Prof. Dr. Matthias Hemmje

Modul 63200

D2: Informatik

Lehrveranstaltung

Daten und Dokumentenmanagement im Internet

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Inhalt

Kurseinheit 1 Das Modell der strukturierten Dokumente

Kurseinheit 2 – Dokumentbeschreibungssprachen

Einleitung

Dies ist die erste Kurseinheit des Kurses 1873 „Daten und Dokumentenmanagement im Internet". Der Kurs ersetzt den Kurs 1865 „Datenformate im Internet" vollständig, baut auf dessen Struktur und Inhalten, die von den Autoren A. Brüggemann-Klein und J. Keller in vielen Bereichen übernommen wurden, auf und erweitert und aktualisiert diesen um wesentliche Inhalte aus dem Bereich der XML-basierten Sprachfamilien. Dank gilt an dieser Stelle dem Fernstudenten Hartwig Massen für wertvolle Kommentare und Korrekturen zur vorliegenden Überarbeitung.

Motivation und Einführung

Die adäquate Repräsentation und Präsentation von Daten und Dokumenten bildet eine wichtige Grundlage für deren effektive Erfassung, Verwaltung, Austausch, Vermittlung und Nutzung. Die Verwendung geeigneter Standards erleichtert dabei den Austausch von Daten und Dokumenten zwischen Menschen, zwischen Menschen und Maschinen sowie zwischen Maschinen. Solche Standards stellen ein gemeinsames Grundverständnis unabhängig von einer individuellen Aushandlung von Formaten sicher und ermöglichen damit auch dynamische und wechselnde Kooperationsbeziehungen, die in unserer durch das Internet global vernetzten und sich kontinuierlich veränderten Welt an Bedeutung gewinnen.

Der Schwerpunkt dieses Kurses

Ein wichtiges Ziel des Kurses ist es, in die Anwendung und aktuelle Entwicklung von verteilten Faktendatenbanken und Dokumentdatenbanken einzuführen sowie Kenntnisse zu allgemein strukturierten Dokumentenkollektionen für das Internet unter Nutzung des World Wide Web zu vermitteln. Zunächst werden allgemeine Formate für Dokumente sowie für strukturierte Dokumente, die im Internet verfügbar sind vorgestellt. Der Dokumentenbegriff wird erläutert und das Modell der strukturierten Dokumente eingeführt. Als übergreifendes Format, das die Definition konkreter Websprachen wie XHTML erlaubt, nimmt die Extensible Markup Language zusammen mit Dokumententypdefinitionen und den „Satelliten"-Standards XML-Schema, XPath, XPointer, XLink und XQuery eine zentrale Stellung ein. Ihre Darstellung bildet einen weiteren Kern des Kurses. Der Kurs beinhaltet auch eine Einführung in die Praxis dynamischer Webdokumente. Als Stichworte seien hier XSLT genannt und für spätere Versionen des Kurses evtl. noch CGI, Dynamic HTML, JavaScript, SAX und Dom, sowie eher Service-orientierte Ansätze auf Basis der gerade entstehenden Web

2.0-Technologien wobei auch näher auf HTML und CSS sowie auf Ontologie eingegangen werden wird.

Lernziele

Das hauptsächliche Lernziel dieses Kurses ist der Erwerb von praktischer Kompetenz und Handlungsfähigkeit im Bereich der aktuellen Daten- und Dokumentenmanagementtechnologien im Internet. Nach Bearbeitung des Kurses sollten die besprochenen Formate gründlich bekannt sein. Dokumente in diesen Formaten sollten erstellt werden können. Eine Übersicht über die zur Verfügung stehenden Werkzeuge sollte bekannt sein und mit einigen von diesen Werkzeugen sollte auch praktische Erfahrung gesammelt werden. Der Kurs soll dazu befähigen, Projekte mit zu konzipieren, technisch zu betreuen und durchzuführen, die die Erstellung und Verwaltung von Daten- und Dokumentenbeständen im Internet zum Ziel haben.

Ein weiteres sehr wichtiges Lernziel dieses Kurses betrifft das Modell der strukturierten Dokumente. Das Modell soll verinnerlicht und seine Bedeutung im Kontext der Informationsgesellschaft beurteilt werden können. Nach dem Durcharbeiten des Kurses sollten Sie wissen, dass strukturierte Dokumente wie gewöhnliche Daten von und mit Anwendungssystemen bearbeitet werden können. Es soll darüber hinaus vermittelt werden, wie Internetformate das Potential, welches in dem Modell der strukturierten Dokumente steckt, realisieren und wie diese Erkenntnisse von Ihnen auch praxisbezogen angewendet werden können.

Schließlich soll der Kurs auch einen Ausblick auf mögliche künftige Entwicklungen eröffnen und insbesondere auch den Anschluss an den Kurs 1874 „Informations- und Wissensmanagement im Internet" erleichtern. Hier lauten die wichtigen Fragen: Was leisten die gegenwärtig verwendeten Formate? Wo sind ihre Grenzen? Welche Weiterentwicklungen wären wünschenswert? Welche Entwicklungstrends sind insbesondere mit Blick auf das Semantic Web und auf das Web 2.0 absehbar?

Zu den vorgestellten Formaten ist jeweils eine systematische und grundlegende Darstellung vorgesehen, die bei besonders zentralen Formaten um Übersichtstabellen ergänzt wird. Daneben wird auf die Originalspezifikationen des W3C sowie auf reichlich vorhandenes tutorielles Material im Internet selbst verwiesen.

Die im Kurs eingeführten Daten- und Dokument-Formate werden im Text ausführlich durch Beispiele illustriert. Jedes Beispiel ist für sich lauffähig. Der Code wird in elektronischer Form zur Verfügung gestellt, sodass dieser ausprobiert und auch selbst modifiziert werden kann. Darüber hinaus kann der

Lernfortschritt fortlaufend anhand von in den Text eingestreuten Selbsttestaufgaben überprüft werden.

Formalia

Um die Praxisteile dieses Kurses bearbeiten zu können, wird ein Arbeitsplatzrechner mit Internetzugang benötigt. Darüber hinaus benötigen Sie Grundkenntnisse im Umgang mit einem Betriebssystem und Sie sollten die Möglichkeit haben, mit einem Texteditor einfache Textdokumente im ASCII-Format zu erstellen bzw. zu bearbeiten und abzuspeichern. Als Texteditoren kommen Programme wie Notepad, vi, Edit oder Emacs in Frage. Bitte aktualisieren Sie auch Ihren Internetbrowser. Der Umgang mit Navigation und Suche im Internet wird als bekannt vorausgesetzt. Sollten dennoch Schwierigkeiten mit der Nutzung des WWW und dem Internet im Allgemeinen bestehen, hilft eventuell ein Blick in die Broschüre [5] der Fernuniversität Hagen weiter.

1 Das Modell der strukturierten Dokumente

In der ersten Kurseinheit des Kurses 1873 wird zunächst der Dokumentenbegriff eingeführt und näher erläutert. Die Erläuterung verdient ein besonderes Augenmerk, denn der Dokumentenbegriff spielt eine zentrale Rolle in der hier dargelegten Thematik.

1.1 Der Dokumentenbegriff

Für statisch-passive Dokumente wurde bereits in den achtziger Jahren das *Modell der strukturierten Dokumente* entwickelt. Durch dieses Modell wird eine Unterteilung in textuellem Inhalt, logischer Struktur und grafischem Format möglich. Die *Quellrepräsentation* des Dokuments bezieht sich auf den textuellen Inhalt und logische Struktur. Gesteuert von unabhängigen Formatvorgaben, so genannten *Stylesheets*, lässt sich daraus das graphische Format generieren.

Quellrepräsentation

Moderne Systeme zur Erstellung und Verwaltung von Dokumenten und großen Dokumentenbeständen profitieren vom Modell der strukturierten Dokumente, da es ein hohes Maß an Flexibilität bietet. Die logische Strukturierung unterstützt Anwendungen der Dokumentenverwaltung wie die automatische Generierung von Katalogeinträgen und Verzeichnissen oder das *Information Retrieval* nach bestimmten logischen Kriterien und erlaubt es ganz allgemein, Dokumente als Daten zu betrachten, die mithilfe von Computersystemen verwaltet und manipuliert werden können.

Stylesheets

Die Unabhängigkeit der Stylesheets von den Dokumenten ermöglicht auf der einen Seite, dass ein- und dasselbe Dokument durch Kombination mit verschiedenen Stylesheets passend zum Anwendungszweck formatiert werden kann; auf der anderen Seite kann man verschiedene Dokumente durch Kombination mit ein- und demselben Stylesheet einheitlich formatieren. Auch lässt sich die Struktur beim Erstellen und Lesen von Dokumenten nutzen, etwa zur Navigation und zur Definition von Sichten etwa von *Outline-Sichten*.

1.2 Der Modellbegriff

Modelle sind ein wichtiges, methodisches Mittel. Die Modellierung ist eine vereinfachte und auf das im aktuellen Kontext Wesentliche reduzierte und - unter geeigneten Gesichtspunkten - systematisierte Darstellung. Modelle sind aus vielen Bereichen bekannt wie etwa aus der Architektur, der Biologie oder der Physik. Modelle können physikalischer oder gedanklicher Art sein; letztere können umgangssprachlich oder formal definiert sein. Gegenstandsbereich einer Modellierung können Konzepte, Vorgänge oder Erscheinungen und ihre Wechselwirkungen sein. In der Softwareentwicklung spielen Modelle bei der Anforderungsdefinition und im Design eine zentrale Rolle. Sie dienen der Kommunikation zwischen Ingenieuren und ihren Kunden und als Ausgangspunkt der weiteren Entwicklung. Die Softwaretechnik entwickelt Vorgehensmodelle auf verschiedenen Ebenen für den Prozess der Softwareentwicklung wie etwa das zwischenzeitlich veraltete Wasserfallmodell, das aktuellere Spiralmodell und neuerdings Verfeinerungen rund um die Modellierungssprache UML. Grundlage relationaler Datenbanksysteme ist das Tabellenmodell für Daten. Datenbankanwendungen selbst basieren auf einer Modellierung der Anwendungsdaten, meistens einem Entity-Relationship-Modell.

