Auszugsweise vorhandene Dokumentation der derzeitigen XML-Dokumente

© Karsten Karkola, Lars Goldhahn Fakultätsrechenzentrum Bauingenieurwesen TU Dresden

1. Anliegen

Bei der Arbeit im Rahmen des Projektes Portiko ist ein entscheidendes Kriterium die Gewährleistung einer nachhaltigen Dokumentenverwaltung. Dabei wurde Augenmerk auf eine freie, skalierbare Lösung gelegt. Im Rahmen mehrerer Projektworkshops wurde sich die XML-basierte Dokumentenerstellung geeinigt. Vorteile dabei sind:

- herstellerunabhängige Definition des Dokumentaufbaus
- weitverbreitete Unterstützung durch professionelle Werkzeuge und Open Source Implementierungen
- Weiterverarbeitung kann automatisch aber auch manuell erfolgen
- benutzerdefinierte Präsentation der Informationen (HTML, PDF, etc.)
- strikte Trennung von Inhalt und Layout

Beim Aufbau des Dokumentenaufbaus wurde auf eine Dokumenttypdefinition (Document Type Definition - DTD) zurückgegriffen. Die Beschreibung auf der Basis der XML Schema Empfehlung des W3C wurde nicht genutzt, da die Unterstützung dieser wesentlich komplizierteren Technologie noch nicht durchgängig gewährleistet wird.

Für den Aufbau wurden Ansätze aus der Arbeit mit LaTeX und DocBook berücksichtigt. Beim Aufbau der DTD wurde sich gegen eine direkte Verwendung von DocBook auf Grund des vergleichsweise komplizierten Aufbaus der DocBook DTDs ausgesprochen. Allerdings ist eine Präsentation der im contentobject Format vorliegenden Inhalte im DocBook Format leicht mit Hilfe eines passenden Stylesheets zu realisieren.

Im Kapitel "Beispiel" kann die gesamte Dokumentation mit der aktuellen DTD und dem letzten Stand des Stylesheets heruntergeladen werden (Stand: Thu Jan 16 14:37:00 2003).

1.1 Prinzipieller Aufbau der contentobject DTD

Die DTD gliedert sich in vier grundlegende Bereiche:

Dokumentenstruktur: beschreibt die Aufteilung des Dokumentes in Vorwort, Kapitel, Anhänge und Verzeichnisse (vgl. Kapitel "Dokumentenstruktur").

vcard Definition: beschreibt die Informationen zu Autoren, Details befinden sich unter vcard Definitionen

Inhaltselemente: der gesamte Inhalt wird innerhalb der body Elemente erfaßt und in den Kapiteln body Element und Tabellen beschrieben.

Inline Elemente: beschreiben die Elemente innerhalb eines Textabschnittes, wie z.B. innerhalb eines para oder title Elementes (siehe Inline Elemente)

Der Aufbau der DTD ist im Diagramm co_gesamt.pdf dargestellt.

1.2 History

1.2.1 History der DTD

1.2.1.1 Entstehung

März bis Mai 2002: Erster Entwurf einer contentobject DTD

Mai 2002 bis September 2002: Einsatz der contentobject DTD für Protokolle und Vorträge, Nutzung entsprechender Stylesheets für HTML und PDF

Oktober 2002: Reengineering der vorhandenen DTD

24. Oktober 2002: Vorstellung der fertiggestellten contentobject DTD im Dresdner Teilprojekt

4. November 2002: Fertigstellung und Veröffentlichung der contentobject DTD, v1.1a

1.2.1.2 Änderungen, Fehlerkorrekturen

Momentan ist keine Weiterentwicklung der DTD geplant. Es werden lediglich kleine Änderungen bzw. Fehlerkorrekturen durchgeführt. Die jeweils aktuellste Version kann mit diesem Beispieldokument unter contentobject_doku.tar.gz heruntergeladen werden.

27. November 2002: Version 1.1b veröffentlicht

Nachtragen des Elements description

Einfügen des Attributes contentobject/@numbering zur Steuerung der Numerierung der Strukturelemente (chapter, etc.)

Erweiterung von chapter, sect1, sect2, sect3, appendix um das Attribut @numbers

1.2.2 History Stylesheet

Das Entwicklerstylesheet wird parallel zur DTD entwickelt. Mit der Fertigstellung der contentobject DTD 1.1a am 4. November 2002 wurde ein entsprechendes Stylesheet veröffentlicht, das den grundlegenden Dokumentenaufbau unterstützt. Mit Fertigstellung der DTD v1.1b wird nur noch am Stylesheet und der vorliegenden Dokumentation gearbeitet.

