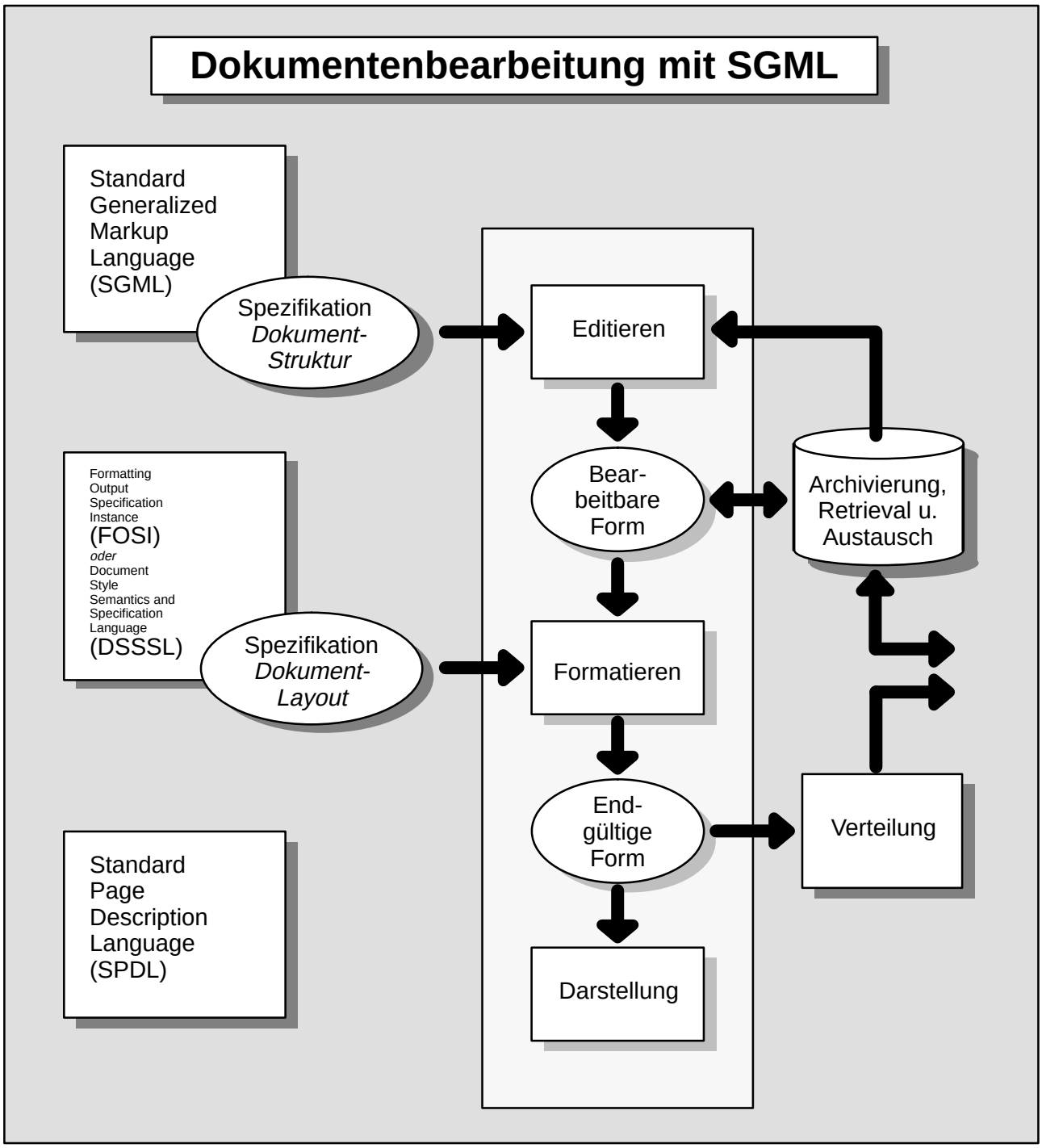


Abbildung 4–1
Dokumentbearbeitung mit SGML

Bei der Bearbeitung von SGML-Dokumenten gibt es insgesamt vier aktive Komponenten: *Editor*, *Formatierer*, *Datenbank* und *Anzeigesystem*. Handelt es sich bei dem Editor um ein WYSIWYG-System, dann sind Editor und Formatierer in diesem einen System zusammengefaßt.

- Der **Editor** liest die Spezifikation der Dokumentstruktur (SGML-DTD). Der Autor kann damit neue Dokumente erstellen oder Dokumente aus der Datenbank weiterbearbeiten. Gespeichert bzw. abgelegt werden die Dokumente als SGML-Instanzen, also in einer strukturorientierten, weiterbearbeitbaren Form. Die Dokumente werden an die Datenbank oder an den Formatierer übergeben.
- Der **Formatierer** liest die Spezifikation des Dokument-Layouts. Diese kann in einer standardisierten Layout-Beschreibungssprache, wie FOSI (Formatted Output Specification Instance, MIL-M-28001 [7]) oder DSSSL (Document Style Semantics and Specification Language, ISO DIS 10179 [8]), aber auch in einem systemspezifischen Format vorliegen.

  Wesentlich ist allerdings, daß der Formatierer eine Papierform und eine elektronische Form des Dokumentes generieren kann und das beide Formen zur Verteilung gelangen. Dabei ist die elektronische Form, wie auch die Papierform, ein vom Leser nicht mehr veränderbares Read-Only-Dokument. Die elektronische Form des Dokumentes kann z. B. auf einer Seitenbeschreibungssprache, wie PostScript [9] oder SPDL (Standard Page Description Language, ISO/IEC 10180 [10]) aufbauen, die mittels eines Anzeigesystems am Bildschirm dargestellt werden kann. Es ist auch denkbar, daß Formatierer und Anzeigesystem in ein Programm integriert sind: das SGML-Dokument wird dann, ohne ein explizites Zwischenformat zu generieren, aufbereitet am Bildschirm angezeigt (s. dazu auch [11]).
- In der **Datenbank**, die im einfachsten Fall auch das Dateisystem sein kann, werden die Dokumente in SGML abgelegt und verwaltet. Dort liegen sie zur Weiterbearbeitung, für Retrieval-Anfragen oder zum Austausch mit anderen Partnern bereit.
- Das **Anzeigesystem** zur Darstellung der formatierten Dokumente am Computer-Bildschirm erlaubt dem Anwender keine Veränderungen am Dokument. Wohl aber sind Such-Operationen und HyperText-Funktionalität denkbar. Der Benutzer kann eigene Annotationen zu Textstellen machen und neue, auch dokumentübergreifende Querverweise anlegen. Die HyperText-Daten werden dabei nicht in den Dokumenten, sondern als Zusatzinformationen gespeichert.

Unberücksichtigt bleibt der Aspekt des eigentlichen Austausches der SGML-Dokumente. Da ein solches Dokument aus mehreren Dateien (DTD, Layout-Beschreibung, Instanz, Graphik-Dateien) bestehen kann (s. Abbildung 4–2), müssen die Daten vor bzw. nach dem Austausch „zusammengepackt“ bzw. „ausgepackt“ werden (s. Abbildung 4–3). Zur Zeit stehen für den Austausch von SGML-Dokumenten die Standards SDIF (SGML Document Interchange Format, ISO 9069 [12]) und CALS-Tape (MIL-STD-1840 [13]) zur Verfügung.

5 DTD — Document Type Definition

Bevor Dokumente mit SGML bearbeitet werden können, müssen ihre logischen Strukturen in einer DTD festgelegt werden. Eine solche logische Struktur kann z. B. sein, daß ein Dokument immer mit einem Titel beginnt, der von einer nicht notwendigen Autorenangabe und diese wiederum von mehreren Kapiteln gefolgt wird. Außerdem besteht die Struktur eines Kapitels aus einer Überschrift, gefolgt von Absätzen, Aufzählungen oder Unterkapiteln.

Es ist also zwingend erforderlich, die Zieldokumente gründlich zu analysieren, um die Dokumentstrukturen zu erkennen und in die DTD umzusetzen. Werden hierbei Fehler gemacht, so gibt es bei der Dokumenterstellung mitunter schwerwiegende Probleme. Zwei Beispiele:

- Die DTD wurde zu „restriktiv“ erstellt: die Autoren haben nicht die Freiheiten, die sie bei der Dokumenterstellung brauchen.
- Die DTD wurde zu „lasch“ erstellt: Die Autoren nutzen alle Freiheiten aus und der gewünschte Effekt von „einheitlichen“ Dokumenten bleibt aus.