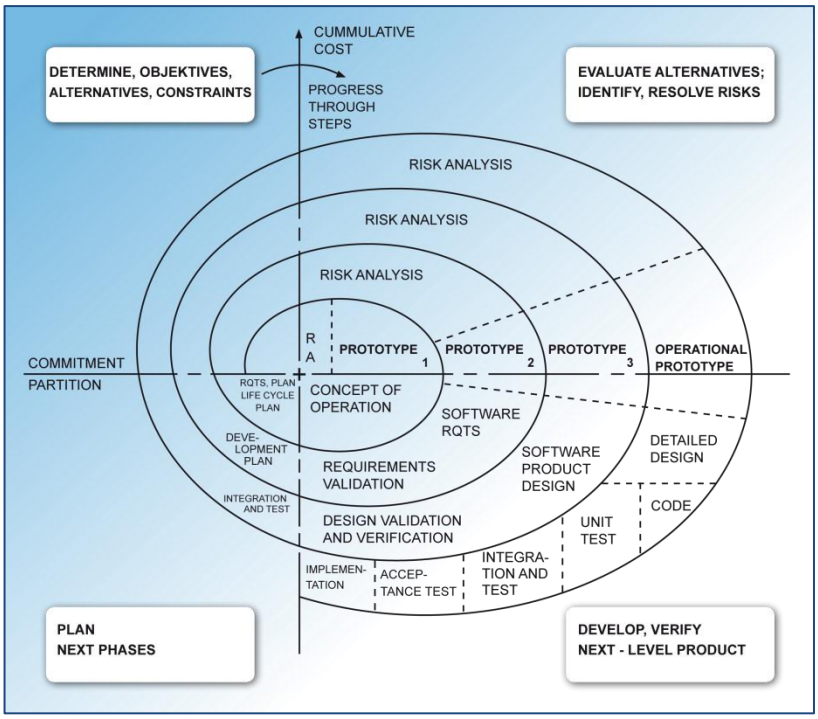


Abb.: 1.1 Das Spiralmodell in der Softwareentwicklung nach Böhm [2]

Modelle können unterschiedlichen Zwecken dienen: etwa der Entwicklung einer gemeinsamen Sicht und Terminologie, der Vermittlung erprobter Vorgehensweisen und Techniken, dem rationalen Beweisen oder Widerlegen von Hypothesen oder der Ableitung von Handlungsempfehlungen.

Zwei zentrale und umgangssprachlich erläuterte Modelle werden in dieser Kurseinheit eingeführt und näher erläutert: das Modell der strukturierten Dokumente und ein Modell für Zeichenkodierungen.

Die Dokumentenformate, die im weiteren Verlauf des gesamten Kurses hinzukommen, basieren ebenfalls auf Modellen. Bei der Definition von Daten- und Dokumentformaten im Internet könnte eine Konzentration auf die Modellierung kürzere, genauere, verständlichere und leichter handhabbare Sprachdefinitionen hervorbringen als es derzeit der Fall ist.

In diesem Kurs werden deshalb Modelle im Mittelpunkt und konkrete *Instanzen* dieser Modelle im zweiten Rang stehen, um dadurch ein vertieftes Verständnis zu ermöglichen.

1.3 Realisierung des Dokumentenmodells

Hier wird anhand verschiedener Beispiele und Systeme erläutert, wie das Modell der strukturierten Dokumente mit Zeichenkodierungen, eingebettetem *Markup*, *Dokumentengrammatiken* und Stylesheets in der Praxis umgesetzt wird. Es

folgen einige Beispiele für Sprachen und Formate sowie für Editier- und Formatiersysteme. Darüber hinaus folgt ein Überblick über weitere, im Internet einsetzbare Dokumentenmodelle und Formate. Auf der technischen Seite wird das Thema der Zeichenkodierungen vertieft. Dazu wird der *Unicode*-Standard vorgestellt.

In den nachfolgenden Kurseinheiten wird das Thema der strukturierten Dokumente anhand gängiger Internetformate beleuchtet. Dazu gehören *XML* und die zugehörigen weiteren XML-basierten Sprachfamilien.

1.3.1 Dokumente näher betrachtet

Historisch gesehen findet sich beispielsweise in Brockhaus' Konversationslexikon in der vierzehnten Auflage von 1894 folgendes unter dem Begriff „Dokument":

Historische Dokumentendefinition

„Im weiteren Sinne jeder Gegenstand, welcher dazu dient, die Wahrheit einer zu erweisenden Tatsache, besonders einer für ein Rechtsverhältnis erheblichen Tatsache, zu bestätigen. Im engeren Sinne versteht man darunter Urkunden oder Schriftstücke, im Gegensatz zu anderen körperlichen Beweisstücken, wie Grenzsteinen, Wappen, beschädigten Sachen."

Ursprünglich waren Dokumente also tatsächlich physische Objekte, Gegenstände, die als einmalige Träger eines in menschliche Sprache gekleideten und schriftlich festgehaltenen Gedankens oder Sachverhalts fungierten. Ein Dokument neuerer Art, etwa ein sauber getippter Bogen Papier mit Name, Adresse, Personalausweisnummer und weiterer Information zur Person erfüllt einfach nicht denselben Zweck, auch wenn er dieselbe sprachliche Information enthält. Die Dauerhaftigkeit der Schriftform gegenüber der gesprochenen Sprache, die Kommunikation unabhängig von Zeit und Raum ermöglicht, und die Authentizität des Trägermediums, die durch Wasserzeichen, Stempel, Siegel, Barcode oder ähnliches erreicht werden kann, machen das Wesen von Dokumenten in diesem ursprünglichen Sinne aus.

In der Brockhaus Enzyklopädie in der siebzehnten Auflage von 1968 findet sich dagegen unter dem Stichwort „Dokument":

Definition des Begriffes „Dokument"

„Als Dokumente können alle Unterlagen betrachtet werden, die Informationen beinhalten, also nicht nur publiziertes Wissen, sondern auch Briefe, Akten, Urkunden, Bildsammlungen, Filme u.v.a."

Die Dokumente neuerer Art sind also Informationsträger. Die Schriftform ist immer noch wesentlich, aber das Trägermedium und die Art, wie die Information auf das Trägermedium aufgebracht ist, sind nebensächlich. Auf den sprachlichen Ausdruck kommt es an.

Den Unterschied zwischen dem ursprünglichen und dem neueren Dokumentenbegriff kann an zwei Arten von Delikten klargemacht werden. Das Fälschen eines Personalausweises oder einer Banknote beinhaltet immer auch eine Fälschung des Trägermediums und betrifft Dokumente im ursprünglichen Sinne. Kopieren hingegen – sei es Fotokopieren oder Abschreiben – generiert immer ein zweites Exemplar des Dokuments, selbst wenn sich die Erscheinungsform dabei total ändert. Die Umstände, unter denen ein solches Kopieren zulässig ist, regelt das Urhebergesetz.

Das Gemeinsame an Dokumenten ursprünglicher und neuerer Art ist, dass sie durch ihre Schriftform eine ursprünglich sprachliche Mitteilung von der auditiven in die visuelle Übertragungsform und von der Flüchtigkeit der mündlichen Erzählung in die Dauerhaftigkeit des geschriebenen Wortes umsetzen. Somit werden die Grenzen von Zeit und Raum, die sonst der Senderin und dem Empfänger flüchtiger sprachlicher Mitteilungen gesetzt sind, überschritten.

Dokumente ursprünglicher und neuerer Art unterscheiden sich auch dadurch, wie viele Exemplare es von einem Dokument geben kann und wie viele Leserinnen und Leser gleichzeitig Gebrauch von so einem Dokument machen können.

Eine weitere und wichtige Unterscheidung, die zu einem dritten, dem modernen Begriff des digitalen Dokuments führt, hängt an der Idee vom Dokument als Informationsträger. Sprachlich formulierte Gedankengänge, Ideen, Fakten oder Aussagen werden erst dann zu *Information*, wenn sie von jemandem aufgenommen und verarbeitet werden.

Wissen in Aktion, ist Information

„Information ist Wissen in Aktion" lautet die Kurzformel der Informationswissenschaft. Nur wie ist „jemand" eigentlich definiert? In der Natur der Dinge liegt es dabei, zunächst an ein menschliches Gegenüber zu denken. Wie verhält es sich jedoch mit Computern, die ja dazu gedacht sind Informationen entgegen zu nehmen, weiterzuverarbeiten und anschließend verändert wieder auszugeben, wie dies ja bereits durch das *EVA* Prinzip definiert wird. Informationsverarbeitung mithilfe von Computerprogrammen führt demnach unmittelbar zum modernen Dokumentenbegriff.

Wesentliche Tätigkeiten, die per Computer dazu beitragen den modernen Dokumentenbegriff zu prägen:

- Gedankengänge organisieren und Dokumente planen
- Gedanken verbalisieren und Dokumente erfassen
- Dokumente editieren, formatieren und publizieren, also der Öffentlichkeit zugänglich machen

- Dokumente ablegen, verwalten, abfragen, zusammenfassen, kombinieren, transformieren, extrahieren
- Dokumente lesen, verstehen, memorieren, aus ihnen lernen und über sie nachdenken.

Der moderne Dokumentenbegriff

Moderne Dokumente sind Informationsträger, die alle diese Funktionen effizient unterstützen. Moderne Dokumente müssen deshalb in ihrer Urform elektronisch gespeichert sein und je nach Anwendungszweck unterschiedliche Präsentationsmedien unterstützen.

Selbsttestaufgabe 1.1

Überlegen Sie, ob auf die folgenden Dokumente eher der ursprüngliche, der neuere oder der moderne Dokumentenbegriff passt oder ob es sich vielleicht gar nicht um Dokumente handelt:

- Personalausweis
- Geldschein
- Gemälde
- Veranstaltungsplakat
- Privatbrief
- E-Mail-Nachricht
- Memo an die Geschäftsleitung
- Gedichtband
- Roman
- Bestellung
- Produktbeschreibung
- Fragebogen
- Nachricht in einem Protokoll zum digitalen Zahlungsverkehr (Electronic Banking)

Nicht alle Beispiele lassen sich eindeutig einordnen. Machen Sie sich klar, welche Argumente für oder gegen eine bestimmte Einordnung sprechen.

Vorangehend wurde eine Übersicht über die wesentlichen Kategorien von Dokument bezogenen Begrifflichkeiten gegeben. Im weiteren Verlauf werden in diesem Kurs jedoch nur noch Dokumente der modernen Art besprochen.

Dokumente sind also nachfolgend immer elektronisch gespeicherte Informationsträger, deren schriftsprachlich festgehaltene Inhalte sowohl von menschlichen Leserinnen und Lesern als auch von Computerprogrammen aufgenommen und weiterverarbeitet werden können. Im nächsten Abschnitt wird analysiert, welche Anforderungen sich daraus für Dokumentenformate

ergeben und wie diese Anforderungen durch das Modell der strukturierten Dokumente erfüllt werden.

1.3.2 Moderne Textdokumente

Unter diesem Begriff werden Textdokumente verstanden die in weltweit verteilten Systemen wie dem Internet und dem World Wide Web, für die unterschiedlichen Nutzungsarten und Anwendungszwecke bereitgestellt werden. Einige Fragen die sich aus diesem Kontext ergeben sind:

- Welche Anforderungen ergeben sich für die Dokumentenformate?
- Welche Formate sind in der Praxis bereits jetzt verfügbar, welche werden sich kurzfristig durchsetzen?
- Welche Werkzeuge stehen zur Verfügung, um die Dokumente zu bearbeiten?
- Was können etablierende und etablierte Disziplinen wie Document Engineering, Content Management und Elektronisches Publizieren beitragen?

Haben Sie während des Lesens versucht diese Fragen zu beantworten? Die Fragestellungen sind dazu gedacht Ihnen ein Gespür für die Thematik, um die es in diesem Kurs geht, zu bekommen. Ausführlichere Antworten erfolgen im Kursverlauf.