27. November 2002: Unterstützung von Numerierungen

29. November 2002: Stylesheet überarbeitet:

Unterstützung aller list Typen

Aufspaltung des Stylesheets durch Heraustrennen aller block Elemente in die Datei inline.xsl

Korrektur der Numerierung (Differenzen zwischen XALAN und Oracle XDK)

6. Januar 2003: Veröffentlichung des neuen Stylesheets

2. Dokumentenstruktur

2.1. Überblick

Das contentobject Element gliedert sich in die folgenden Abschnitte

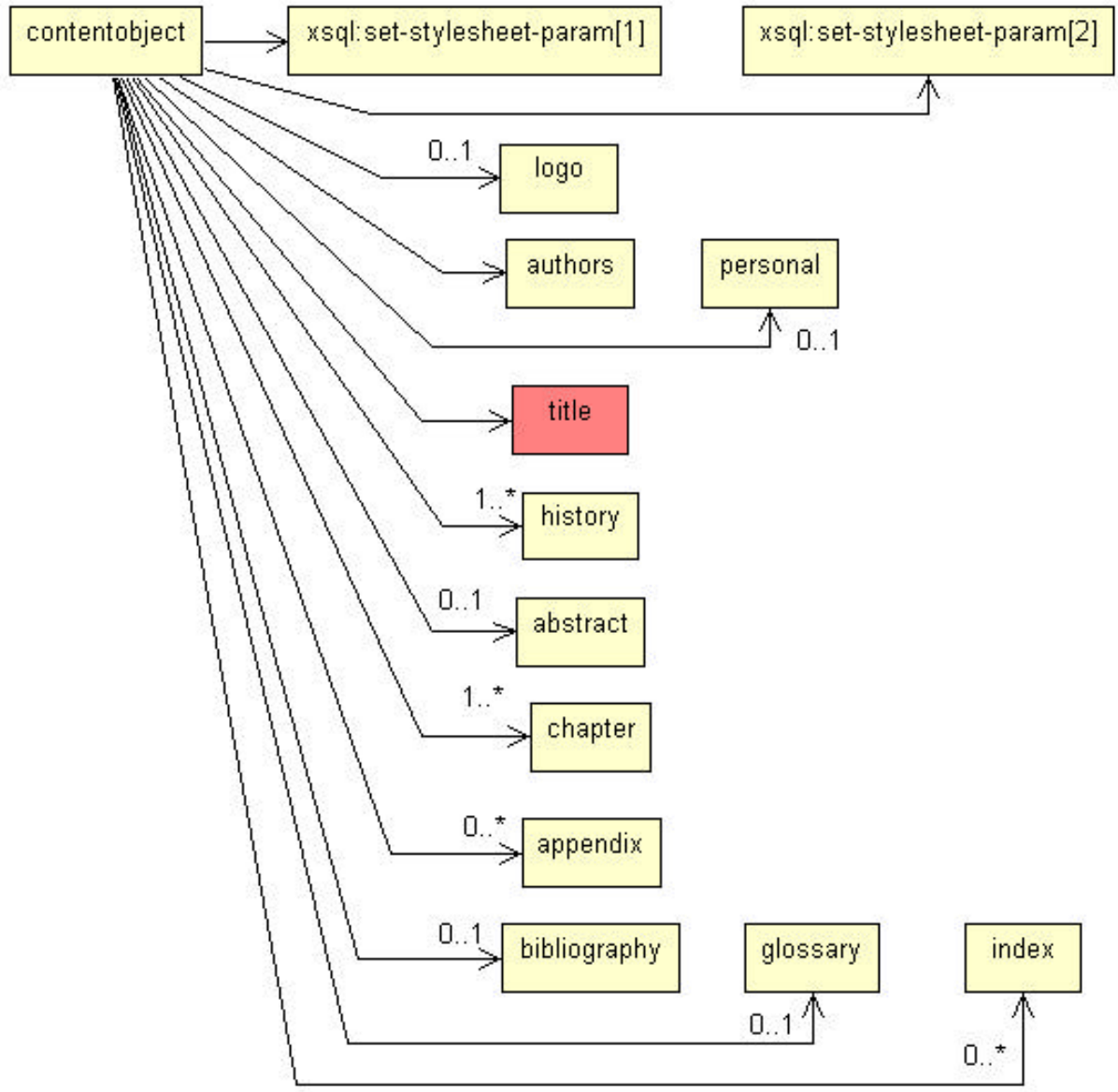

Abbildung 2.1 Gesamtstruktur eines contentobject Elements (PDF: structure_gesamt.pdf)