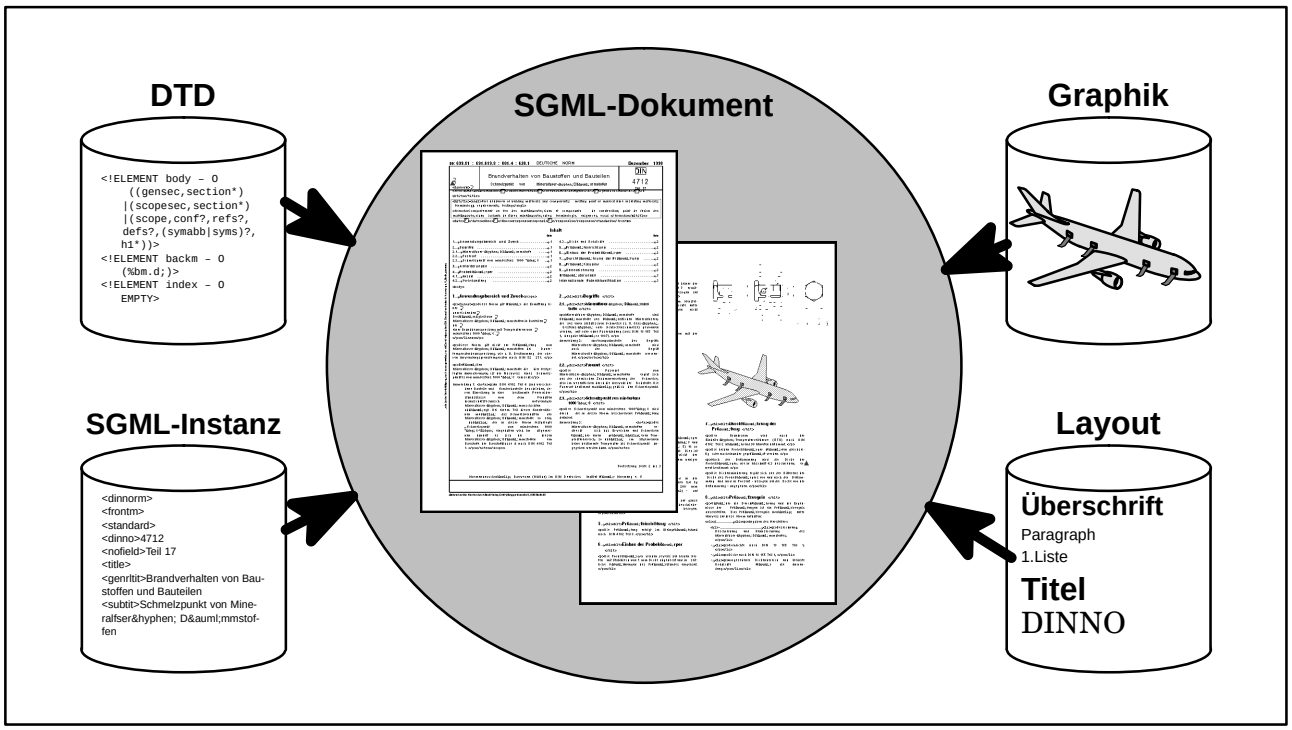


Abbildung 4–2
Teile eines SGML-Dokumentes

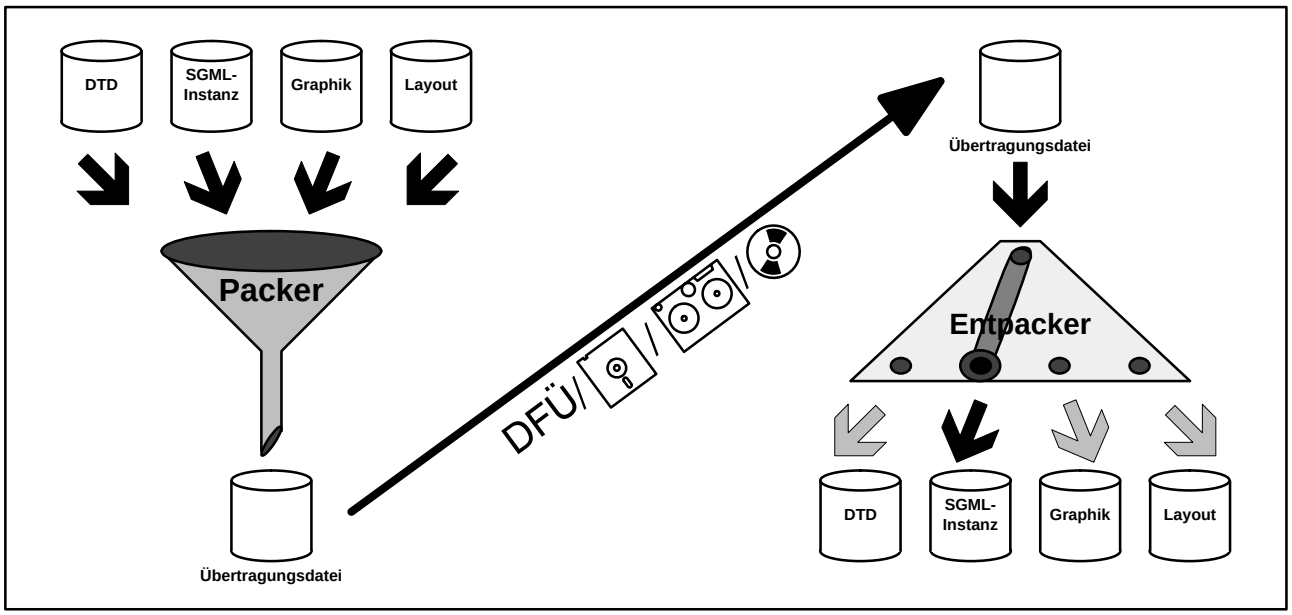


Abbildung 4–3
„Zusammenpacken“ (Packer) und „auspacken“ (Entpacker) der Teile eines SGML-Dokumentes

Diese beiden extremen Beispiele deuten an, daß die DTD-Entwicklung erhebliche Auswirkungen auf die Arbeitsweise der Autoren hat. Es ist deshalb von Vorteil, die Autoren bereits beim Festlegen der DTD hinzuzuziehen. Bei einer nachträglichen Änderung der DTD ist es einfacher, von einer „restriktiven“ zu einer „laschen“ DTD überzugehen als umgekehrt. Dies ist besonders dann bedeutend, wenn bereits sehr viele Dokumente nach der DTD erstellt wurden und sich Unzulänglichkeiten zeigten. Generell ist eine DTD-Entwicklung eine gemeinsame Aufgabe von SGML-Experten und Autoren.

Nachfolgend soll ein kurzer Überblick über die Möglichkeiten einer DTD gegeben werden. Genauere Angaben sind in der Literatur [1][14][15][16][17] zu finden.

Anmerkung: Einige Hinweise zur angegebenen SGML-Literatur: Die ISO-Norm [1] ist nur für Software-Entwickler interessant, die sehr tief in die Sytax und Semantik von SGML einsteigen müssen. Die Anhänge der Norm bringen die meiste Information. Das gleiche gilt für das SGML-Handbook [15], das im wesentlichen eine ausführlich kommentierte Version der ISO-Norm ist. Durch seinen sehr guten Index eignet es sich allerdings hervorragend als SGML-Nachschlagewerk. Die beiden Bücher von van Herwijnen [16] und Bryan [14] sind dagegen an einem Stück lesbar und eignen sich deshalb zum Einarbeiten in SGML für den Autor, den DTD- oder Software-Entwickler. Das Buch von Bryan geht dabei etwas mehr in die Tiefe, ist aber durch seine gewöhnungsbedürftige Kapitelreihenfolge beim ersten Lesen etwas verwirrend. Die SGML-Einführung von van Herwijnen [17] ist ein Online-Tutorial für PCs mit MS-Windows 3.x. Bis jetzt gibt es keine deutschsprachige Literatur für SGML.

Eine DTD besteht im wesentlichen aus sechs Teilen[4]:

1) **Angabe des zu definierenden Dokumenttyps** (01, 28): Der Name des *DOCTYPES* (hier „document“) dient zur Identifizierung der DTD und der Zuordnung Instanz↔DTD.

2) **Angabe von Sonderzeichen** (03–04): Sonderzeichen, die über die Zeichen des 7-Bit-ASCII-Zeichensatzes hinausgehen[5] sind in sog. Sonderzeichensätzen (hier „Added latin 1“[6]) in ISO 8879 definiert worden. Diese Sonderzeichensätze enthalten aber nur die symbolischen Namen der Zeichen (z.B. „auml“ für ein „ä“), nicht aber die zugehörenden Steuerzeichen für die Formatierer. In den Instanzen werden die Sonderzeichen über ihren symbolischen Namen (z.B. „ä“) angesprochen. Ein derartiger symbolischer Name hat den SGML-Fachausdruck *General Entity* (engl. für allgemeine Einheit). Sie werden in der DTD definiert und in den Instanzen verwendet.