Noch vor wenigen Jahren gab es nur ein relevantes Dokumentenformat im Web, nämlich die *H*yper*t*ext *M*arkup *L*anguage. Einführungen in das Erstellen von HTML-Dokumenten umfassten einschließlich Darstellungen zur Geschichte des Internet und anderer historischer Anmerkungen gut hundert Buchseiten. HTML

Die erste konsolidierte und von der *I*nternet *E*ngineering *T*ask *F*orce verabschiedete HTML-Version 2.0 umfasste ca. 150 Sprachelemente, wobei mit Sprachelementen sowohl die Elemente als auch die Attribute gemeint sind. Die besagte HTML-Version wurde im November 1995als RFC 1866[1] eingeführt. Elektronisch arbeitende Produzenten und Verleger von Inhalten so genannte *Content Provider* konnten mit einer Handvoll von HTML-Befehlen auskommen, um eine dem aktuellen Stand der Technik entsprechende Webseite zu erstellen. IETF RFC

[1] http://rfc-ref.org/RFC-TEXTS/1866/index.html

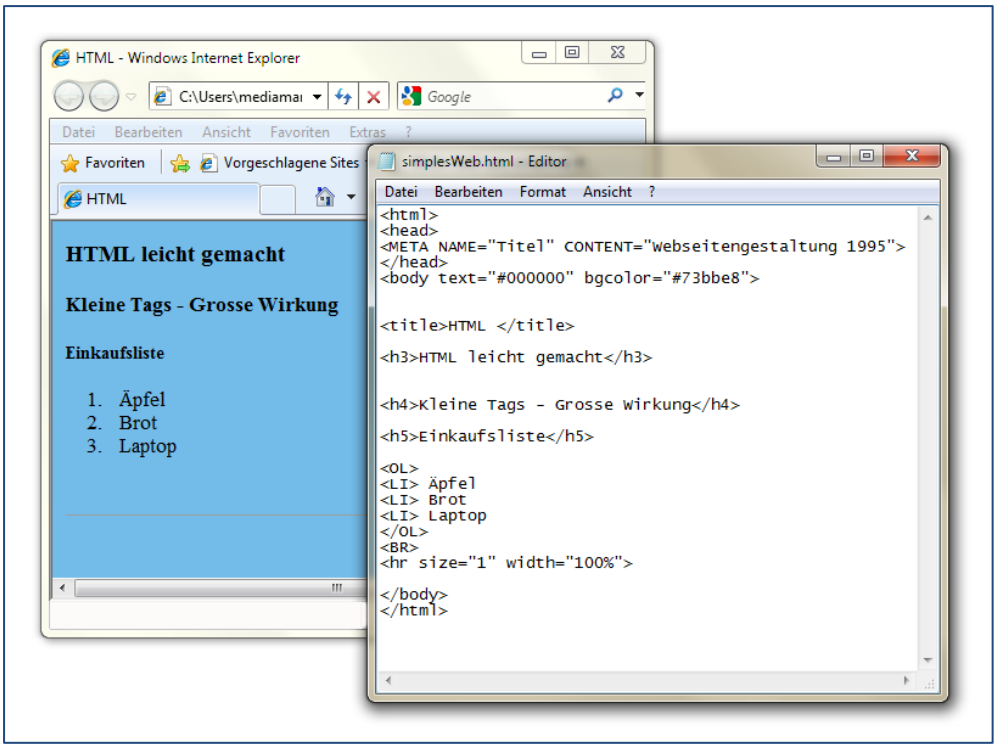


Abb.: 1.2 Webseitengestaltung von 1995

Die HTML-Version 4.0, welche am 24. April 1998 vom *W*orld *W*ide *W*eb *C*onsortium verabschiedet wurde, umfasste dann bereits ca. 1.600 Sprachelemente; an für Textdokumente relevanten Formaten sind zu HTML die Sprachen: W3C

- *E*xtensible *M*arkup *L*anguage
- *C*ascading *S*tyle *S*heets
- *E*xtensible *S*tyle *L*anguage
- XML Schema
- XML Linking Language
- XML Path Language
- XML Pointer Language

hinzugekommen und es erschien das einschlägige Handbuch von Danny Goodman[1] zu der nun programmierbaren HTML-Erweiterung *D*ynamic *HTML*. Mittlerweile erfordert es über 1.000 Buchseiten, um alle systemspezifischen Varianten und Strategien und deren Handhabung in einem weltweiten Publikationsmedium darzustellen. dHTML

Neben den statisch-passiven Textdokumenten, die klassischen auf Papier gedruckten Dokumenten ähnlich sind, gewinnen im Internet in zunehmendem

[1] http://www.dannyg.com/pubs/index.html

Maße dynamisch-aktive Textdokumente an Bedeutung. Sie enthalten aktive Komponenten, mit denen Leserinnen und Leser interagieren können. Es können sich auch der Inhalt und das Format dynamisch ändern. Beispiele für aktive Komponenten sind mathematische Formeln, die mittels eines Computer-Algebra-Pakets auf Knopfdruck evaluiert werden können. Auch Visualisierungen von Algorithmen sind so möglich, wie etwa in diesem Beispiel[1]. Weitere Beispiele für dynamische Dokumente sind sich regelmäßig aktualisierende Börsenkurse oder auf Knopfdruck expandierbare Outline- Sichten.

Zahl, Umfang und Funktionsbereich der relevanten Empfehlungen des W3C[2] und deren Implementierungen haben bis heute derart zugenommen, dass eine systematische Einführung in die Sprachen selbst und in erprobte Praktiken der professionellen und systematischen Dokumenterstellung und Verarbeitung einen organisierten Einstieg in die Internetpraxis sehr erleichtert.

Die neuen Dokumentenformate für statisch-passive Dokumente im Internet folgen ebenfalls dem Modell der strukturierten Dokumente. Neben HTML hat im Internet die Entwicklung im Kontext mit XML in den vergangenen Jahren immer mehr an Bedeutung gewonnen.

1.3.3 Dokumentenbestandteile

Dokumente moderner Art sind als Informationsträger zu sehen. Wie aber kommen die Leserinnen und Leser an die in einem Dokument enthaltene Information? Als Einstieg in Beantwortung dieser Frage kann zunächst einmal ein konkretes Dokument betrachtet werden: Die Journalseite in Abschnitt 1.6.1 im Anhang.

Es handelt sich hierbei um eine beispielhafte Annonce der Frauenbeauftragten der Technischen Universität München zu einem Computer-Propädeutikum.

Bei der Lektüre dieses Dokumentes lässt sich feststellen, dass eine Reihe von Themen vorgeschlagen werden, die in einem Kurs des Computer-Propädeutikums behandelt werden können. Wie genau aber wurde diese Information im Dokument kodiert, vom Leser wahrgenommen und gedanklich verarbeitet?

Die Ausgangsbasis sind die in dem Dokument enthaltenen Buchstaben und Interpunktionszeichen, die beim Lesen vom Menschen wie auch von Maschinen zu Wörtern und Sätzen zusammengesetzt werden. Darüber hinaus werden diese

---

[1] http://www.cs.auckland.ac.nz/software/AlgAnim/heapsort.html

[2] http://www.w3.org/

von Menschen aber auch intellektuell verarbeitet und mit Bedeutung versehen. Die Belegung von Symbolen mit Bedeutungen wird *Semantik* genannt. Der informationelle Inhalt des Dokuments wird also vom Leser erfasst. Dies ist eine Fähigkeit, die der Mensch den Maschinen zunächst voraus hat. In der letzten Phase der Erfassung gibt es also wesentliche Unterschiede zwischen der Art und Weise wie ein Mensch und wie eine Maschine Inhalte wahrnimmt, deren informationellen Gehalt erfasst und verarbeitet.

Es sind also zunächst nicht nur die Buchstaben und Interpunktionszeichen alleine, die den Erfassungsprozess ermöglichen. Zu dieser Einsicht verhilft folgender Gegentest: Betrachtet ein menschlicher Leser die unformatierte Version des gleichen Dokumentes, die sich unter 1.6.2 im Anhang befindet, und versucht aus dieser Version die Gliederung des Konzepts herauszulesen, so stellt sich dies als mühsam bis unmöglich heraus.

Offenbar unterstützt also die Formatierung des Dokuments – etwa die Wahl der Schriften (groß oder klein, fett oder kursiv), die Ausrichtung von Textpassagen (linksbündig, rechtsbündig oder zentriert) und die Abstände zwischen einzelnen Textpassagen – die visuelle Aufnahme und Verarbeitung der Information durch den Menschen. Geht man zurück zur zuerst betrachteten formatierten Version des Konzepts, so wird einem menschlichen Leser deshalb sofort die Gliederung in die Abschnitte „Hintergrund und Ziele", „Das Kursangebot des Computer-Propädeutikums" usw. klar. Die visuelle Formatierung der Überschriften in Fettschrift und mit deutlichen Abständen zum umgebenden Text geben ausreichende Hinweise um die Gliederungselemente zu erkennen.

Die Buchstaben, Interpunktionszeichen und sonstigen Symbole, die in einem Dokument enthalten sind, zusammen mit dem Format, also der typografischen, sprich visuellen Ausprägung der Symbole und der geometrischen Anordnung oder dem Layout, ermöglichen es also den menschlichen Leserinnen und Lesern, den Informationsgehalt eines Dokumentes leichter, d. h. kognitiv effizienter zu erfassen.

Was aber ist genau das Ergebnis dieses menschlichen Verarbeitungsprozesses? Zunächst einmal werden die Aussagen wörtlich erfasst und verstanden. Hinzu kommt jedoch auch das Erfassen der einzelnen Textpassagen und deren Rolle im Gesamttext. So identifiziert ein menschlicher Leser wahrscheinlich beim Betrachten der formatierten Version des Konzepts das Textstück „Ein Computer-Propädeutikum von Studentinnen für Studentinnen" als Überschrift des gesamten Dokuments und das Textstück „Hintergrund und Ziele" als Überschrift eines Abschnitts, der bis zur nächsten Überschrift reicht.

Zusätzlich zur Aussage eines Dokuments erkennen menschliche Leserinnen und Leser also aufgrund des Inhalts und des Formats eine Strukturierung in logische Einheiten. In dem Konzept zum Computer-Propädeutikum ließen sich

beispielsweise die folgenden Strukturelemente identifizieren: Briefkopf mit Namen der Autorinnen und Angaben zur Institution, Hauptüberschrift, Nebenüberschrift mit Datums- und Versionsangaben, Abschnitte mit Überschriften, Absätzen und Listen, Geldbeträge, Namen der Organisatorinnen und Datumsangaben.

Die von den Leserinnen und Lesern erkannte Struktur ist nicht eindeutig. Hinweise auf die Struktur ergeben sich aus Inhalt und Format aufgrund von dem semantischen Verständnis der Textpassagen und der kulturellen, durch den Umgang mit zahlreichen Dokumenten ähnlicher Art gewonnenen Erfahrung.

Für menschliche Leserinnen und Leser stellt sich also der Prozess der Informationsaufnahme und -verarbeitung folgendermaßen dar: Das Dokument wird mit Inhalt und Format präsentiert. Menschliche Leserinnen und Leser extrahieren daraus die Aussage des Dokumentes und konstruieren eine Struktur für die einzelnen Textelemente. Durch visuelle Wahrnehmung und gedankliche Arbeit unter Rückgriff auf das Sprachverständnis und das kulturelle Wissen wird die Bedeutung der Aussage interpretiert. Die Verwendung von Formatmerkmalen, Aussage und Struktur bilden die Grundlage für die weitere intellektuelle Informationsverarbeitung.