2.2. Aufbau der Kapitelelemente chapter

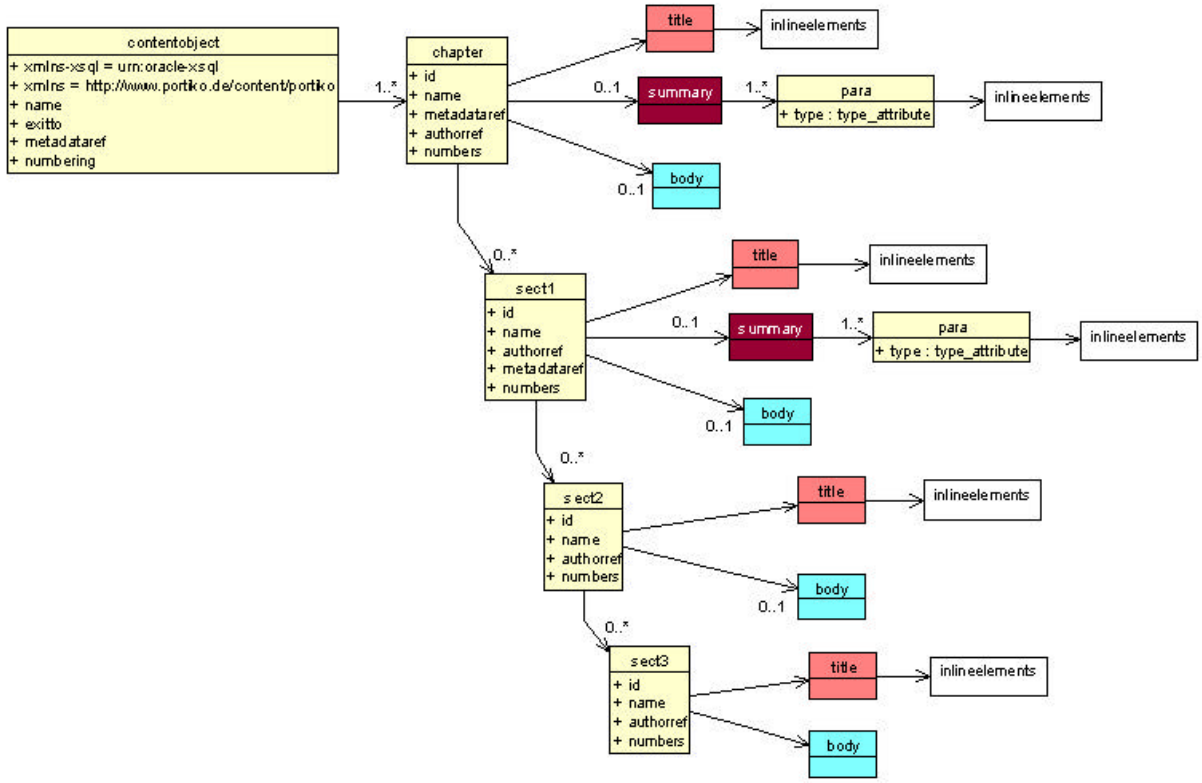

Abbildung 2.2 Kapitel eines contentobject Elements (PDF: chapters.pdf)

3. vcard Definition

3.1 Überblick zur vcard Definition

4. body-Element

Das body Element enthält den eigentlichen Inhalt des Dokuments und ist in die folgenden Elemente gegliedert:

para: ein Absatz, der Text und sogennante inline Elemente enthalten kann

figure: mit Hilfe dieses Elementes können Bilder eingefügt werden

listing: enthält z.B. Programmtexte

formula: zur Darstellung von Formeln in Form von Grafiken oder MathML

sound: referenziert Sounddateien

video: referenziert Videodateien

list: mit diesem Element können Aufzählungslisten realisisert werden

table: eine Umgebung zur tabellarischen Darstellung von Informationen

application: wird für Inhalte vewendet, die zu keinem der oben genannten Inhaltselemente passen, einige spezielle Applikation werden bei Angabe des MIME

Types gesondert behandelt, bekannte MIME Typen sind im Abschnitt application aufgeführt