3) **Definition von Parameter Entities** (06–07): Parameter Entities (engl. für Parametereinheit) können als Makros verstanden werden. Sie dienen dazu, das Schreiben der DTD zu vereinfachen. Sie sind textuelle Platzhalter (hier „outline“ und „text“) und machen die DTD übersichtlich und leichter erweiterbar. Parameter Entities wirken sich entgegen den General Entities nur in der DTD aus. Sie werden in der DTD definiert und auch in der DTD verwendet (z.B. „%text;“).

4) **Definition der Elemente** (10–20): Die inhaltlich relevanten Strukturteile des Dokumentes werden auf Elemente (hier z.B. „title“) abgebildet. Elemente haben weitere Unterelemente und/oder Text („#PCDATA“) als Inhalt. Das erste Element, welches identisch mit dem Dokumenttyp ist (s.o.), wird als *Basis Element* bezeichnet. Der definierte Inhalt eines Elementes wird *Content Model* (engl. für Inhaltsmodell) genannt.

   Im Inhaltsmodell legen SGML-Operatoren die logischen Strukturzusammenhänge der Unterelemente fest[7]: die Klammern regeln die Gruppierung von Unterstrukturen bzgl. den Operatoren[8], das Komma (,) legt die Reihenfolge fest, der senkrechte Strich (|) stellt eine 1-aus-*n*-Auswahl dar, das Und-Zeichen (&) gibt die notwendigen Elemente ohne bestimmte Reihenfolge fest, das Fragezeichen (?) markiert ein optionales Element, der Stern (*) markiert ein optionales, wiederholbares Element und das Plus-Zeichen (+) markiert ein erforderliches, wiederholbares Element. Die letzten drei Operatoren (?, *, +) können auch auf Gruppen angewendet werden. Fehlen an einem Element oder einer Gruppe diese Operatoren, so ist das Element oder die Gruppe erforderlich, darf aber nicht wiederholt werden.

   Neben dem Inhaltsmodell können Elemente auch *Inclusions* (engl. für Einschließungen, Inklusionen, s. Zeile 10 „+(fn)“) und *Exclusions* (engl. für Ausschließungen, Exklusionen, s. Zeile 19 „–(fn)“) besitzen. Eine Inklusion erlaubt die Verwendung der angegebenen Elemente (hier „fn“) an jeder beliebigen Stelle im Inhaltsmodell der Elementdefinition und in den Inhaltsmodellen von allen Unterelementen. Die Exklusion

---

4) Die angegebenen Zahlen sind die Zeilennummern des nachfolgenden Beispiels.

5) Der 7-Bit-ASCII-Zeichensatz [18] ist der Basiszeichensatz von SGML.

6) Dieser Sonderzeichensatz entspricht dem ISO 8859 Zeichensatz und enthält die deutschen, französischen, skandinavischen, etc. Sonderzeichen.

7) Die Operatoren ähneln den Operatoren von regulären Ausdrücken wie sie in formalen Sprachen bekannt sind [19].

8) SGML kennt keine Operatorpräferenzen.

hebt dies wieder auf, um z. B. rekursive Strukturen verhindern zu können. Zudem kann eine Exklusion die Verwendung von optionalen Elementen verbieten. Es ist möglich, mehreren Elementen das gleiche Inhaltsmodell zuzugeordnen (s. Zeile 17,18).

Ein Minus-Zeichen (–) bzw. der Buchstabe O geben an, ob das Start-Tag bzw. das End-Tag in der Instanz erscheinen muß (Minus) oder nicht (Buchstabe O für „Omitted“). Dieser Mechanismus nennt sich *Minimization* (engl. für Auszeichnungsverminderung) und dient dazu, die Schreibarbeit zu verringen und die Instanz lesbarer zu machen[9].

5) **Definition von Attributen** (22–27): Einzelnen Elementen können mittels Attributen für die späteren Dokument-Instanzen bestimmte, vorgegebene Eigenschaften zugeordnet werden. Diese Eigenschaften können unterschiedlichster Art sein: Layoutauswirkungen (hier der Präfix bei einer Aufzählung, s. Zeile 26), Querverweise (s. Zeilen 23, 25, 27) oder beliebige andere Daten wie z. B. Datenbank-Informationen.

   Eine Attribut-Definition besteht aus dem Elementnamen, dem Attributnamen, dessen Wertebereich und einem Default-Wert. Gibt es keinen Default-Wert, so kann das Attribut optional sein („#IMPLIED“) oder es muß in der Instanz mit einem Wert belegt werden („#REQUIRED“).

6) **Kommentare** (02, 05, 08–09, 21–22): Kommentare dienen hauptsächlich dazu, die DTD verständlicher zu machen und zu gliedern.

5.1 Eine Beispiel-DTD

```
01 <!DOCTYPE document [
02 <!-- == Entity für Sonderzeichen ================================== -->
03 <!ENTITY % ISOlat1 PUBLIC "ISO 8879-1986//ENTITIES Added latin 1//EN" >
04 %ISOlat1;
05 <!-- == Parameter-Entity-Definitionen ============================== -->
06 <!ENTITY % outline "para | list" >
07 <!ENTITY % text    "(#PCDATA | xref)" >
08 <!-- == Element-Definitionen ======================================== -->
09 <!--      Name     Min  Inhaltsmodell             In-/Exklusionen     -->
10 <!ELEMENT document - - (title, author?, chapter+) +(fn) >
11 <!ELEMENT title    O O (#PCDATA) >
12 <!ELEMENT author   - O (#PCDATA) >
13 <!ELEMENT chapter  - O (heading, (((%outline;)+, subchap*) | subchap+)) >
14 <!ELEMENT subchap  - O (heading, (%outline;)+) >
15 <!ELEMENT heading  O O (#PCDATA) >
16 <!ELEMENT list     - - (item, item+) >
17 <!ELEMENT (para | item)
18                    - O (%text;)+ >
19 <!ELEMENT fn       - - (%text;)+                  -(fn) >
20 <!ELEMENT xref     - O EMPTY >
21 <!-- == Attribut-Definitionen ====================================== -->
22 <!--      Elem-Name Attr-Name Type                    Default        -->
23 <!ATTLIST chapter   id        ID                      #IMPLIED
24                    db-key    CDATA                   #IMPLIED >
25 <!ATTLIST subchap   id        ID                      #IMPLIED >
26 <!ATTLIST list      prefix    (bullet | dash | number) "bullet" >
27 <!ATTLIST xref      refid     IDREF                   #REQUIRED >
28 ]>
```

5.2 Graphische Darstellung der DTD

Die Inhaltsmodelle der Elementdefinitionen lassen sich auf zwei Arten graphisch Darstellen. Die Darstellung als Baum mit textuellen Operatorangaben zeigt gut die hierarchischen Beziehungen zwischen den Elementen

[9] Beide Argumente sind bei der Anwendung von komfortablen SGML-Editoren nicht mehr zeitgemäß, da die Tags automatisch vom SGML-Editor erzeugt werden.

bei kleinen DTDs auf, während die Darstellung mit Syntaxdiagrammen sehr prägnant die Reihenfolge der Elemente bei beliebig großen DTDs aufzeigt. Beide Repräsentationen sind (mitunter in leichten Variationen) in der Praxis gebräuchlich und sollen anhand der Beispiel-DTD vorgestellt werden.

Anmerkung: Beide Präsentationsformen haben Probleme mit der anschaulichen Darstellung von In- und Exklusionen.

5.2.1 Darstellung als Baum

Die DTD wird als ein Baum gezeichnet, dessen Wurzel das Basis Element darstellt. Der Baum wächst nach unten. Die Knoten sind die Elemente oder Parameter Entities. Die Blätter sind entweder die leeren Elemente (EMPTY) oder Texte (#PCDATA). Jede Kante wird mit einer dreibuchstabigen, englischen Abkürzung des jeweiligen SGML-Operators ausgezeichnet.[10)] Sind Unterstrukturen bereits an anderer Stelle im Baum dargestellt, so wird der Knoten grau hinterlegt (siehe Abbildung 5–1).

5.2.2 Darstellung als Syntaxdiagramm

Jede Elementdefinition bzw. dessen Inhaltsmodell wird als ein Syntaxdiagramm gezeichnet. Dabei werden die SGML-Operatoren durch Pfeile, die Elemente bzw. Parameter Entities durch runde Boxen und die leeren Elemente (EMPTY) bzw. Texte (#PCDATA) durch eckige Boxen dargestellt. Zu jeder runden Box gibt es ein weiteres Syntaxdiagramm, das die Unterstruktur beschreibt. Die Syntaxdiagramme werden von links nach rechts gelesen, indem man einfach den Pfeilen folgt (siehe Abbildung 5–2).