Ein Dokument kann in ganz unterschiedlichen grafischen Aufmachungen dargestellt werden. Die Darstellung hängt auch vom Präsentationsmedium und der Lesesituation ab. Beispielsweise kann ein Dokument für das ermüdungsfreie Lesen am Bildschirm mit einer anderen Farbpalette und mit anderen Schriften formatiert sein als für das Lesen von einem Papierausdruck; oder Querverweise können für die Onlineversion als Hypertextlinks und für die Druckversion als Hinweise auf Seitenzahlen realisiert werden.

Entscheidend für die „Identität" eines Dokumentes ist, dass aus den verschiedenen Formaten immer noch dieselbe Aussage und im Wesentlichen dieselbe Struktur erkannt wird.

1.3.4 Dokumente strukturieren

Bisher wurde das Erfassen der in einem Dokument enthaltenen Information von der Warte der menschlichen Leserinnen und Leser aus betrachtet. Was ist aber, wenn Computerprogramme zur Informationsverarbeitung herangezogen werden sollen? Wollte man den menschlichen Prozess des Textverstehens nachbilden, der bei einer formatierten und für das menschliche Wahrnehmungssystem „Auge" realisierten Version des Dokuments startet, so müsste man Computerprogramme mit intelligenten Fähigkeiten ausstatten, die Zeichenerkennung, Sprachverstehen und Strukturerkennung einschließen. Von

diesen Möglichkeiten ist man heute jedoch noch relativ weit entfernt, sie sind allerdings bereits Gegenstand intensiver Forschung.

Formatvorlagen zur automatischen Layout-Generierung

Der Ansatz der strukturierten Dokumente sieht folgenden Ausgangspunkt für die computergestützte Informationsverarbeitung vor: eine explizite *Repräsentation* von Inhalt und vor allem auch Struktur des Dokumentes im Sinne eines *Repräsentationsmedium*s. Autorin oder Autor eines strukturierten Dokumentes editieren den eigentlichen Inhalt des Dokumentes und markieren die notwendigen Strukturelemente. Das zu den Textelementen passende Format kann aus einer separaten, vom Dokument unabhängigen und bei Bedarf auswechselbaren Formatvorlage, auch *Stylesheet* genannt, berechnet werden.

Das Stylesheet enthält allgemeine Regeln, wie die einzelnen Strukturelemente zu formatieren sind. Durch eine Regel kann beispielsweise festgelegt werden, dass Überschriften in einer fetten Kursivschrift mit einem gewissen Abstand zum umgebenden Text formatiert werden sollen.

Ein so genannter *Formatierer* bringt das mit Strukturmarkierungen versehene Quelldokument mit dem Stylesheet zusammen und formatiert das Dokument entsprechend der Vorgaben in dem Stylesheet.

1.3.5 Vorteile des Dokumentenmodells

Ein solches Vorgehen hat als entscheidenden Vorteil die Flexibilität, mit welcher die Formatierung der Texte immer wieder geändert werden kann. Die logische Strukturierung unterstützt aber auch Anwendungen der Dokumentenverwaltung, sodass etwa die automatische Generierung von Katalogeinträgen und Verzeichnissen möglich wird oder das Wiederfinden von Information nach bestimmten logischen Kriterien. Die eigentliche Informationsverarbeitung kann sich also direkt auf die logische Struktur stützen. Die Unabhängigkeit der Stylesheets von den Dokumenten ermöglicht, dass ein Dokument durch Kombination mit verschiedenen Stylesheets passend zum Anwendungszweck formatiert werden kann. Außerdem können verschiedene Dokumente durch Kombination mit ein- und demselben Stylesheet einheitlich formatiert werden. Auch lässt sich die Struktur beim Erstellen und Lesen von Dokumenten nutzen, etwa zur Navigation und zur Definition von Sichten etwa Outline-Sichten. Von diesen Vorteilen profitieren besonders die Anwendungen, die große Dokumentenbestände erstellen und verwalten müssen.

Eine umfangreiche Sammlung von Projektvorschlägen, die alle nach dem gleichen Muster logisch strukturiert und markiert sind, kann systematisch nach anwendungsnahen Kriterien abgefragt werden. Auch etwa die im letzten halben Jahr neu hinzugekommenen Vorschläge können mit Titel und Datum ausgegeben werden. Eine weitere Anwendung wäre, automatisch nach allen Projekten zu suchen, die ein bestimmtes gemeinsames Ziel verfolgen. Auch für die Verwaltung des Dokumentenbestandes bietet das Modell der strukturierten

Dokumente Vorteile. Die mit der Verwaltung befassten Personen brauchen für den gesamten Bestand nur einmal die zu verwendenden Strukturelemente festzulegen. Für jedes Präsentationsmedium ist nur ein einziges Stylesheet nötig, welches dann auf alle Dokumente des Bestandes anwendbar ist. Für das Beispiel der Projektvorschläge würden zwei Stylesheets ausreichen, eines für das Lesen am Bildschirm und eines für das Steuern eines Ausdrucks.

Das Modell der strukturierten Dokumente wurde in den achtziger Jahren entwickelt [1, 5, 9]; seine Wurzeln gehen sogar bis in die sechziger Jahre zurück [4]. Alle modernen Textverarbeitungssysteme unterstützen das Modell zumindest in Ansätzen.

1.3.6 Verfeinerungen des Dokumentenmodells

Freie und vordefinierte Strukturelemente

Zum vollen Modell der strukturierten Dokumente gehört wesentlich dazu, dass die Namen der Strukturelemente frei wählbar sind. Wenn man beispielsweise ein Aufgabenblatt logisch strukturieren möchte, dann müssen die Strukturelemente wie Aufgabe, Abgabedatum, Punktezahl, Lösungshinweis oder Schwierigkeitsgrad angegeben werden können. Ist der Sprachvorrat, aus dem Strukturelemente gewählt werden können, fest vorgegeben und beschränkt, kann er nicht alle sich im Laufe der Zeit ergebenden Anwendungen abdecken. Dies verwässert allerdings die Vorteile der strukturierten Dokumente für die computergestützte Informationsverarbeitung, da semantische Information über die Rollen der Strukturelemente durch eventuelle Mehrfachbelegung verloren geht.

Historisch ist übrigens anzumerken, dass es u. a. im Verlagswesen mehrere Bestrebungen gegeben hat, einen universell einsetzbaren Vorrat an Strukturelementen zu definieren. Für einige Spezialgebiete konnte auch tatsächlich eine Einigung über einen solchen Vorrat erzielt werden. Das bekannteste Beispiel ist die Sprachdefinition der Text Encoding Initiative (TEI) für den Bereich der textkritischen Apparate im geisteswissenschaftlichen Bereich; der Sprachvorrat, der hier definiert wurde, ist allerdings sehr umfangreich. In der Regel sind solche Versuche jedoch gescheitert und zugunsten von frei wählbaren Strukturelementen aufgegeben worden.

Eine weitere Überlegung zum Modell der strukturierten Dokumente ist, ob der zugrunde liegende Vorrat an Strukturelementen formal definiert werden soll. Eine Strukturvorgabe ist eine mit formalen Methoden festgelegte Definition des Vorrats an Strukturelementen, die auch Einschränkungen zur Verwendung der Elemente beinhalten kann. Eine Strukturvorgabe hat Vorteile. Beispielsweise können Dokumente automatisch auf ihre strukturelle Korrektheit überprüft werden oder Editierhilfen können solche Vorgaben auswerten und zur Unterstützung der Autorin oder des Autors einsetzen. Schließlich wird durch

eine formale Strukturvorgabe auch das Vokabular einer Dokumentenanwendung dokumentiert.

1.3.7 Strukturdarstellungen

Zur Markierung der Strukturelemente gibt es im Wesentlichen zwei schon lange etablierte Konzepte. Bei beiden Konzepten wird davon ausgegangen, dass die logische Strukturierung streng hierarchisch angelegt ist. Strukturelemente können also andere Strukturelemente vollständig enthalten, sich aber nicht mit ihnen überlappen. Die Strukturelemente, die im Beispiel der Annonce zu dem Computer-Propädeutikums in Abbildung unter 1.6.1 eingeführt worden sind, gehorchen dem Gesetz der hierarchischen Schachtelung.

Bei dem ersten Konzept bzw. bei der ersten Methode wird die hierarchische Struktur graphisch dargestellt. Hier können zunächst drei Varianten vorgestellt werden. Abbildungen zu allen drei Varianten befinden sich im Anhang im Abschnitt 1.6.3.

Bei der ersten Variante werden ineinander geschachtelte Rechtecke oder Boxen, die mit dem Namen des jeweiligen Elementes gekennzeichnet sind, genutzt. Die zweite – dynamische - Variante, bei der mit Einrückungen und Ikonen gearbeitet wird, könnte aus Dateibrowsern bereits bekannt sein. Eine dritte – ebenfalls dynamische – Variante besteht darin, die logische Struktur als einen Baum darzustellen.

Die grafischen Möglichkeiten stellen hohe Anforderungen an die Formatierungsmöglichkeiten des darstellenden und ja auch möglichst noch Interaktionen zulassenden Systems.

Eine nicht so anspruchsvolle Möglichkeit besteht darin, die hierarchische Struktur mittels einer Klammerstruktur zu *linearisieren*. Zu diesem Zweck wird jeder Textbereich, der Inhalt eines logischen Elementes ist, an seinem Anfang und an seinem Ende mit einer öffnenden und einer schließenden Klammer versehen, die den Namen des entsprechenden Strukturelementes enthält. Eine solche Klammer ist einfach so genannter Klartext, der sich durch syntaktische Konventionen vom eigentlichen Inhalt abhebt. Der Inhalt ist unformatiert. Bei der XML-Konvention wird der Anfang eines *Tags* durch den Begriff der in spitzen Klammern steht signalisiert <XXX>. Soll das Tag nicht mehr gelten, wird der Schrägstrich eingesetzt: </XXX>. Der Klartext für die öffnende oder schließende Klammer heißt *Tag*, was Auszeichnung bedeutet. Der Oberbegriff zu Tags und eventuellen weiteren Möglichkeiten, nicht zum Inhalt zählende Zeichen in einen Text einzustreuen, ist *Markup*. Eine grafische Verfeinerung dieser Darstellung wird erreicht, wenn mit Einrückungen und mit typografischer Differenzierung von Text und Markup gearbeitet wird.

Da der Markup mit im Text enthalten ist, wenn auch über syntaktische Konventionen vom eigentlichen Inhalt getrennt, spricht man auch von einem eingebetteten Markup. Von einem prinzipiellen Standpunkt aus gesehen, sind alle Darstellungsformen für strukturierte Dokumente äquivalent. Die Repräsentation der hierarchischen Struktur durch eingebetteten Markup ist technisch am einfachsten zu handhaben und ist deshalb derzeit noch die gängigste. Sie hat den Vorteil, dass Dokumente mit eingebettetem Markup mit jedem einfachen ASCII-Texteditor angesehen und bearbeitet werden können. Die Darstellungsweise ist allerdings unübersichtlich und nicht sehr eingängig. Die anderen Darstellungsformen stoßen an ihre Grenzen, wenn die Schachtelungstiefe sehr groß wird oder die Granularität der Struktur sehr fein. Probleme tauchen unter anderem schnell bei der logischen Beschreibung mathematischer Formeln auf.