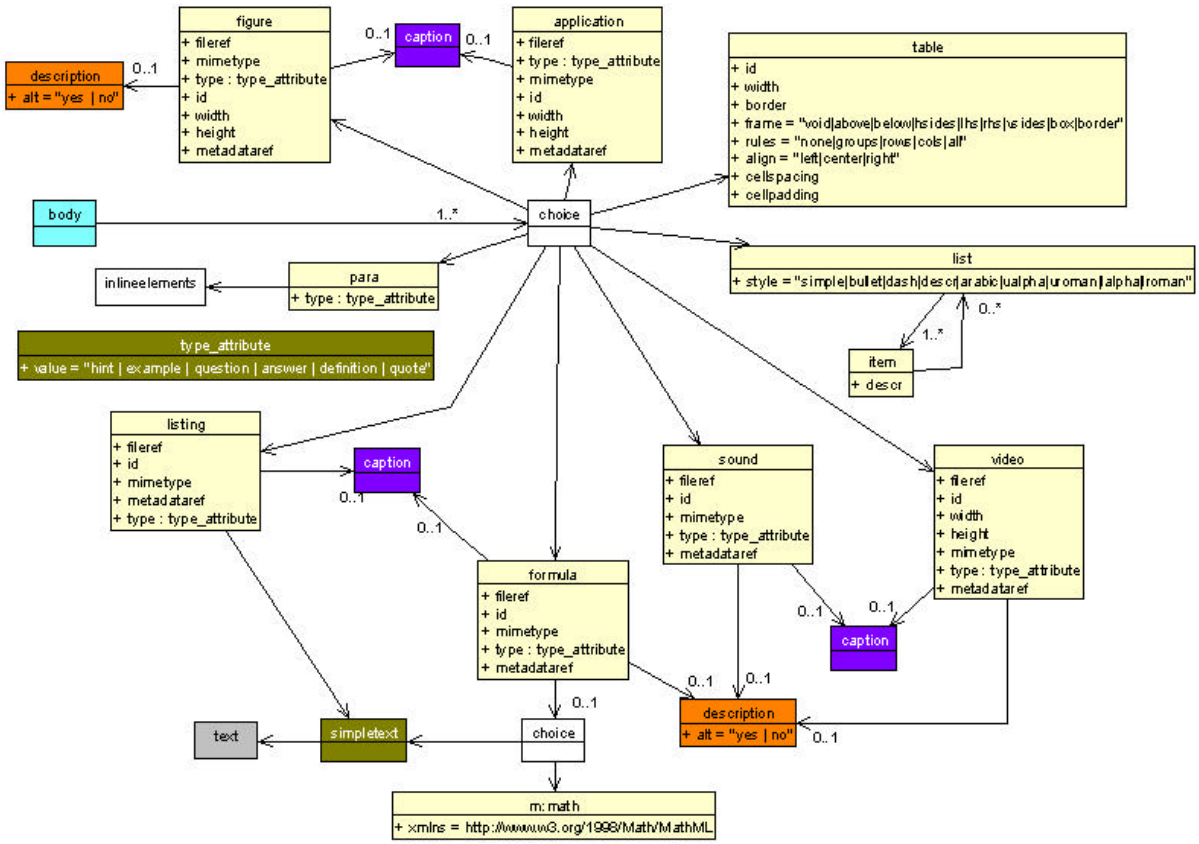

Abbildung 4.1 Aufbau des body Elements

4.2 Aufzählungslisten

4.2.1 Einfache Listen

Einfache Listen stellen die einzelnen Listenelemente ohne Listensymbol eingerückt dar:

Listenelement 1

Listenelement 2

Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="simple">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.2 Listen mit Punkt

Listen mit Punkt stellen die einzelnen Listenelemente mit einem Punkt (bullet) als Listensymbol eingerückt dar:

- Listenelement 1
- Listenelement 2
- Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="bullet">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.3 Listen mit Anstrich

Listen mit Anstrich stellen die einzelnen Listenelemente mit einem Anstrich (dash) als Listensymbol eingerückt dar:

- Listenelement 1
- Listenelement 2
- Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="dash">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.4 Beschreibungslisten

Diese Listen stellen die einzelnen Listenelemente mit einem hervorgehobenen Wort bzw. einer hervorgehobenen Wortgruppe dar:

Ele 1: Listenelement 1

Ele 2: Listenelement 2

Ele 3: Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="descr">
  <item descr="Ele 1:">Listenelement 1</item>
  <item descr="Ele 2:">Listenelement 2</item>
  <item descr="Ele 3:">Listenelement 3</item>
</list>
```

4.2.5 Geordnete Listen mit Numerierung

Listen mit Numerierung stellen die einzelnen Listenelemente geordnet mit einer Dezimalzahl als Listensymbol eingerückt dar:

1. Listenelement 1
2. Listenelement 2
3. Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="arabic">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.6 Geordnete Listen mit Großbuchstaben

Geordnete Listen mit Großbuchstaben stellen die einzelnen Listenelemente mit einem Großbuchstaben als Listensymbol eingerückt dar:

A. Listenelement 1
B. Listenelement 2
C. Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="ualpha">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.7 Geordnete Listen mit römischer Numerierung (groß)

Diese Listen stellen die einzelnen Listenelemente mit römischer Numerierung (groß) als Listensymbol eingerückt dar:

I. Listenelement 1

II. Listenelement 2

III. Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="uroman">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.8 Geordnete Listen mit Kleinbuchstaben

Geordnete Listen mit KLeinbuchstaben stellen die einzelnen Listenelemente mit einem Kleinbuchstaben als Listensymbol eingerückt dar:

a. Listenelement 1

b. Listenelement 2

c. Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="lalpha">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
</list>
```