5.3 Eine Beispiel-Instanz

Zu Beginn einer Instanz wird mit der *DOCTYPE*-Angabe (01–03) bekanntgegeben, zu welcher DTD die Instanz gehört. Mit der *SYSTEM*-Angabe wird hier explizit der Dateiname der DTD angegeben. In Zeile 02 wird die DTD für diese Instanz um den General Entity „sgml“ erweitert. Dieser Entity ist nun für diese Instanz zugelassen und er steht für den angegebenen Text „Standard Generalized Markup Language“. Dieser erste Teil der Instanz, der eigentlich die DTD darstellt, wird auch *Prolog* (engl. für Vorspann) genannt.

Mit dem *Start-Tag* des Basis-Elementes „document“ (04) beginnt die eigentliche Instanz. Sie endet mit dem zugehörenden *End-Tag* (22). Zur Verringerung der Schreibarbeit wurden alle Tags, deren Angabe nicht unbedingt erforderlich ist, weggelassen (s. o. „omitted Tags“ bzw. „Minimization“). Dies sind die Start- und End-Tags von *title* und *heading* (05, 07, 12, 20) und alle End-Tags bis auf die von *list* und *document*. End-Tags müssen nicht den Elementnamen beinhalten, es reicht die Angabe von „</>“ (17); sie bezieht sich dann auf das letzte noch nicht abgeschlossene Start-Tag.

Attribute werden beim Start-Tag gesetzt, so wie dies bei *chapter* (06), *xref* (13) und *list* (15) zu sehen ist. Die zur Realisierung von Querverweisen eingesetzten Attributtypen *ID* und *IDREF* werden vom Parser überprüft; d. h. kein Wert der ID-Attribute darf doppelt vergeben werden und zu jedem IDREF-Attribut muß es ein ID-Attribut mit dem gleichen Wert geben (06, 13).

```
01 <!DOCTYPE document SYSTEM "dtd-file" [
02 <!ENTITY sgml "Standard Generalized Markup Language" >
03 ]>

04 <document>
05 SGML - &sgml;

06 <chapter id=einl db-key="DB-1-234">
07 Einleitung

08 <para>Um auch in SGML&hyphen;Dokumenten Sonderzeichen, wie z. B.
09 deutsche Umlaute zu verwenden, gibt es den Mechanismus Entity. Ein
10 &amp;Auml; steht f&uuml;r ein gro&szlig;es "Ae".

11 <chapter>
12 SGML&hyphen;Sprachkonstrukte
```

[10)] Die Abkürzungen und ihre Bedeutungen: REQ = required (erforderlich), OPT = optional, REP = repeatable (wiederholbar), SEQ = Sequence (Sequenz), CHO = Choice (Auswahl), AND = And (Und).

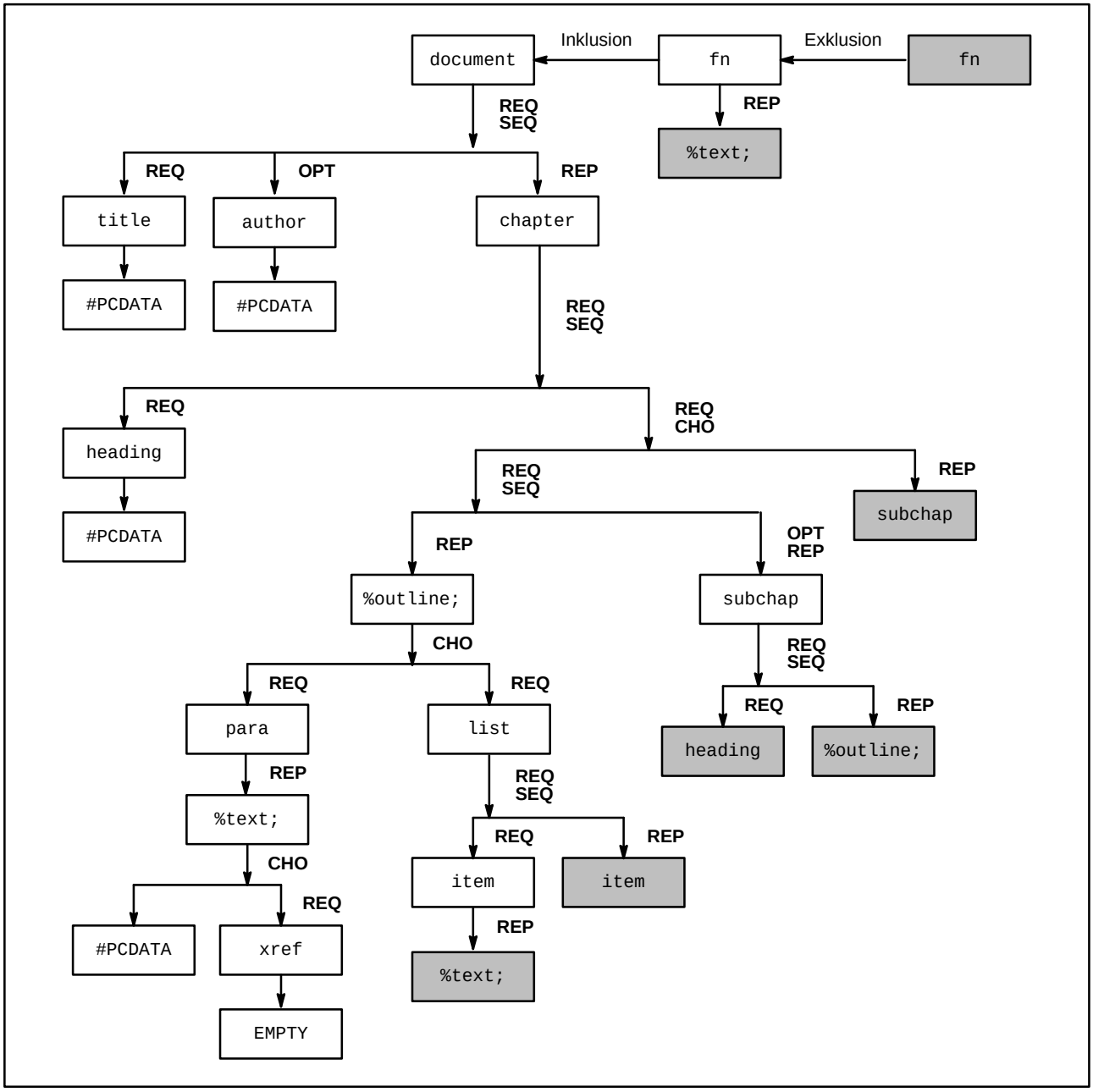
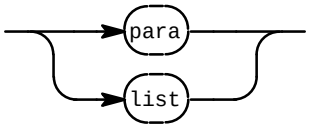


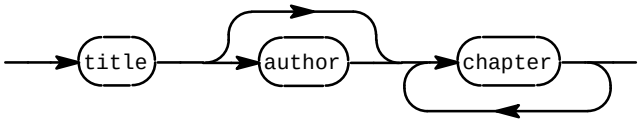
Abbildung 5–1
Graphische Darstellung der DTD als Baum

```
13 <para>Wie bereits in Abschnitt <xref refid=einl> erw&auml;hnt gibt es
14 den Sprachkonstrukt Entity. Nachfolgend einige weitere Konstrukte:
15 <list prefix=dash>
16 <item>SGML&hyphen;Declaration in der Reference Concrete Syntax
17 <item>Doctype in der DTD<fn>DTD = Document Type Definition</>
18 </list>
19 <subchap>
20 Sprachkonstrukte der DTD
21 <para>In der DTD werden Entities, Elemente und Attribute definiert.
22 </document>
```

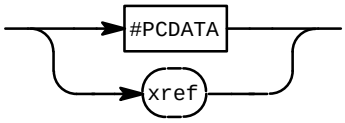
Entity: %outline



Entity: %text

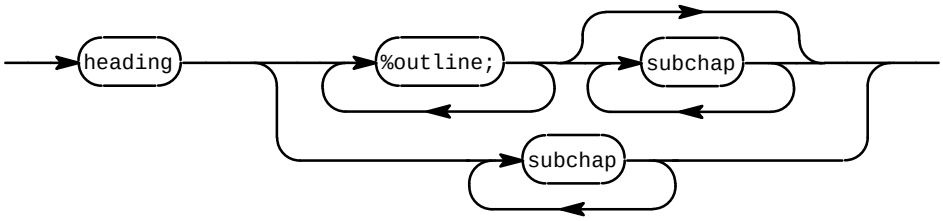


Element: document (Inkl.: ”fn”)

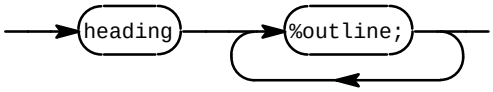


Elemente: title, author, heading

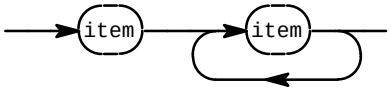
Element: chapter



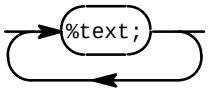
Element: subchap



Element: list



Elemente: para, item, fn (Exkl.: ”fn”)



Element: xref

Abbildung 5–2
Graphische Darstellung der DTD als Syntaxdiagramm

Sollte eine SGML-Instanz alle Tags (auch die, die omitted sind) enthalten, so wird sie *Vollauszeichnung* genannt. Beinhaltet sie auch noch alle Attributwerte einschließlich ihrer Default-Werte, so nennt man dies die *kanonische Form*. Diese stellt eine normalisierte Form der Instanz dar, die auch von einfacheren Software-Tools[11)] als einem SGML-Parser eingelesen und weiterverarbeitet werden kann.