1.3.8 Diskussion des Dokumentenmodells

Das Modell der strukturierten Dokumente wird eingesetzt, um Dokumente für die Computerverarbeitung vorzubereiten. Die zentrale Idee ist, die Kerndaten des Dokuments redundanzfrei in einer Art Reinform vorzuhalten, wie man es auch beim Design relationaler Faktendatenbanken mit Normalformen verfolgt. Faktendaten, die sich aus dieser Quellform berechnen lassen oder die variabel gehalten werden sollen, werden nicht in der Quellform mit kodiert sondern bei Bedarf berechnet und gegebenenfalls als Sichten generiert. Verarbeitungsfunktionen können auf dieser Reinform der Daten aufsetzen und werden nicht belastet durch *Idiosynkrasien* im Datenbestand, die durch einzelne Anwendungen begründet sind.

Im Datenbankbereich dient diese Methode gleichzeitig der logischen Datenunabhängigkeit. Änderungen an der Struktur der Quelldaten können vor Endsystemen, die über Sichten mit der Datenbasis zu tun haben, verborgen bleiben. Das Potential der Datenunabhängigkeit ist durch das Modell der strukturierten Dokumente und seine Umsetzung mit XML auch im Bereich der Webdokumente gegeben. Allerdings ist die Technik noch nicht so weit fortgeschritten wie im Datenbankbereich, sodass etwa mit entsprechenden Zugangsvoraussetzungen ausgestattete Personen einen Bestand von XML-Dokumenten über mit XSLT-generierten Sichten bearbeiten könnten. In diesem Bereich sind noch interessante Forschungsarbeiten zu erledigen. Auch eine Theorie der Normalformen, wie sie im Datenbankbereich etabliert ist, gibt es für den Bereich der strukturierten Dokumente noch nicht.

Das Modell der strukturierten Dokumente dient dazu, Inhalt und Struktur eines Dokumentes, auf denen die automatische Verarbeitung basieren soll, zu repräsentieren. Dabei ist es wichtig, die Strukturelemente je nach Anwendungsfall frei definieren und die entsprechenden Definitionen auch formal

festschreiben zu können. Auf dieser Basis können Verarbeitungsfunktionen aufgesetzt und Sichten generiert werden, wobei die Basis je nach Anwendungsfall durch zusätzliche und separate Steuerungsdaten ergänzt werden kann.

Document Management

Historisch stammt das Modell aus dem Bereich der Dokumentenverarbeitung und Dokumentenverwaltung. Ein wichtiger Anwendungsfall in diesem Bereich ist die Formatierung. Vorgaben für die Formatierung werden in separaten und auswechselbaren Stylesheets vorgehalten und Formate aus beiden Beschreibungen zusammen berechnet. Über die Generierung formatierter Sichten hinaus sind Systeme des Document Management typischerweise in der Lage, Nummerierungen, Verzeichnisse und Querverweise aus der expliziten Repräsentation von Inhalt und Struktur zu erzeugen. Dies geschieht unter Hinzunahme von oft ebenfalls in Stylesheets enthaltenen Steuerungsdaten.

Knowledge Management

Das Modell unterstützt jedoch nicht nur Anwendungsfälle des Document Management im engeren Sinne, sondern trägt auch Anwendungen im weiteren Bereich des Wissensmanagements. Das Modell der strukturierten Dokumente dient als Stütze, den Schritt Texte zu gut zu verwaltenden Texten weiterzuentwickeln und die maschinelle Lesbarkeit zu ermöglichen.

Zudem sollen Bereiche im Inhalt eines Dokuments mit zusätzlicher semantischer Information markiert und die Rolle des entsprechenden Textstücks benannt werden. Das Modell platziert sich damit in einem Spektrum, das von einer rein grafischen Repräsentation eines Dokuments, etwa als Pixelmuster oder als PostScript-Dokument, bis zu einer expliziten Repräsentation der Semantik, etwa als semantisches Netz reicht.

Die rein grafische Repräsentation erfordert zur computergestützten Verarbeitung intelligente Software, die derzeit noch nicht in ausreichendem Umfang zur Verfügung steht. Eine ausgefeilte Repräsentation der Semantik erfordert einen hohen Overhead beim Erstellen und Bearbeiten der Dokumente. In der Praxis hat sich der Kompromiss, der bei der Konzeption des Modells eingegangen ist, seit langem bewährt.

Bei Dokumentensystemen, die das Modell der strukturierten Dokumente implementieren, werden die Strukturelemente in der Regel mit eingebettetem Markup kodiert. Das ist der zweite pragmatische Kompromiss im Bereich des Modells, der sich jetzt auf die Umsetzung des Modells bezieht. Zwar erscheinen immer wieder Positionspapiere mit Titeln wie „Embedded Markup Considered Harmful“ [10, 11], aber auch hier scheint der Kompromiss den Test der Zeit zu bestehen.

Was in diesem Kurs „Modell“ genannt wird, heißt in der älteren Literatur auch manchmal Dokumentenarchitektur. Der Begriff der Architektur macht klarer deutlich, worum es in einer ersten Annäherung an Dokumente und ihre Formate

geht: relevante Aspekte von Dokumenten sollen identifizierbar und das Zusammenwirken transparent gemacht werden

Es werden Überlegungen dazu angestellt werden, welche abstrakten Modelle den sich im Internet manifestierenden Formaten zugrunde liegen. Es steht dabei nicht immer für jede Ebene der Modellierung eine separate Begrifflichkeit – etwa wie hier „Architektur" – zur Verfügung. Bei dem noch folgenden Modell für die Zeichenkodierung wird klar, warum in diesem Kurs der Begriff Modell eingeführt wird und wir die Thematik nicht mit „Architektur" der strukturierten Dokumente erläutern.

1.3.9 Umsetzungen des Dokumentenmodells

Anhand der meisten modernen Textverarbeitungssysteme wird das Modell der strukturierten Dokumente in irgendeiner Form umgesetzt. Die beiden Systeme *LaTeX* und *MS-Word* decken vom Typ her zusammen das Spektrum von Textsystemen ab, die in akademischen und technischen Dokumentierungen sowie im Büro zu finden sind. Die folgende Einführung der beiden Beispielsysteme soll dabei aufzeigen, in welcher Weise Verarbeitungssoftware für Dokumente heute das Modell der strukturierten Dokumente erfüllt.

Die Diskussion soll darüber hinaus auch dafür sensibilisieren, bei der Erstellung von neuen eigenen Dokumenten die angebotenen Mechanismen eines Dokumentensystems zur Unterstützung des Modells der strukturierten Dokumente möglichst weitgehend auszunutzen. Dies ist im Interesse eines effizienteren Document Management und einer flexibleren Weiterverwendung der daraus resultierenden Dokument- und Datensammlungen.

Umsetzung des Modells mit LaTeX

Das Beispiel des Konzeptes zum Computer-Propädeutikum befindet sich im LaTeX-Format im Anhang unter 1.6.5. Der textliche Inhalt ist gleich dem in der Anzeige unter 1.6.1 und dem im XML-Format unter 1.6.4. Es ist auch eine gewisse Übereinstimmung bei den markierten Strukturen festzustellen. Die in der LaTeX-Version explizit markierten logischen Elemente sind die Angaben über die Autorinnen `\author{...}`, die Hauptüberschrift `\title{...}`, die Abschnittüberschriften `\section{...}`, die Liste `\begin{itemize} ... \end{itemize}`, die Aufzählungspunkte `\item`, das Dokument als Ganzes `\begin{document} ... \end{document}`, das Datum `\Datum` und die Versionsnummer `\Version{...}`.

Die syntaktischen Formen des Markup in LaTeX sind vielfältiger als in XML, auch wenn es gewisse Ähnlichkeiten gibt. Eine direkte Entsprechung zu XMLs Konstrukt `<XXX>...</XXX>`, um ein logisches Element XXX zu markieren, ist

`\begin{XXX} ... \end{XXX}`, aber es existieren auch die Formen `\XXX{...}` und sogar `\XXX ...` ohne explizite Markierung des Elementes.

Die Funktionsbedeutung ergibt sich aus der Anfangsmarkierung eines Elements zusammen mit der Metainformation, ob dieses nächste Element im Element `XXX` enthalten sein kann. Einen Sonderfall stellt die implizite Abgrenzung von Absätzen durch Leerzeilen dar.

Das zur Verfügung stehende Repertoire von logischen Elementen ist aus der zugrunde liegenden Dokumentenklasse `\documentclass{...}` und den hinzugeladenen Paketen `\usepackage[...]{...}` ersichtlich. Hier werden externe Dateien referenziert, die systemweit zur Verfügung stehen können. Es werden die Sprachelemente definiert, die im Dokument vorkommen. Am Anfang der LaTeX-Version befinden sich außerdem auch intern definierte Elemente, nämlich `\Datum` und `\Version`.

Bei LaTeX ist also vorgesehen, dass die Autoren ihr eigenes Vokabular von Elementen definieren und dass eine Gruppe von Autoren ein gemeinsames Vokabular benutzt. Dieses wird dann nur einmal definiert und kann von verschiedenen Dokumenten aus referenziert werden. Das Vokabular selbst ist formal definiert und weitere syntaktische Einschränkungen sind implizit.

Mit LaTeX wird somit die erste Anforderung des Modells der strukturierten Dokumente erfüllt: Der Inhalt und die logische Struktur eines Dokumentes werden explizit repräsentiert. Die Strukturelemente können je nach Anwendungsfall frei definiert werden. Ansatzweise ist es auch möglich, die Strukturelemente formal festzulegen.

Wie sieht es mit der zweiten Anforderung aus, dass Vorgaben für die Formatierung in separaten und auswechselbaren Stylesheets vorzuhalten sind? Hier kommen wieder die Konstrukte `\documentclass{...}` und `\usepackage[...]{...}` ins Spiel, die schon bei der Definition der Strukturelemente herangezogen worden sind. Die hier referenzierten externen Dateien erfüllen demnach eine Doppelfunktion: Die Festlegung darauf, welche Strukturelemente im Dokument vorkommen können und die Vorgabe, wie die Strukturelemente formatiert werden sollen.

Sie können diese Doppelfunktion nachvollziehen, wenn Sie sich die interne Definition von `\Datum` und `\Version` anschauen. Die Definition von `\Version` besagt beispielsweise, dass das nach der Markierung in geschweifte Klammern eingeschlossene Argument, also die Versionsnummer, fett gesetzt werden soll: `fett=boldface` und wird abgekürzt zu `bf`. Die Definition von `\Datum` impliziert, dass die Systemfunktion `\today` aufgerufen wird, die dann das aktuelle Datum einsetzt.

Die Forderung nach Stylesheets ist erfüllt. Im Falle von LaTeX sind diese extern und somit auswechselbar, oder intern und an das Dokument gebunden.

Die Schwachstellen, die es in der Umsetzung des Modells der strukturierten Dokumente bei LaTeX gibt, werden bei einer genaueren Analyse offensichtlich. Die Vorgehensweise, mit der bei LaTeX die logische Markierung mit der Formatvorgabe verbunden wird, und wie das Format selber definiert ist weist einige Schwächen auf. In LaTeX wird die Markierung als Kommando einer so genannten Makroprogrammiersprache behandelt und die Formatdefinition als Programm in dieser Programmiersprache. Mit dieser Makroprogrammierung kann mehr erreicht werden als nur die Formatierung festzulegen. So speichern die Programme für `\author` und `\title` ihre Argumente nur intern. Erst das Kommando `\maketitle` im Eingabedokument, das gar keine offensichtliche logische Funktion hat, platziert die entsprechenden Angaben auf der Seite und legt ihre grafische Erscheinungsform fest. Im Extremfall können Makroprogramme sogar andere Kommandos umdefinieren. Beispielsweise könnten die Leerzeilen, die Absatzgrenzen markieren, so umdefiniert werden, dass das jeweils erste Wort eines Absatzes als Schlüsselwort fett gedruckt wird. Ebenso bewirkt das Klammerpaar `\begin{verbatim}` und `\end{verbatim}`, dass dazwischen eingeschlossenes Markup im Klartext ausgegeben und nicht interpretiert wird.