4.2.9 Geordnete Listen mit römischer Numerierung (klein)

Diese Listen stellen die einzelnen Listenelemente mit römischer Numerierung (klein) als Listensymbol eingerückt dar:

i. Listenelement 1

ii. Listenelement 2

iii. Listenelement 3

Innerhalb des XML Dokumentes wird diese Liste mit den folgenden Elementen erstellt:

```
<list style="lroman">
  <item>Listenelement 1</item>
  <item>Listenelement 2</item>
  <item>Listenelement 3</item>
```

```
</list>
```

4.2.10 Geschachtelte Listen

Im Stylesheet und in der contentobject DTD werden ebenfalls geschachtelte Listen unterstützt. Die Art des Symbols muß aber vorgegeben werden, es erfolgt keine automatische Änderung des Sysmbols:

I. Listenelement 1
    - wichtiger Unterpunkt zu Listenelement 1
    - superwichtiger Unterpunkt zu Listenelement 1

II. Listenelement 2

III. Listenelement 3
    i. Listenelement 3 benötigt noch diesen Hinweis
    ii. Listenelement 3 benötigt noch jenen Hinweis

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<list style="uroman">
  <item>Listenelement 1
    <list>
      <item>wichtiger Unterpunkt zu Listenelement 1</item>
      <item>superwichtiger Unterpunkt zu Listenelement 1</item>
    </list>
  </item>
  <item>Listenelement 2</item>
  <item>Listenelement 3
    <list style="lroman">
      <item>Listenelement 3 benötigt noch diesen Hinweis</item>
      <item>Listenelement 3 benötigt noch jenen Hinweis</item>
    </list>
  </item>
</list>
```

4.3 Das Element <para> und seine Attribute

4.3.1 Das Attribut [hint]

Das Attribut hint dient zum Darstellen von Hinweisen und wird folgendermaßen dargestellt:

Hinweistext. Hinweistext. Hinweistext. Hinweistext. Hinweistext Hinweistext.

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<para type="hint">
       Hinweistext. Hinweistext. Hinweistext.
       Hinweistext. Hinweistext Hinweistext.
</para>
```

4.3.2 Das Attribut [example]

Das Attribut example dient zum Darstellen von Beispielen und wird folgendermaßen dargestellt:

Beispieltext. Beispieltext. Beispieltext. Beispieltext. Beispieltext Beispieltext.

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<para type="example">
       Beispieltext. Beispieltext. Beispieltext.
       Beispieltext. Beispieltext Beispieltext.
</para>
```

4.3.3 Das Attribut [question]

Das Attribut question dient zum Darstellen von Fragen und wird folgendermaßen dargestellt:

Fragetext. Fragetext. Fragetext. Fragetext. Fragetext Fragetext.

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<para type="question">
       Fragetext. Fragetext. Fragetext.
```

```
        Fragetext. Fragetext Fragetext.
</para>
```

4.3.4 Das Attribut [answer]

Das Attribut answer dient zum Darstellen von Antworten und wird folgendermaßen dargestellt:

Antworttext. Antworttext. Antworttext. Antworttext. Antworttext Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext. Antworttext.

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<para type="answer">
        Antworttext. Antworttext. Antworttext.
        Antworttext. Antworttext Antworttext.
```

4.3.5 Das Attribut [definition]

Das Attribut definition dient zum Darstellen von Definitionen und wird folgendermaßen dargestellt:

Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext. Definitionstext.

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<para type="definition">
        Definitionstext. Definitionstext. Definitionstext.
        Definitionstext. Definitionstext Definitionstext.
</para>
```

4.3.6 Das Attribut [quote]

Das Attribut quote dient zum Darstellen von Einrückungen und wird folgendermaßen dargestellt:

> Eingerückter Text.

Die Umsetzung erfolgt wie im folgenden Beispiel beschrieben:

```
<para type="quote">
       Eingerückter Text. Eingerückter Text.
       Eingerückter Text. Eingerückter Text.
       Eingerückter Text. Eingerückter Text.
</para>
```

5. Tabellen

5.1 Überblick

6. Inline-Elemente

6.1 Überblick zu den einzelnen inline Elementen

7. Beispiel

7.1 Überblick

Um den Umgang mit der contentobject DTD zu erleichtern, wird in diesem Abschnitt an Hand der vorliegenden Dokumentation das Erstellen eines umfangreicheren Textes demonstriert. Um das Beispiel einfach nachvollziehen zu können, kann die Dokumentation unter

- contentobject_doku.tar.gz

heruntergeladen. Dieses Archiv enthält neben den XML-Dokumenten die aktuelle DTD und die letzte Version des Stylesheets. Die in der Dokumentation referenzierten Ressourcen sind nicht in diesem Archiv enthalten, sie können gesondert heruntergeladen werden:

- Emacs: Referenzkarte emacs-refcard-a4.pdf
- Emacs: vorkonfigurierte Musterdatei site-start.el
- Emacs: DTDs zum Einbinden dtds.zip
- Dateidatum ändern: touch.exe
- Rechtschreibkontrolle: ispell-win32.tar.gz

7.2 Aufbau eines größeren XML Dokumentes auf Basis der contentobject DTD

Größere Dokumente gliedern sich oft in mehrere Abschnitte, die in verschiedenen Dateien gespeichert werden können, um eine übersichtlichere Bearbeitung zu ermöglichen. Für die Umsetzung wird hier das Arbeiten mit sogenannten Entitäten vorgestellt. Innerhalb eines Dokumentes werden Entitäten über das kaufmännische Und (&), den nachfolgenden Namen und ein abschließendes Semikolon referenziert. Während des Verarbeitungsprozesses wird dieser Eintrag einfach gegen den kompletten Inhalt der Entität getauscht.

Das contentobject gliedert sich zum Beispiel in chapter, die in externe Entitäten ausgelagert werden können. Ein Dokumentausschnitt könnte wie folgt aussehen:

```
<contentobject>
  <title>Dokumentation der Content DTD</title>
  <!-- Referenz auf die Entität chap_einleitung -->
  &chap_einleitung;
  <!-- Referenz auf die Entität chap_structure -->
  &chap_structure;
</contentobject>
```

Die Entitäten selbst werden innerhalb der DTD definiert. Da man aber nicht immer die DTD ändern möchte, wenn wann einzelne Entitäten definiert, besteht die Möglichkeit die DTD zu erweitern, dazu wird vor das abschließende > in <!DOCTYPE contentobject PUBLIC "-//Portiko//DTD Content v1.1 20021018//EN" "dtd/content.dtd"> ein Klammernpaar [ ] eingefügt, innerhalb des Klammerpaares werden dann externe Entitäten deklariert, hinter dem Schlüsselwort SYSTEM wird der Dateiname angegeben. Im nachfolgenden Beispiel werden die beiden Entitäten chap_einleitung und chap_structure definiert, die beiden Dateien liegen im selben Verzeichnis wie das Hauptdokument:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="styles/contentobject.xsl"?>
<!DOCTYPE contentobject PUBLIC "-//Portiko//DTD Content v1.1 20021018//EN"
"dtd/content.dtd" [
  <!ENTITY chap_einleitung SYSTEM "chap_einleitung.xml">
  <!ENTITY chap_structure SYSTEM "chap_structure.xml">
]>
```

Wichtig ist, dass sich die eigentliche DTD unterhalb des Hauptdokumentes im Ordner dtd befindet. Der Eintrag mit der DTD darf nicht auskommentiert werden, sonst können die Entitäten später nicht aufgelöst werden. In der Datei chap_einleitung.xml wird dann mit dem passenden XML Element begonnen:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<chapter id="chap_einleitung" name="Einleitung">
  <title>Einleitung</title>
  <sect1 id="sect_first" name="Anliegen">
    <title>Anliegen</title>
    <!-- one of (sect2 body) -->
  </sect1>
</chapter>
```

Damit geänderte Entitäten richtig ausgewertet werden, muß das Hauptdokument modifiziert werden. Das kann z.B. unter UNIX mit touch meinhauptdok.xml in der Shell muß geschehen. Für Windows gibt es dieses Programm auch: touch.exe. Es muß über den Systempfad erreichbar sein und kann dann in der DOS-Box (Aufruf: touch meinhauptdok.xml) genutzt werden.

7.3 Kontrolle der Rechtschreibung

7.3.1 Allgemeine Hinweise

Im Rahmen des Projektes Portiko wird zu Fragen der XML-Erstellung Support für den Editor Emacs angeboten. Neben speziellen Erweiterungen für den Umgang mit XML-Dateien unterstützt Emacs ebenfalls die Rechtschreibkontrolle. Dazu wird das externe Programm ispell genutzt. Diese Software wurde für UNIX und Derivate entwickelt, nähere Informationen gibt es unter http://fmg-www.cs.ucla.edu/geoff/ispell.html. Hier sind auch Portierungen auf andere Betriebssysteme und Wörterbücher zu finden.

7.3.2 Installation von ispell

Die Installation eines etwas älteren Win32-Ports verzichtet auf UNIX Emulatoren (z.B. Cygwin), das Archiv ispell-win32.tar.gz muß auf Laufwerk C:\ entpackt werden. Dabei wird der Pfad ispell automatisch angelegt. Die Umgebungsvariable PATH muß erweitert werden:

```
C:\ispell\bin
```

Im Archiv ist bereits ein deutschsprachiges Wörterbuch im Verzeichnis C:\ispell\libenthalten.