5.4 Die Instanz als formatiertes Dokument

SGML – Standard Generalized Markup Language

1 Einleitung

Um auch in SGML-Dokumenten Sonderzeichen, wie z. B. deutsche Umlaute zu verwenden, gibt es den Mechanismus Entity. Ein Ä steht für ein großes ”Ae”.

2 SGML-Sprachkonstrukte

Wie bereits in Abschnitt 1 erwähnt gibt es den Sprachkonstrukt Entity. Nachfolgend einige weitere Konstrukte:
— SGML-Declaration in der Reference Concrete Syntax
— Doctype in der DTD[1)]

2.1 Sprachkonstrukte der DTD

In der DTD werden Entities, Elemente und Attribute definiert.

1) DTD = Document Type Definition

5.5 Graphische Darstellung der Instanz

Auch die Struktur einer Instanz läßt sich graphisch darstellen. Hierzu eignet sich am besten eine Darstellung als Baum. Da der Baum nicht sehr tief, dafür aber umso breiter sein wird, wächst er von links nach rechts. Die Wurzel ist wieder das Basis Element. Die Knoten sind die in der Instanz verwendeten Elemente. Die Blätter sind der Text bzw. die leeren Elemente. Die Kanten repräsentieren die Hierarchieebenen. Um die Hierarchieebenen noch stärker hervorzuheben wird ein schematisierter Baum verwendet. (siehe Abbildung 5–3).

6 Zusammenfassung der Vor- und Nachteile von SGML

Das SGML in Firmen und Institutionen angewendet wird, ist nicht nur das Resultat von äußeren Zwängen, wie Gesetze oder Kundenwünsche. Der Einsatz von SGML in einem Unternehmen ist von zahlreichen Vorteilen begleitet, die hier genannt werden sollen. Aber auch die Nachteile sollen nicht verschwiegen werden.

6.1 Vorteile

6.1.1 Allgemeine Vorteile

Etablierter, internationaler Standard

SGML stellt ein Dokumentformat dar, das firmenunabhängig in einem internationalen Standard normiert ist. Seit Ende der 80er Jahre wird SGML in immer mehr Applikationen angewendet. Es ist also kein *exotisches* Format, das nur eine Randerscheinung ist. Viele namhafte Software-Hersteller unterstützen SGML in ihren

11) Dies könnte z. B. ein mit den UNIX-Dienstprogrammen *lex* und *yacc* [20] generiertes Einleseprogramm sein.

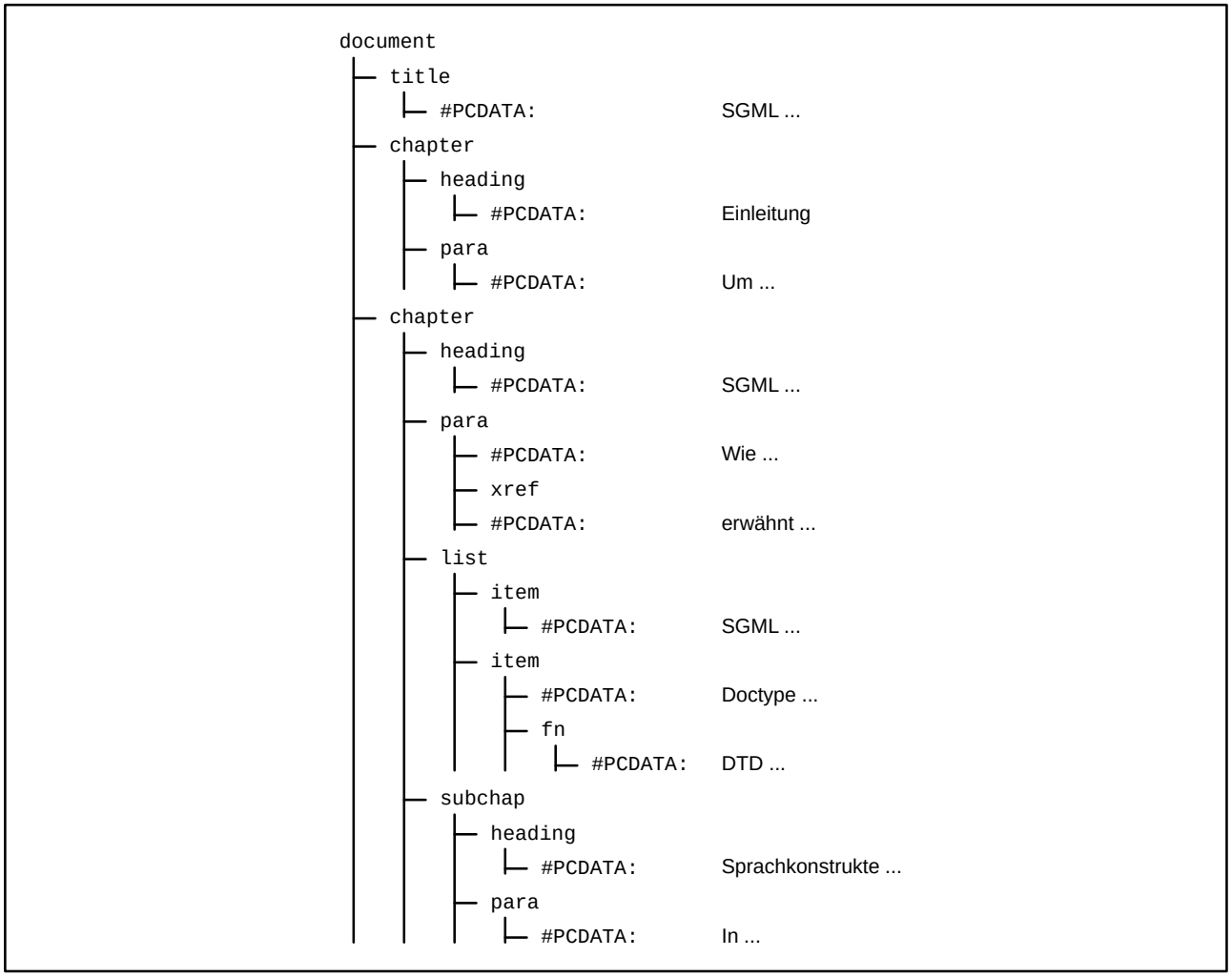


Abbildung 5–3
Graphische Darstellung der Instanz als schematisierter Baum

Produkten oder haben es zumindestens angekündigt und in einigen Anwendungsbereichen gehört SGML bereits zum Tagesgeschäft bei der Dokumenterstellung: Automobilbau, Luftfahrt, Normen, Rüstungsindustrie.[12)]

Unabhängig von Hard- und Software

Mit der Entscheidung für SGML legt sich der Anwender weder auf eine bestimmte Hardware noch auf eine bestimmte Software fest. SGML wird auf den verschiedenstens Hardware-Plattformen (vom PC bis zum Mainframe) unterstützt. Die Anzahl der Programme, die SGML verarbeiten können, nimmt ständig zu. Der Anwender hat also die Gewißheit, daß er nicht von einem Hersteller abhängig ist, sondern die für ihn am besten geeignete Lösung auswählen kann.