In einer reinen Markupsprache werden nur die Markierungen bereitgestellt, die im Dokument verwendet werden. Bei LaTeX wird die Funktion von Markupsprache, Formatdefinition und Programmierung untrennbar aneinander gekoppelt. Dies muss nicht unbedingt problematisch sein – allerdings widerspricht es dem allgemeinen und wohlbegründeten Informatik-Prinzip, der Trennung von konzeptuellen Dimensionen. In der Praxis bedeutet das, dass eine Person, die ein Stylefile für LaTeX erstellen möchte, über ein ganzes Spektrum von Fertigkeiten verfügen muss. Die gewünschte logische Struktur muss geplant und organisiert werden, das grafische Design muss entworfen und die so entwickelten Vorstellungen sollten dann auch durch Computerprogramme realisiert werden. Die Einstiegshürde für einen kreativen Gebrauch von LaTeX ist also hoch. Im Gegenzug ist LaTeX durch seine Programmierbarkeit ein mächtiges und in seiner Funktionalität bei Bedarf beträchtlich erweiterbares System.

LaTeX ist ursprünglich als Textsatzsystem entwickelt worden. Die Funktionen beschränken sich also im Wesentlichen auf den Bereich des Document Management. Da jedoch bei LaTeX die Technik des eingebetteten Markups zur Repräsentation von Strukturelementen zur Verfügung steht, können mithilfe von weiteren Programmen sowie entsprechenden Schnittstellen die LaTeX-

Quelldokumente als Text eingelesen und automatisch weiterverarbeitet werden.

Die LaTeX-Software ist frei verfügbar. Die deutsche TeX Users Group DANTE e.V. stellt LaTeX für verschiedene Plattformen unter http://www.dante.de zur Verfügung.

Zusammenfassend lässt sich also feststellen, dass LaTeX das Modell der strukturierten Dokumente weitgehend erfüllt. Die Markupsprache besteht jedoch nicht aus einfach deklarativen Markierungen, sondern ist eine Programmiersprache ist.

Umsetzung mit Microsoft Word

Das System Microsoft Word, das nun näher betrachtet wird, steht LaTeX in seinen grundlegenden Eigenschaften fast diametral entgegen. Einfachheit der Benutzung als Bürosystem ist oberstes Ziel, für das Abstriche bei der Mächtigkeit, Flexibilität und Erweiterbarkeit in Kauf genommen werden.

What you see is what you get

Ein Word-Dokument wird in einer bereits formatierten Sicht editiert. Markup wird in der Grundeinstellung nicht angezeigt. Die zugeordnete Formatierung wird gleich umgesetzt und am Bildschirm angezeigt. Das *WYSIWYG* Prinzip ist somit so gut wie perfekt umgesetzt wie die Abbildung unter 1.6.1 zeigt.

Der Anhang enthält bei Abb.: 1.11 ein Bildschirmfoto des Beispiel-Dokuments mit struktureller Unterteilung. Wer ein Word-System installiert hat, kann vielleicht nachvollziehen, welche Strukturelemente im vorliegenden Beispiel verwendet wurden: Autorin, Titel, Hauptüberschrift, Nebenüberschrift, Datum, Version, Abschnittüberschrift, Absatz und Aufzählungspunkt.

Autoren können mit Word Zeichenketten oder ganzen Absätzen eine Formatierung zuzuweisen. Die unterschiedlichen Formatierungen werden als Satz von Formatier-Parametern mit den jeweiligen Werten unter einem individuellen Bezeichner abgespeichert.

Vom Modell der strukturierten Dokumente aus gesehen entsprechen die Namen der Formatvorlagen dem Vokabular an Strukturelementen, die Formatvorlage selber entspricht einer Regel im Stylesheet und die Zuweisungen der Formatvorlagen an die Absätze und Zeichenketten im Dokument entsprechen der Markierung mit logischen Strukturelementen. Mit Word wird damit die explizite Repräsentation von Inhalt und logischer Struktur eines Dokumentes ermöglicht. Die Strukturelemente können je nach Anwendungsfall frei definiert werden und ansatzweise ist es möglich, die Strukturelemente formal festzulegen.

Word und Stylesheets

Darüber hinaus ist es in Word möglich in so genannten *Document Templates*, die als Stylesheets fungieren, Sammlungen von Formatvorlagen anzulegen, sie zu verwalten und sie Dokumenten zuzuordnen. Stylesheets in Word sind auf diese Weise auswechselbar.

Mit Word wird das Modell der strukturierten Dokumente nicht in zufrieden stellendem Maße erfüllt. Der Grund liegt darin, dass außer für Absätze und darin enthaltene Zeichenketten keine logischen Markierungen im Dokument angebracht werden können. Beispielsweise können die einzelnen Aufzählungspunkte der Liste nicht zu einem logischen Element Liste zusammengefasst werden. Die Abschnittüberschrift mit den zugehörigen Absätzen kann nicht zu einem Abschnitt gebündelt und entsprechend markiert werden. Die Schachtelungstiefe von Strukturelementen in Word ist limitiert.

Selbsttestaufgabe 1.2

Zählen Sie einige Beispiele auf, in denen Sie eine tiefere Schachtelung von Strukturelementen für sinnvoll halten als Word sie zulässt.

Bei Word wird kein eingebettetes Markup für die Repräsentation von Strukturelementen im Dokument verwendet. Word-Dokumente liegen vielmehr in einem proprietären, nicht dokumentierten und sich häufig ändernden Format vor. Soll ein Word-Dokument automatisch weiterverarbeitet werden und dabei auch die Struktur der Formatvorlage ausgewertet werden, gibt es zwei Möglichkeiten:

1. Microsoft hat eine stabile Zugangsschnittstelle - die *API* - für Fremdanwendungen definiert. So können die Word-Dokumente programmgesteuert verarbeitet werden. Diese API ist von einigen Programmiersprachen wie Visual Basic oder C++ aus nutzbar.

2. Das Word- Dokument wird in das stabile und dokumentierte Format *RTF* exportiert. Anschließend kann auf die RTF-Daten zugegriffen werden. Der Haken bei diesem Ansatz ist allerdings, dass die Abbildung der Strukturelemente aus Word nach RTF instabil und mit Informationsverlust behaftet ist. Bei Word sind also erheblich höhere Hürden damit verbunden, die Dokumentdaten einer Fremdanwendung zugänglich zu machen als bei LaTeX.

Nach ähnlichen Prinzipien wie Word operieren *Word Perfect*, *FrameMaker*, *Microstar Office* und *Quark Express*.

Selbsttestaufgabe 1.3

Welches Textsystem benutzen Sie im Alltag oder im Beruf? Analysieren Sie, inwieweit Ihr Textsystem das Modell der strukturierten Dokumente unterstützt.

Selbsttestaufgabe 1.4

Nehmen Sie sich ein Dokument vor, das Sie vor kurzem geschrieben haben. Analysieren Sie, welche Strukturelemente darin vorkommen. Nehmen Sie eine logische Markierung dieses Dokuments vor, zunächst auf Papier und dann mit Ihrem Textsystem. Definieren Sie zwei verschiedene Formate für Ihr Dokument und erstellen Sie entsprechende Stylesheets. Erstellen Sie ein zweites Dokument vom selben Typ wie Ihr Ausgangsdokument mit den gleichen Strukturelementen und mit anderem Inhalt. Wenden Sie Ihre beiden Stylesheets auf das neue Dokument an.

1.4 Zeichenkodierungen

Glatter Text

Der Inhalt ist ein grundlegender Bestandteil eines strukturierten Dokuments. Die Repräsentation im Computer, also die kodierte Datei, spielt somit auch eine wichtige Rolle. Es wird dabei davon ausgegangen, dass der Inhalt aus „glattem" Text besteht. Dies ist eine Folge von elementaren Texteinheiten wie Buchstaben, Ziffern, Interpunktionszeichen und anderen Zeichen. Diese Zeichen müssen letztendlich in Bits und Bytes kodiert werden. Eine Zeichenkodierung gibt dazu eine Vorschrift an.

1.4.1 Übersicht

Die bekannteste Zeichenkodierung ist *US-ASCII*, standardisiert als *ANSI X3.4* im Jahre 1968. Sie ist eine die Variante zu *ISO 646-US* und somit von *ISO 646* aus dem Jahre 1972. Der Zeichensatz von US-ASCII bei Abb.: 1.3 reicht lediglich um lateinische und US-amerikanische Texte sowie Texte in einigen wenigen weiteren Sprachen zu kodieren.

Für das Deutsche, das Dänische und vierzehn weitere Sprachen sieht ISO 646 deshalb nationale Varianten vor, welche die US-ASCII-Zeichen „#" (Hash), „$" (Dollar), „@" (At), „[" (eckige Klammer auf), „\" (Backslash), „]" (eckige Klammer zu), „^" (Caret), „`" (Grave), „{" (geschweifte Klammer auf), „|" (Strich), „}" (geschweifte Klammer zu) und „~" (Tilde) durch nationale Zeichen ersetzen. Die deutsche Variante ISO 646-DE bei Abb.: 1.4 nimmt beispielsweise die in Tab.: 1.1 aufgeführten Ersetzungen vor.

ISO 646-US	ISO 646-DE
[	Ä
\	Ö

]	Ü
{	ä
|	ö
}	ü
~	ß
@	§

Tab.: 1.1 Die Ersetzungen der deutschen Variante von ISO 646

Aus diesem Grund erscheint ein Absender „Technische Universität München, Arcisstraße 21" gelegentlich auch heute noch in amerikanischen E-Mail-Programmen in der Form „Technische Universit{t M}nchen, Arcisstra~e 21". Die ursprünglich mit ISO 646-DE angelegte Kodierung der Adresse wird mithilfe von ISO 646-US ausgegeben.