Abschließend ist Emacs noch für den Einsatz des deutschsprachigen Wörterbuches zu konfigurieren. Prinzipiell kann das Wörterbuch immer über den Menüeintrag Tools -> Spell Checking -> Select Deutsch8 Dict ausgewählt werden. Mit einigen Einträgen in der Datei Emacs-21.x\site-lisp\site-start.el kann dieser Wert voreingestellt werden:

```
;; ispell Einstellungen
(setq ispell-skip-sgml t)             ; Ueberspringen von SGML-Markup
(setq ispell-skip-html t)             ; Ueberspringen von HTML/XML-Markup
(setq ispell-dictionary "deutsch8")   ; Auswahl des Woerterbuches
(setq ispell-personal-dictionary "~/.ispell_deutsch")
                                      ;   Definieren   des   persoenlichen
Woerterbuches
                                      ; ~ bezieht sich auf die Variable
HOME
                                      ; es kann auch ein beliebiger
                                      ; Pfad gewaehlt werden
(setq flyspell-default-dictionary "deutsch8")
```

```
(put 'xml-mode 'flyspell-mode-predicate 'sgml-mode-flyspell-verify)
                                        ;      Einstellungen      fuer      den
flyspellmodus
```

Nun sollte das Wörterbuch mit Emacs einsatzbereit sein.

7.3.3 Nutzung von ispell

Für die Nutzung von ispell in Emacs ist im zu prüfenden Dokument einfach der Menübefehl Tools -> Spell Checking -> Spell-Check Buffer aufzurufen oder die Tastenkombination ALT + x ispell auszuführen. Die bei der Überprüfung möglichen Kommandos von ispell können über den Menüpunkt Tools -> Spell Checking -> Help aufgerufen werden.

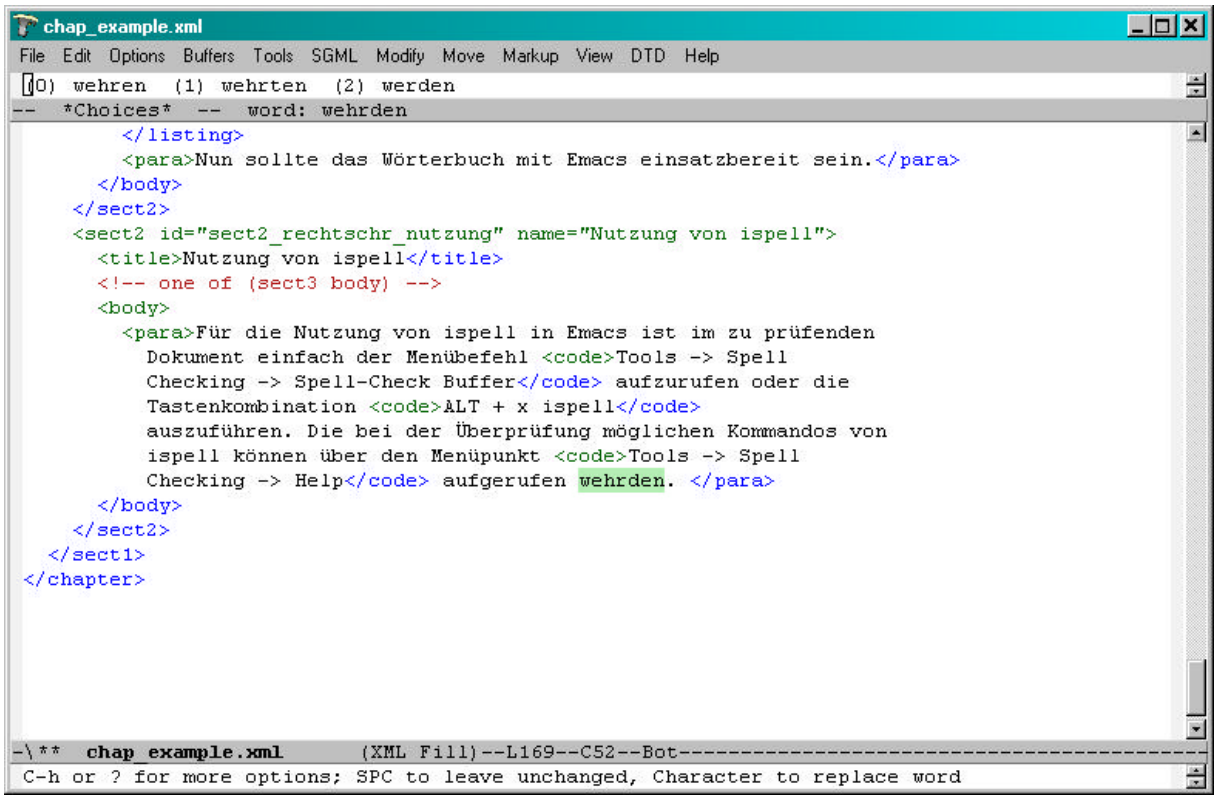

Abbildung 7.1 Rechtschreibkontrolle mit Emacs und ispell

Bei fehlerhaften Worten werden im oberen Fenster von Emacs gefundene Korrekturmöglichkeiten angeboten, die durch Auswahl über die entsprechende Ziffer in den Text übernommen werden können. Die wichtigsten Tastenkürzel sind:

[0..9] das Wort durch den entsprechenden Vorschlag im oberen Buffer *Choices*ersetzen

i ein unbekanntes Wort in das eigene Wörterbuch übernehmen

a ein unbekanntes Wort für den Verlauf der Sitzung akzeptieren

Space das unbekannte Wort einmal akzeptieren

r das unbekannte Wort in der unteren Befehlszeile von Emacs korrigieren und dann in den Text übernehmen

Beim Schreiben kann ein einzelnes Wort mit Hilfe der Tastenkombination ALT + $ geprüft werden. Dabei ist es ausrecihend, daß der Eingabecursor im Wort bzw. unmittelbar hinter dem Wort steht.

Die Rechtschreibkontrolle läßt sich auch permanent aktivieren, dazu muß der flyspell-Modus durch die Tastenkombination ALT + x flyspell-mode aktiviert werden. Alle fehlerhaften Worte werden farblich hervorgehoben und können nun korrigiert werden.

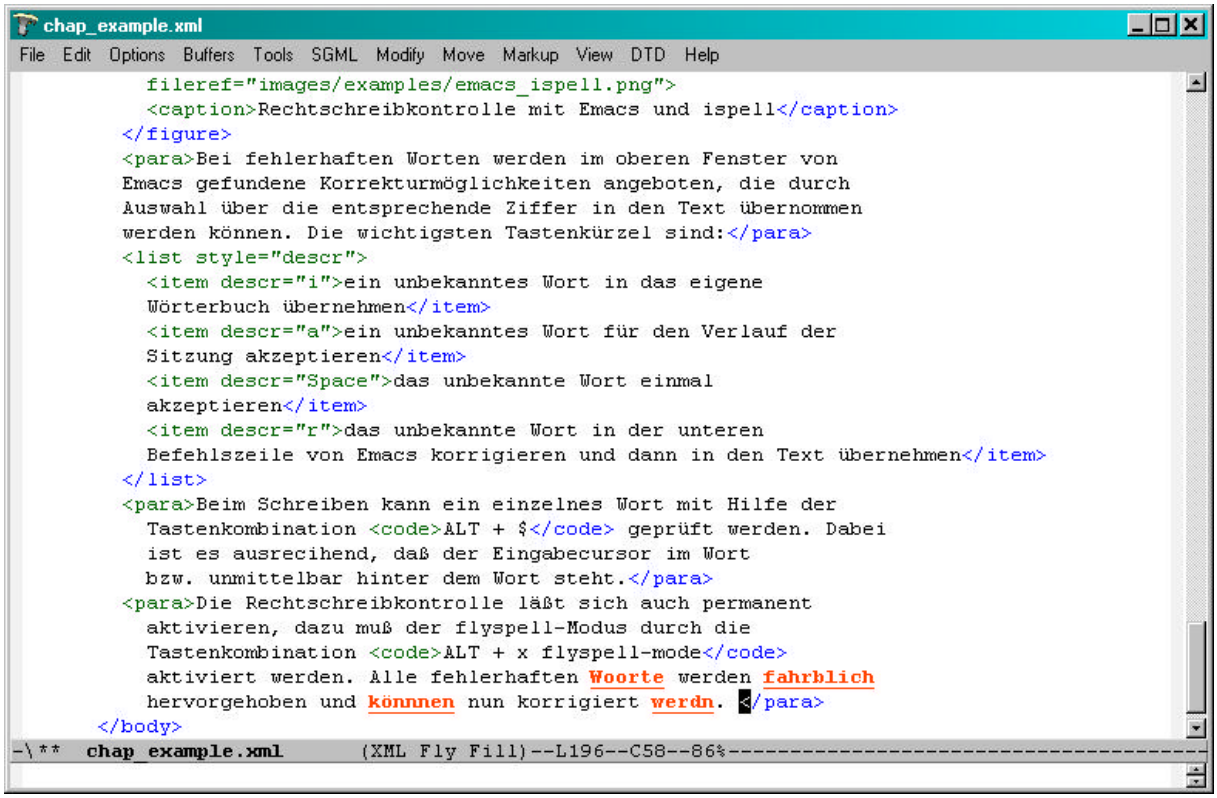

Abbildung 7.2 Rechtschreibkontrolle mit Emacs und flyspell-mode