Langlebigkeit der SGML-Dokumente

Anders als bei einem Hersteller-spezifischem Speicherformat, bei dem niemand (unter Umständen nicht einmal der Hersteller selbst) wissen kann, wie lange es noch von einem Programm unterstützt werden wird, ist das

[12)] Automobilbau: die U.S.-amerikanische Umweltbehörde verlangt ab 1996 die Fahrzeugdaten in SGML nach der DTD *J2008*; Luftfahrt: die ATA, die Vereinigung der führenden Flugzeughersteller und Luftfahrtgesellschaften, stellt für die Dokumentation DTDs unter dem Konzept *ATA 100* bereit; Normen: für deutsche, europäische und internationale Normen gibt es die standardisierten DTDs *DIN V 33900*, *EN V 1285*, *ISO TR 9573-11*; Rüstungsindustrie: das U.S.-amerikanische Verteidigungsministerium erwartet jegliche Dokumentation in SGML nach der *CALS*-DTD (Computer-aided Acquisition and Logistics Support).

SGML-Format durch seine Standardisierung langfristig festgelegt[13]. Da SGML-Dokumente unabhängig von dem System sind mit dem sie erstellt wurden, kann ein solches Dokument auch noch nach Jahrzehnten von einem SGML-Werkzeug der *n*-ten Generation gelesen und bearbeitet werden. Daher eignet sich SGML ganz besonders für strategisch wichtige, langlebige Dokumente.

Es ist also nicht richtig, wie häufig irrtümlich angenommen, daß nur nicht revidierbare Rasterformate, wie z. B. CCITT FAX Gruppe 3 bzw. 4 für eine langfriste, elektronische Archivierung in Frage kommen.

Wohldefinierte Dokumentstrukturen

Mit dem Einsatz von SGML werden gleichzeitig die Dokumentstrukturen festgelegt und sind danach auch bzgl. ihrer Einhaltung überprüfbar. Dadurch kann erzwungen werden, daß wichtige Dokumentteile im Dokument vorhanden sein müssen[14] und daß die vorgegebene, logische Reihenfolge der Strukturteile eingehalten wird.

Enabling Standard

In der SGML ISO-Norm wird keine spezifische (Dokument-) Struktur festgelegt; die Norm liefert die Hilfsmittel zur Beschreibung von beliebigen Strukturen. Das bedeutet, daß SGML auf Dokumentarten mit verschiedensten, selbst komplexesten Strukturen (z. B. Tabellen, siehe [21]) anwendbar ist, da im Standard keine Voraussetzungen bzw. Einschränkungen bzgl. der Einsatzbereiche enthalten sind. Die Umsetzung der in der SGML-DTD formal beschreibbaren Strukturen ist lediglich von den Möglichkeiten der verfügbaren SGML-Systeme abhängig. Dadurch erlaubt SGML nicht nur die strukturierte Auszeichnung von Texten, vielmehr ist hier der Begriff Information anzuwenden. So ist z. B. mit SMDL (ISO/IEC CD 10743, [22]) eine SGML-Anwendung zur Beschreibung von Musik formuliert worden. Ebenso ist HyTime (ISO/IEC 10744, [23]) ein Formalismus zur Definition von multimedialen Hyperstrukturen, der auf SGML aufsetzt und die bestehenden Methoden einer DTD verwendet (s. u.).

Mehrfachnutzung

Die Nutzung von SGML-Dokumenten ist nicht nur auf die Bearbeitung in SGML-Editoren oder die Formatierung mit SGML-Systemen beschränkt. Von solchen Programmen wird zwar die herkömmliche Papier- und Bildschirmausgabe unterstützt, doch lassen sich in SGML erstellte Dokumente auch anders verwenden. Hervorzuheben sind hier der Austausch der Dokumente über Speichermedien (z. B. Diskette, Band, CD-ROM) oder über Netzwerke und die Archivierung der Dokumente in (Volltext-) Datenbanken zum späteren Retrieval nach Textteilen oder anderen speziellen Daten der Dokumente (s. u.).

Geringer Speicherplatzbedarf

Da in den SGML-Dateien nur die Texte mit ihren Strukturinformationen abgespeichert sind, nicht aber die für die Formatierung notwendigen Layout-Daten, ist der Speicherplatzbedarf eines SGML-Dokumentes um ein Vielfaches geringer als bei einem gleichen Dokument, das in dem Format eines Textverarbeitungssystems gespeichert ist. Dies Argument mag bei den heutigen Preisen für Festplatten nicht stark ins Gewicht fallen, wesentlich relevanter ist die Dateigröße aber bei der Preiskalkulation für die Übertragung der Daten über Netzwerke, bei denen häufig volumenorientiert abgerechnet wird.

6.1.2 Vorteile beim Publizieren mit SGML

Einheitliche Dokumente

Schreibt der Verlag allen seinen Autoren die Verwendung von einer Verlags-DTD vor, so ist damit sichergestellt, daß alle Dokumente einheitlich strukturiert sind. Damit ist eine einheitliche Weiterbearbeitung im Publikationsprozeß möglich.

---

[13] Zwar werden ISO-Normen alle fünf Jahre überarbeitet, bei SGML ist aber durch das Standardisierungsgremium sichergestellt, daß es aufwärtskompatibel bleibt.

[14] Dies kann z. B. ein Sicherheitshinweis in einem Wartungshandbuch sein, der bei der Verwendung von giftigen Stoffen während der Reparatur gesetzlich vorgeschrieben ist.

Flexibel anpaßbare Formatierung

Der Verlag hat die Möglichkeit, die Dokumente nach seinen Layout-Vorstellungen einheitlich zur formatieren. Dies kann sogar von einer auf die nächste Ausgabe geändert werden, ohne daß irgendein Dokument neu erfaßt werden muß oder die Autoren davon betroffen sind. Die Formatierung ist immer an die Anforderungen des Ausgabemediums[15)] anpaßbar.

Einfache Autorensysteme

Das Zuarbeiten der Autoren für einen oder mehrere Verlage vereinfacht sich mit dem Einsatz von SGML erheblich. Dem Autor reicht ein einfaches SGML-Autorensystem ohne komfortable Formatierung des Dokumentes, da die ohnehin erst endgültig beim Verlag durchgeführt wird. Der Verlag stellt ihm die DTD(s) zur Verfügung nach der die Dokumente erstellt werden sollen. Der Austausch der Dokumente erfolgt im SGML-Format, so daß keinerlei Informationen verloren gehen oder vom Verlag neu erfaßt werden müssen. Die Autoren haben mit SGML die Freiheit, mit einem Software-System gleichzeitig für mehrere Verlage zu arbeiten, so daß auch hier relativ geringe Kosten entstehen.

Übersetzung in Fremdsprachen

Die Übersetzung von in SGML vorliegenden Texten in eine oder mehrere Fremdsprachen ist denkbar einfach.[16)] Der Übersetzer (oder die Übersetzungs-Software) kann den Text elementweise hernehmen und übersetzen, ohne dabei die im Dokument vorhandene Struktur zu zerstören. Mit dem SGML-Feature *Marked Section*[17)] könnten sogar alle Übersetzungen und der Originaltext „umschaltbar“ in einem Dokument gehalten werden. Das übersetzte Dokument besitzt die gleiche Struktur wie das Ausgangsdokument und muß nur noch formatiert werden. Mit dem Austauschformat SGML ist auch die Schnittstelle zwischen Redaktion und Übersetzer eindeutig festgelegt.

6.1.3 Archivierung und Retrieval von Dokumenten

Volltext-Recherche

Vom theoretischen Ansatz her bietet SGML die besten Möglichkeiten zur Dokument-Recherche. Dabei könnte direkt auf den SGML-Dokumenten recherchiert werden. Notwendig sind dazu allerdings Datenbanken bzw. Schnittstellen zu Datenbanken, die den streng hierarchischen Datenaufbau der SGML-Dokumente optimal abbilden, um auch eine akzeptable Performanz zu erreichen. Mit einem solchen Werkzeug wären dann Anfrage der Art „Welche anderen Dokumente werden in diesem Dokument referiert?“, „In welchen Dokumenten wird das Dokument XYZ referiert?“ oder „In welchen Dokumenten vom Autor N.Iemand kommt der Begriff ABC im Titel vor?“ denkbar. Eine wichtige Anforderung an eine derartige Datenbank wäre die Offenheit bzgl. beliebigen DTDs, d.h. Dokumentenarten.

Datendurchgängigkeit

Die Mehrfachverwendung von Daten/Informationen während des Produkt-Lebenszyklus ist eine der Grundideen des *Engineering Data Management* (EDM). Vom ersten Entwurf des Produktes über die Fertigung bis hin zur Wartung und dem Recycling begleitet die Produktdokumentation das Produkt. Wesentlich hierbei ist, daß von Anfang an auf die gleichen Daten/Informationen zugegriffen werden kann, um Doppelerfassung und

---

15) Print-Medien: Tageszeitung, Magazin, Buch, etc.; elektr. Medien: Bildschirme mit unterschiedlicher Auflösung, verschiedene Fenster-Systeme; Ausgaben für Sehbehinderte: Text in Großschrift, Braille-Schrift

16) Bei der Übersetzung von SGML-Dokumenten muß berücksichtigt werden, ob bei der Formatierung für bestimmte Start-/End-Tags feste Texte generiert werden. Diese müssen natürlich auch übersetzt werden. D.h. daß entweder der Formatierer bzw. das SGML-System mehrsprachig gehalten werden muß oder daß *keinerlei* feste Texte generiert werden dürfen; das wiederum bedeutet, daß jeder, evtl. noch so redundante Text in der DTD spezifiziert sein und in allen Instanzen vorkommen muß.