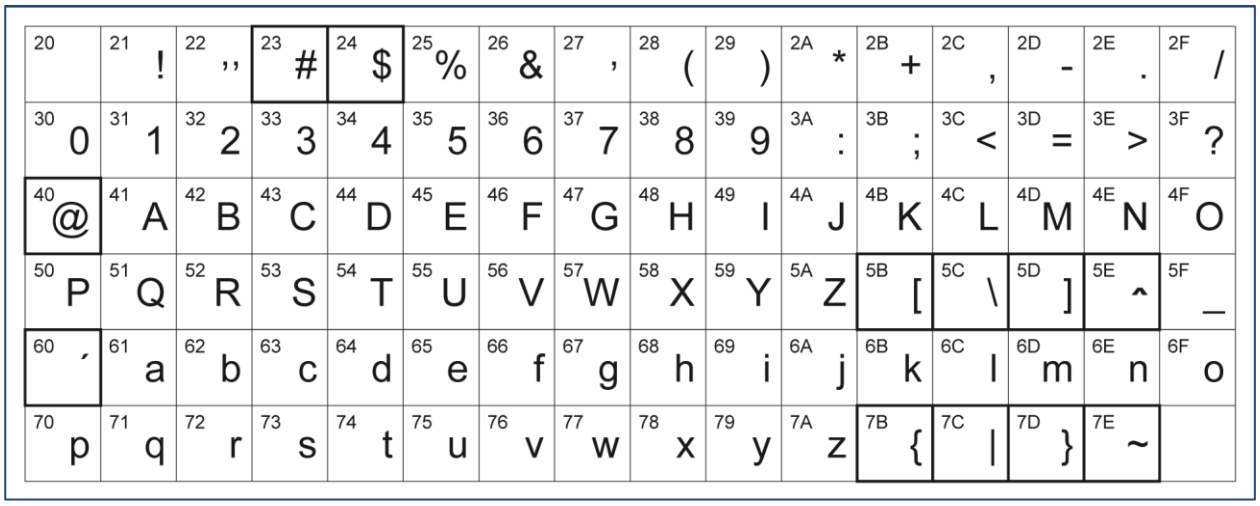


Abb.: 1.3: Der Zeichensatz ISO 646-US (US-ASCII)

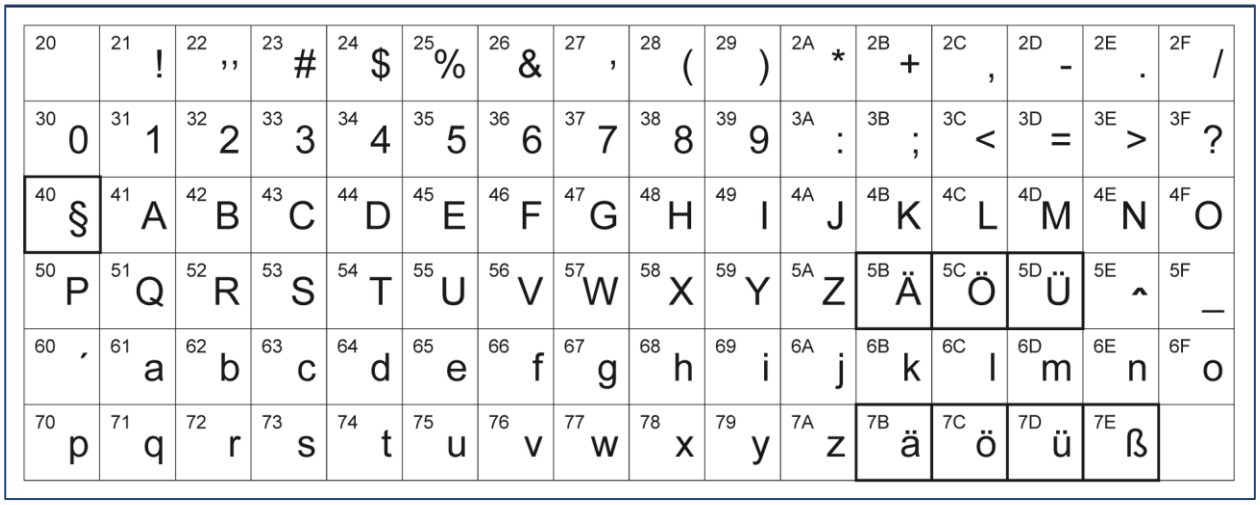
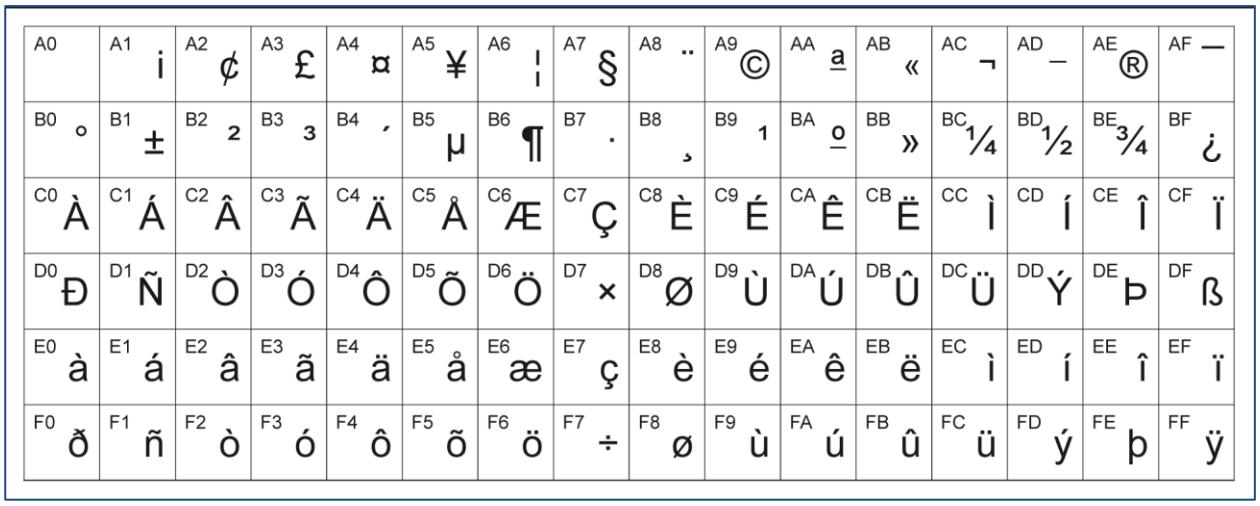


Abb.: 1.4: Die deutsche Variante ISO 646-DE von ISO 646



A0	A1 ¡	A2 ¢	A3 £	A4 ¤	A5 ¥	A6 ¦	A7 §	A8 ¨	A9 ©	AA ª	AB «	AC ¬	AD	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 ´	B5 µ	B6 ¶	B7 ·	B8 ¸	B9 ¹	BA º	BB »	BC ¼	BD ½	BE ¾	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

Abb.: 1.5: Der Zeichensatz ISO 8859-1 (ISO Latin-1)

ECMA

Mitte der achtziger Jahre erweiterte die *E*uropean *C*omputer *M*anufacturer's *A*ssociation den Zeichensatz US-ASCII zu einer Familie von Zeichensätzen, mit denen die alphabetischen Schriftsysteme kodiert werden können. Die inzwischen von der ISO unter dem Namen *ISO 8859*[i][1] kodierte Zeichensatzfamilie besteht aus den Zeichensätzen ISO 8859-1 bis ISO 8859-15. Jeder Zeichensatz ISO 8859-X umfasst die 128 US-ASCII-Zeichen und ergänzt sie um weitere 128 Zeichen. Für die meisten westeuropäischen Sprachen wie das Deutsche, Dänische oder Französische, ist ISO 8859-1, auch ISO Latin-1 (siehe Abb.: 1.5: Der Zeichensatz ISO 8859-1 (ISO Latin-1)) genannt ISO 8859-7 deckt das griechische Alphabet ab und ISO-8859-15 ersetzt im Wesentlichen das internationale Währungssymbol mit dem Eurozeichen. Die so genannte Code Page 1252[2] von Microsoft Windows ist in weiten Teilen identisch mit ISO 8859-1, ersetzt aber einige Kontrollzeichen von ISO 8859-1 durch druckbare Zeichen. So ist u. a. das Eurozeichen € vorhanden.

Anfang der neunziger Jahre kamen *Unicode* und *ISO/IEC 10646* heraus. Das Ziel dieser beiden in einem Sinne, den wir noch genauer kennen lernen werden, äquivalenten Zeichenkodierungen ist Universalität, also Texte aus sämtlichen Sprachen der Welt eindeutig zu kodieren. Die aktuellen Versionen Unicode 3.0 und ISO/IEC 10646-1:2000 kodieren 49.194 Zeichen, die alle modernen und viele klassische Sprachen abdecken. Unicode und ISO/IEC 10646 werden laufend um weitere historisch bedeutsame Zeichen und Sprachen ergänzt.

1.4.2 Was gehört zu einer Zeichenkodierung?

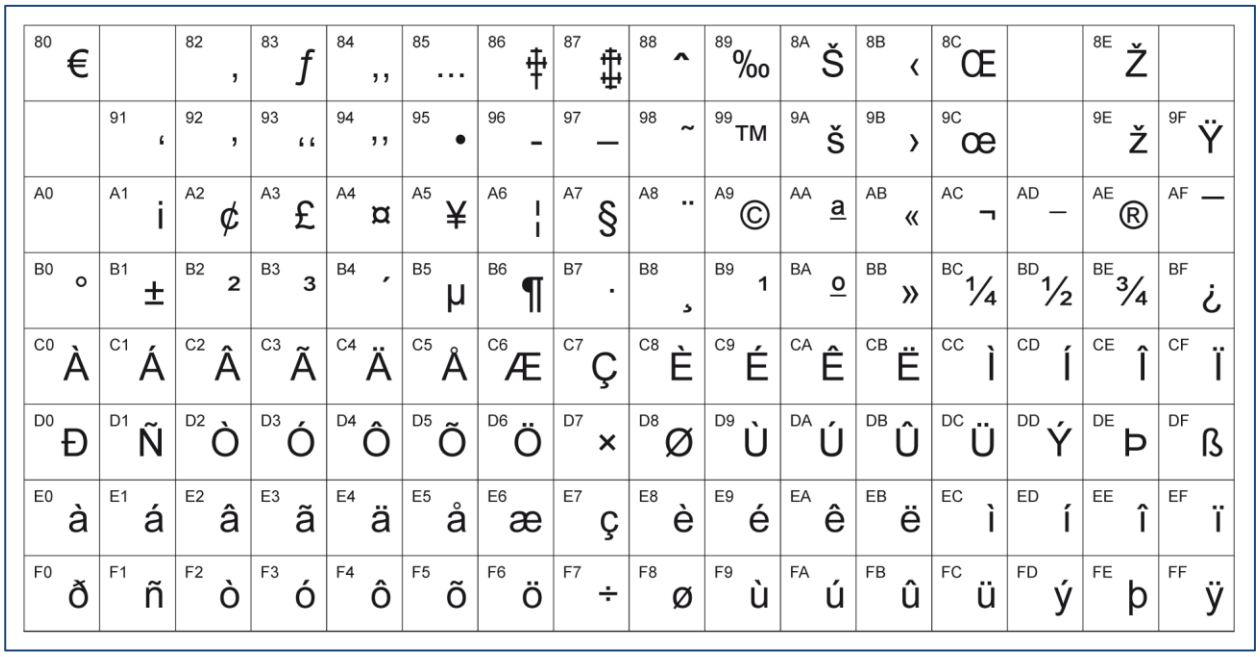
Die Kodierungen ISO 8859 und Unicode sind von zentraler Bedeutung für die Internetformate HTML und XML. Ohne internationale Standards für die

---

[1] http://www.cs.tut.fi/~jkorpela/iso8859/

[2] http://msdn.microsoft.com/en-us/goglobal/cc305145.aspx

Kodierung universeller Zeichensätze wäre der grenzüberschreitende Dokumentenaustausch im Internet und im Web zum Scheitern verurteilt.