17) Engl. für markierter Abschnitt. Eine *Marked Section* ist ein Textbereich, der für das Gesamtdokument aktiv oder inaktiv geschaltet werden kann. Damit ist eine Art Versionskontrolle in SGML-Dokumenten realisierbar.

die damit verbundene Fehleranfälligkeit zu vermeiden.[18] SGML unterstützt diesen Ansatz, da selbst Dokumentteile strukturiert vorliegen und somit sehr einfach zu neuen Dokumenten zusammengesetzt oder in andere Dokumente integriert werden können. SGML kann hier der Forderung des Produkthaftungsgesetzes entgegenkommen, indem unter anderem gefordert wird, daß Produkt und Dokumentation immer übereinstimmen müssen.

6.1.4 Versionenverwaltung durch ein Redaktionssystem

Verteilte Dokumenterstellung

An der Dokumenterstellung in größerem Rahmen sind mehrere Personen (Redakteure, Autoren) beteiligt. Wichtig ist hierbei das reibungslose Zusammenspiel bei der Bearbeitung der einzelnen Dokumentteile. Dies kann durch Festlegung der Zuständigkeiten gelöst sein, was aber keine rechnergestützte Überwachung zuläßt. Werden alle Dokumente bzw. Dokumentteile mit SGML ausgezeichnet in einer Datenbank abgelegt, so können die Autoren ein Dokumentteil aus der Datenbank *aus-checken*, es mit ihrem lokalen SGML-Editor bearbeiten und wieder in die Datenbank *ein-checken*. Ein ausgecheckter Dokumentteil ist gegen die Bearbeitbarkeit durch andere geschützt, er kann nur gelesen werden bis er wieder eingecheckt ist. Ein solches System kann zentral darüber informieren, in welchem Status sich die Dokumentation befindet und kann gleichzeitig Recherchemöglichkeiten anbieten. Es ist natürlich nicht auf SGML angewiesen, was allerdings die eindeutige Abgrenzung der Bearbeitungseinheiten (s.u.) und die Flexibilität bei der Auswahl der Editoren für die Autoren erleichtert und weitreichende Recherchen unterstützt.

Abgrenzung von Bearbeitungseinheiten

In der DTD können die Bearbeitungseinheiten der Autoren bzgl. Umfang und Inhalt formal genau festgelegt werden. Dadurch ist sichergestellt, daß auch die Gesamtdokumentation am Ende bei der Publikation korrekt und vollständig ist.

6.1.5 Austausch von Dokumenten

Verteilung über Netzwerk

Aufgrund des geringen Platzbedarfes der SGML-Dateien wären die Kosten für eine Übertragung der bei der Recherche gefundenen Dokumente über ein WAN (Wide Area Network: engl. für überregionales Netzwerk) nicht allzu hoch. Der Benutzer würde die SGML-Dateien auf seinen lokalen Rechner kopiert bekommen und könnte sie sich dort formatiert anzeigen oder ausdrucken lassen.[19]

6.1.6 Multimedia und HyperMedia

Einbinden externer Formate

Die Kombination der Mechanismen *Notation* und *External Entity* erlauben die Integration von beliebigen externen Formaten in ein Dokument. Damit läßt sich sehr einfach ein multimediales Dokument mit SGML beschreiben. Die externen Formate (z.B. Graphik, Animationen, Audio, Video) sind identifizierbar und können entsprechend bearbeitet werden. Notwendig sind lediglich die Werkzeuge zur Anzeige oder Bearbeitung der nicht-SGML Daten.

Aufbau von Hyper-Strukturen

Verbindungen (Links) zwischen den einzelnen Teilen (Elementen) *eines* SGML-Dokumentes sind durch den Standard-Mechanismus ID-IDREF abgedeckt. Hyper-Strukturen, die *mehrere* Dokumente miteinander verbinden, lassen sich durch Konventionen bzgl. der Auswertung von Attributwerten der SGML-Elemente generieren. Eine solche auf SGML-Attributen aufbauende Konvention wurde im internationalen Standard HyTime (Hypermedia/Time-based Structuring Language, ISO 10744 [23]) festgeschrieben.

---

[18] Z.B. wäre es denkbar, daß in der ersten Spezifikation festgelegte Grenzwerte auch noch in einem Wartungshandbuch oder beim Recycling-Prozeß auftauchen.

[19] Der Formatierungsprozeß würde lokal beim Benutzer ablaufen.

6.2 Nachteile

Obwohl SGML inzwischen in vielen Anwendungsbereichen etabliert ist, ist sein Einsatz auch mit einigen Problemen behaftet, die hier genannt werden sollen. Es ist aber anzunehmen, daß viele dieser Punkte in den nächsten Jahren durch zusätzliche Standards und verbesserte Software-Produkte nichtig geworden sind.

6.2.1 Einsatz von SGML

Kein Plug and Play möglich

Bei der Einführung von SGML reicht es nicht aus, einen SGML-Editor zu erwerben, um sofort mit der Dokumenterstellung zu beginnen. Zuerst müssen DTD und Layout festgelegt und in den Editor (oder das SGML-System) integriert werden.

Man braucht eine DTD

Die Bearbeitung von Dokumenten mit SGML macht es zwingend erforderlich, je eine DTD für jede zu bearbeitende Dokumentart zu formulieren. Ohne eine DTD kann SGML nicht eingesetzt werden.

Analyse der Dokumentstrukturen notwendig

Nur bei sehr einfachen Dokumentarten (wie Brief oder Memo) kann eine DTD direkt niedergeschrieben werden. Bei allen anderen, komplexeren Dokumentarten ist zuvor eine sorgfältige Dokumentanalyse notwendig. Erst das Ergebnis der Analyse wird in eine DTD umgesetzt.

Zuhilfenahme von SGML-Experten erforderlich

Entgegen anderen, *normalen* Textverarbeitungs- bzw. Publishing-Systemen reicht beim Einsatz von SGML eine einfache Nutzerschulung zur Bedienung der Software nicht aus. Meistens müssen (zumindest in der Anfangsphase) SGML-Experten hinzugezogen werden, die bei der Dokumentanalyse, der Formulierung der DTD, der Festlegung des Layouts und der *Programmierung* des SGML-Systems Unterstützung leisten.

6.2.2 Layout und Formatierung

Layoutbeschreibung ist notwendig

Nach der Formulierung der DTD kann zwar mit der Eingabe von strukturierten Dokumenten begonnen werden; was fehlt, ist aber noch die Information, wie das Dokument zu formatieren ist. Da in der DTD nur die Dokumentstruktur formal definiert wurde nicht aber das Layout, muß für jede Dokumentart mindestens[20)] eine Layoutbeschreibung erstellt werden.

Allgemeingültige Layoutbeschreibungssprache fehlt

Zur allgemeinen Formulierung des Layouts fehlen zur Zeit standardisierte, allgemeingültige Formalismen. Existierende Layoutbeschreibungssprachen sind entweder nicht mächtig genug oder befinden sich noch im Standardisierungsprozeß. Daher ist man zur Zeit gezwungen, die Layoutbeschreibung in Form von Beispielen informell in Prosa niederzuschreiben, anstatt eine exakte Definition in einer formalen Notation abzulegen.

Hoher Integrationsaufwand in SGML-System

Dadurch daß eine allgemeingültige Layoutbeschreibungssprache fehlt, ist der Aufwand zur Integration des (in Prosa beschriebene Layouts) in ein SGML-System relativ hoch. Gäbe es einen solchen Formalismus, müßte das Layout nur einmal in ihm definiert werden und könnte von jedem SGML-System eingelesen werden. So muß der Integrationsaufwand für jedes System erneut durchgeführt werden, was unnötig hohe Kosten verursacht.

---

20) Mehrere Layoutbeschreibungen sind notwendig, wenn eine Dokumentart für die verschiedenen Anforderungen unterschiedlich formatiert werden soll.

6.2.3 Austausch von SGML-Dokumenten

Verschiedene DTDs für verschiedene Anwendungen

Obwohl SGML als **das** Austauschformat gilt, hat es auch in diesem Bereich seine Probleme. Die beginnen nämlich dann, wenn Dokumentteile aus verscheidenen Dokumentarten ausgetauscht werden sollen. Sind die DTDs nicht kompatibel, d.h. für die Dokumentteile wurden ungleiche Strukturen in den DTDs definiert, müssen Konvertierungen durchgeführt werden. Diese sind zwar immer möglich, doch müssen dazu spezielle Software-Werkzeuge angeschafft und programmiert werden. Man sollte also bereits beim DTD-Design darauf achten, mit welchen anderen Dokumentarten man Teile austauschen möchte. Insbesondere bei den Bereichen Tabellen und mathematische Formeln sollte man auf existierende und bewährte (= verbreitete) DTD-Fragmente zurückgreifen.

Allgemeine Transportschale notwendig

Um ein SGML-Dokument mit einem Partner auszutauschen, reicht es nicht, die SGML-Datei zu verschicken. Zu der SGML-Datei gehören unter Umständen mehrere Graphik-Dateien, die die Bilder des Dokumentes beinhalten, oder Dateien mit anderen externen Formaten. Außerdem sollten sicherheitshalber die DTD und die verwendeten SGML-Sonderzeichensätze (Entity-Sets) mit verschickt werden, um zu gewährleisten, daß der Empfänger das Dokument auch mit seinem SGML-System einlesen kann. Theoretisch müßte man auch die formale Layoutbeschreibung mit versenden (sofern es diese gibt). Aus einer SGML-Datei sind nun plötzlich eine ganze Reihe von Dateien geworden, die alle in einer festen Beziehung zueinander stehen. Zum geregelten Austausch von SGML-Dokumenten ist eine *Transportschale* notwendig, die alle Dateien *verpackt*, so daß sie wieder eindeutig und ohne Informationsverlust vom Empfänger *entpackt* werden können. Es existieren bereits einige unterschiedliche Ansätze[21)] für eine solche Transportschale, allerdings hat sich bisher noch keiner durchgesetzt (siehe auch [24]).

Technische Realiserung fehlt

Für alle Lösungsansätze gilt zur Zeit, daß eine globale, technische Realisierung fehlt. Denkbar wäre hier insbesondere der Austausch über WAN (engl. für überregionales Netzwerk) oder elektronische Post (Elektronic Mail, EMail). Hierzu müßten die Pack-/Entpack-Programme die Schnittstelle zwischen Sender und Empfänger sein; wobei man auch davon ausgehen sollte, daß dies zwei Datenbanken sind, die ihren Datenbestand austauschen oder angleichen, d.h. das Versenden muß ohne menschliches Eingreifen vonstatten gehen.

6.2.4 Datenbanken

Spezielle Datenbankstrukturen notwendig

Will man nicht nur einzelne SGML-Dateien in einer Datenbank ablegen, sondern gezielt auch auf die Strukturen in diesen Dateien für z.B. Volltext-Recherche zugreifen, so müssen die Dateien zerlegt werden und in Datenbankstrukturen abgelegt werden, die eher hierarisch denn relational oder objektorientiert sind. Bislang fehlen hier solide Lösungen, die vor allem allgemeingültig, d.h. für beliebige DTDs anwendbar, und effizient sind. Einige Anforderungen zu diesem Punkt wurden in [25] formuliert und betreffen besonders die Anwendungen in kostengünstigen Hard-/Software-Umgebungen.

Effiziente Speicherung und Zugriff

Der effiziente Direktzugriff auf bestimmte Unterstrukturen des Dokumentes wird von SGML dadurch behindert, daß es keine Längenangaben zu den einzelnen Strukturteilen gibt, die eine schnelle Bestimmung des Zugriffpunktes erlauben würden. Deshalb muß eine SGML-Datei entweder zeichenweise durchsucht werden, bis die gesuchte Stelle gefunden ist, oder die Datei muß indiziert in der Datenbank abgelegt werden, was einen zusätzlichen (Verwaltungs-) Aufwand bedeutet.

---

21) CALS-Tape, MIL-M-28001 [13]; SDIF, ISO 9096 [12]

7 Literatur

[1] International Organization for Standardization: *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, ISO 8879:1986, ISO, 1986.

[2] Goldfarb, C.F.; Mosher, E.J.; Peterson, T.I.: *An Online System for Integrated Text Processing*, In: Proceedings of the American Society for Information Science, 7, S. 147–150, 1970.

[3] Knuth, D.E.: *The TeXbook*, Addison-Wesley, Massachusetts, 1984.

[4] Lamport, L.: *LaTeX: A Document Preparation System*, Addison-Wesley, Massachusetts, 1986.

[5] Schirmer, C.: *troff-Programmierung*, Carl Hanser Verlag, München, Wien, 1990, ISBN 3-446-15649-6.

[6] Deutsches Institut für Normung (Hrsg.): *Informationsverarbeitung — Textverarbeitung und -kommunikation — Genormte Verallgemeinerte Auszeichnungssprache (SGML)*, DIN EN 28879:1991, Beuth Verlag, 1991.

[7] U.S. Department of Defense: *Markup requirements and generic style specification for electronic printed output and exchange of text — Appendix B: Output specification*, MIL-M-28001A, U.S. Department of Defense, 1990.

[8] International Organization for Standardization: *Information technology — Text and office systems — Document Style Semantics and Specification Language (DSSSL)*, ISO/IEC DIS 10179:1991, ISO/IEC, 1991.

[9] Adobe Systems, Inc.: *PostScript Reference Manual*, Addison-Wesley, 1995, ISBN 0-201-10174-2.

[10] International Organization for Standardization: *Information technology — Text composition — Standard Page Description Language (SPDL)*, ISO/IEC 10180:1992, ISO/IEC, 1992.

[11] Hofmeyer, J.: *Realisierung von Werkzeugen zur Bearbeitung von Struktur, Layout und Inhalten technischer Dokumente*, Diplomarbeit, TH Darmstadt, Fachbereich 20 (Informatik), Fachgebiet Graphisch-Interaktive Systeme, 1993.

[12] International Organization for Standardization: *Information processing — SGML support facilities — SGML Document Interchange Format (SDIF)*, ISO 9069:1988, ISO, 1988.

[13] U.S. Department of Defense: *Automated interchange of technical information*, MIL-STD-1840B, U.S. Department of Defense, 1992.

[14] Bryan, M.: *SGML — An Author's Guide to the Standard Generalized Markup Language*, Addison-Wesley, 1988, ISBN 0-201-17535-5.

[15] Goldfarb, C.F.: *The SGML Handbook*, Oxford University Press, 1990, ISBN 0-19-853737-9.

[16] Herwijnen, E. van: *Practical SGML*, Kluwer Academic Publishers, Dordrecht/Boston/London, U.S.A., 1990, ISBN 0-7923-0635-X.

[17] Herwijnen, E. van: *SGML Tutorial*, Kluwer Academic Publishers, Dordrecht/Boston/London, U.S.A., 1992, 2 Disketten.

[18] International Organization for Standardization: *Information technology — ISO 7-bit coded character set for information interchange*, ISO/IEC 646:1991, ISO/IEC, 1991.

[19] Salomaa, A.K.: *Formale Sprachen*, Übersetzt aus dem Englischen von E.-W. Dieterich, Studienreihe Informatik, W. Brauner, G. Goos (Hrsg.), Springer-Verlag, 1978.

[20] Mason, T.; Brown, D.: *lex & yacc*, O'Reilly & Associates, Sebastopol (California), 1990, ISBN 0-937175-49-8.

[21] Rath, H.H.: *Tabellen in SGML*, ZGDV Bericht 75/93, Zentrum für Graphische Datenverarbeitung e. V., Darmstadt, 1993.

[22] International Organization for Standardization: *Information technology — Standard Music Description Language (SMDL)*, ISO/IEC CD 10743:1991, ISO, 1991.

[23] International Organization for Standardization: *Information technology — Hypermedia/Time-based Structuring Language (HyTime)*, ISO/IEC 10744:1992, ISO/IEC, 1992.

[24] Häfemeier, F.; Meißner, E.; Rath, H.H.: *Elektronische Speicherung und Handhabung technischer Dokumente — Arbeitspunkt 5: Transportschale*, Bundesministerium für Wirtschaft, Verbundprojekt Nr. 68504, 1992/93.

[25] Friederich, P.; Geibel, D.; Höfler, J.; Rath, H.H.; Wiedemann, R.; Wiedling, H.-P.: *Elektronische Speicherung und Handhabung technischer Dokumente — Arbeitspunkt 6: Machbarkeitsstudie & Pflichtenheft*, Bundesministerium für Wirtschaft, Verbundprojekt Nr. 68504, 1992/93.