80 €		82 ‚	83 ƒ	84 „	85 …	86 †	87 ‡	88 ˆ	89 ‰	8A Š	8B ‹	8C Œ		8E Ž	
	91 ‘	92 ’	93 “	94 ”	95 •	96 –	97 —	98 ˜	99 ™	9A š	9B ›	9C œ		9E ž	9F Ÿ
A0	A1 ¡	A2 ¢	A3 £	A4 ¤	A5 ¥	A6 ¦	A7 §	A8 ¨	A9 ©	AA ª	AB «	AC ¬	AD	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 ´	B5 µ	B6 ¶	B7 ·	B8 ¸	B9 ¹	BA º	BB »	BC ¼	BD ½	BE ¾	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

Abb.: 1.6 Der Windows-Zeichensatz 1252

Im Folgenden werden diese Kodierungen genauer vorgestellt, und zwar im Rahmen eines abstrakten Kodierungsmodells für Zeichen [3, 6, 13].

Das Kodierungsmodell stellt einen konzeptuellen Rahmen dar, innerhalb dessen man die Kodierung von potentiell einigen Milliarden Zeichen in Bitmuster strukturieren und so besser verstehen kann.

Die einzelnen Bestandteile des Kodierungsmodells sind:

- ein abstrakter Zeichensatz
- eine Codetabelle
- ein Kodierungsformat
- ein Kodierungsschema
- eine Übertragungssyntax, die über so genannten „glatten Text" hinaus geht

Die einzelnen Bestandteile des Kodierungsmodells entsprechen zunehmend konkreten Repräsentationen von Zeichen, von einem abstrakten Begriff hin zu Bitmustern. Zwischen einer abstrakten Repräsentationsebene und der nächstliegenden konkreteren Repräsentationsebene besteht eine Abbil-

dungsvorschrift im mathematischen Sinne. Um über mehrere Ebenen hinweg von einer Zeichenposition in einer Codetabelle direkt zu seiner Kodierung in einem Kodierungsschema zu gelangen, können die verschiedenen Abbildungsvorschriften konkateniert werden. Daraus entsteht eine so genannte Zeichenkarte.

1.4.3 Abstrakter Zeichensatz

Die obere abstrakte Ebene des Kodierungsmodells ist der abstrakte Zeichensatz. Dieser besteht aus einer Menge so genannter abstrakter Zeichen. Ein abstraktes Zeichen ist eine Informationseinheit, die zur Repräsentation, Organisation oder Kontrolle von Text dient. Es handelt sich hierbei also um Buchstaben, Ziffern, Interpunktionszeichen, Akzente, grafische Symbole, ideografische Zeichen, Leerzeichen, Tabulatoren, Zeilenweiterschaltung und -vorschub und Kontrollkodes für Auswahl-Markierungen. Der Zeichensatz ISO 646-US (US-ASCII) beispielsweise besteht aus 33 Kontrollzeichen und 95 druckbaren Zeichen.

Ein abstrakter Zeichensatz beinhaltet jedoch noch keine Ordnung oder Nummerierung der im Zeichensatz enthaltenen Zeichen. Dieser Aspekt kommt in der nächsten Stufe des Kodierungsmodells in Form einer Codetabelle hinzu.

Abstrakte Zeichensätze wurden in der Regel festgelegt, um alle Texte einer bestimmten Sprache oder Sprachfamilie angemessen zu kodieren. Der Anspruch, alle lebenden und alle historisch bedeutsamen Sprachen mit einem einzigen Zeichensatz kodieren zu können, führte zu den (identischen) Zeichensätzen von Unicode und ISO/IEC 10646, dem so genannten *U*niversal *C*haracter *S*et kurz UCS genannt.

Zur visuellen Präsentation von Text dienen abstrakte graphische Einheiten, die *Glyphen* genannt werden. Glyphen entsprechen auf der Präsentationsebene den Zeichen auf der Kodierungsebene. Die Liste

A, A, **A**, *A*, A, ***A***

List.: 1.1 Typographie und Auswirkungen

enthält grafische Ausprägungen der Glyphen für den Buchstaben „A", so genannte *Glyphenbilder*. Nach Design und anderen Charakteristika zusammengehörige Glyphenbilder bilden einen Font.

Die Beziehung zwischen abstrakten Zeichen und Glyphen ist komplex. In Tab.: 1.2 gibt es eine beispielhafte Gegenüberstellung. Im einfachsten Fall präsentiert sich ein einzelnes Zeichen als genau eine Glyphe. Einzelne Glyphen können jedoch auch ganze Folgen von abstrakten Zeichen darstellen, wie etwa bei den Ligaturen „fi", „fl", „ff" oder „ffl". Ein einzelnes abstraktes Zeichen mit Akzent,

wie das Zeichen „ö", kann durch zwei Glyphen dargestellt sein, nämlich die Glyphe für das Basiszeichen „o" und die darüber positionierte Glyphe für den Akzent. Im Tamilischen gibt es Buchstaben, die durch ein Paar von Glyphen repräsentiert werden. Dieses Glyphenpaar umrahmt dann in der formatierten Sicht einen weiteren Glyphen, der den Nachbarbuchstaben im Text darstellt. Im Arabischen schließlich gibt es *kontextuelle Formen*: ein Zeichen wird je nach dem Kontext seiner Nachbarzeichen durch Glyphen ganz verschiedener grafischer Form dargestellt. Ähnliche Phänomene sind aus der mittelalterlichen Satztechnik bekannt.

Gutenberg verwendete bereits für ein- und denselben Buchstaben verschieden breite Glyphen. Er verfügte auch bereits über abkürzende Symbole für Wörter oder Silben, um einen gleichmäßigen rechten Rand zu erzeugen. Hier gibt also die Formatierung den Kontext für die Wahl einer bzw. mehrerer Glyphen.

In vielen Fällen sind die elementaren Texteinheiten, die als abstrakte Zeichen Eingang in einen Zeichensatz finden sollen, nicht offensichtlich. Sowohl aus den verschiedenen Sprachen, deren Texte kodierbar sein sollen, als auch aus den verschiedenen Verarbeitungsfunktionen, die auf die Texte Anwendung finden sollen, können Anforderungen und somit Konflikte erwachsen.

Abstrakte Zeichen	Glyphen
A	A
f+i	Fi
f+f+l	Ffl
ö	o+¨
→	oder →

Tab.: 1.2 Das Verhältnis von abstrakten Zeichen und Glyphen

Im Schwedischen beispielsweise sind „ä" und „ö" Buchstaben für sich, die hinter „z" im Alphabet angeordnet sind. Im Französischen dagegen gibt die *Diaräse* über einem Vokal lediglich einen Hinweis, dass der Vokal separat zu sprechen ist, etwa in „duët". Anstelle eines einzigen abstrakten Zeichens „ë" wären also die beiden abstrakten Zeichen „e" und „¨" mit zwei kombinierbaren Glyphen natürlicher.

Für die Darstellung eines deutschen Textes in gedruckter Form ist die Unterscheidung von Groß- und Kleinbuchstaben im Zeichensatz äußerst bequem. Bei der Nutzung einer Sortierfunktion stellt sie jedoch ein kleines Hindernis dar. Im Spanischen gilt „ll" hinsichtlich des alphabetische Sortieren als

ein einziger Buchstabe, für die Formatierung wird „ll" jedoch als Folge von zwei Buchstaben behandelt.

Die Festlegung eines abstrakten Zeichensatzes erfordert also eine sorgfältige Abwägung von verschiedenen Alternativen, wobei Sprachen, Schriften und Bearbeitungsfunktionen zu berücksichtigen sind. Das zugegebenermaßen sehr allgemein formulierte Ziel bei der Entwicklung des Unicode-Zeichensatzes und der gesamten Unicode-Kodierung war es, die Implementierung nützlicher Verarbeitungsprozesse für Textdaten zu ermöglichen.

Unicode 3.0 formuliert zehn Designprinzipien für den Unicode-Standard. Zwei davon lassen sich bereits mit den Begriffen auf der Ebene des Zeichensatzes formulieren:

- Unicode kodiert Zeichen, nicht Glyphen.
- Unicode kodiert glatten Text, keine Formatinformation.

Glyphen spielen bei der Formatierung von Texten eine Rolle und sind auch bei der automatischen Texterkennung interessant. Für die computerisierte Weiterverarbeitung von Inhalten sind sie naturgemäß nicht besonders wichtig.

1.4.4 Codetabelle

Auf der nächsten Stufe des Kodierungsmodells steht die Codetabelle, die den abstrakten Zeichen im Zeichensatz eine Codeposition zuweist. Eine Codeposition ist immer eine natürliche Zahl, also größer oder gleich 0[1].

Der Code-Raum einer Zeichenkodierung ist immer ein Abschnitt der natürlichen Zahlen, der alle Codepositionen enthält. Der Code-Raum kann jedoch auch Zahlen enthalten, die keine zulässigen Codepositionen sind.

Codepositionen werden üblicherweise in Hexadezimalnotation angegeben, wodurch eine natürliche Beziehung zu Bitmustern hergestellt wird. Diese Beziehung kommt auf der nächsten Stufe des Kodierungsmodells – dem Kodierungsformat – zum Tragen.

Ein Bit dient zur Darstellung oder Speicherung von 2 Zuständen, die durch die Ziffern 1 und 0 repräsentiert werden. Die sechzehn möglichen Sequenzen von vier Bits werden üblicherweise mit den Ziffern 0,...,9 und den Buchstaben A,...,F, auch Hexadezimal-Ziffern genannt, bezeichnet.

---

[1] Für diesen Kurs ist die Menge der natürlichen Zahlen so definiert, dass sie die 0 enthält: Es gilt demzufolge IN entspricht IN^{+0}

Die Hexadezimal-Ziffern 0,...,9 und A,...,F stehen für die Dezimalzahlen von 0 bis 15. Jede Hexadezimalzahl stellt also eine natürliche Zahl dar. Eine Sequenz von Hexadezimal-Ziffern $h_n...h_0$ kann in eine Dezimalzahl umgewandelt werden mit folgender Summationsregel:

$$h_0 + h_1 16_d + h_2 16^2_d + ... + h_n 16^n_d$$

Die Zahl 16 ist hierbei in Dezimalschreibweise – erkennbar durch das Zeichen „$_d$". Die Tabelle Tab.: 1.3 zeigt eine Beziehung zwischen dezimaler, hex- und binärer Schreibweise.

Die Dezimal-Zahlen von 0 bis 255 können somit durch Sequenzen von zwei Hexadezimal-Ziffern dargestellt werden und Zahlen von 0 bis 65.535 durch Sequenzen von vier Hexadezimal-Ziffern und damit durch 16 Bit. Sequenzen mit einer Länge von 16 Bit werden auch *Word* genannt.

Selbsttestaufgabe 1.5

Erstellen Sie eine Additions- und eine Multiplikationstabelle für Hex-Ziffern.

Wie in Abb.: 1.3 gezeigt, umfasst der Code-Raum für US-ASCII 128 Positionen. In hexadezimaler Schreibweise geht der Code-Raum also von 0 bis 7F.

Hex-Ziffer	Bitmuster	Dezimalschreibweise
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Tab.: 1.3 Die sechzehn Hex-Ziffern und ihre Repräsentationen als Bitmuster

Der Code-Raum für ISO 8859-X umfasst die 256_d Positionen von 0 bis FF_h, der für Unicode die 65.536_d Positionen von 0 bis $FFFF_h$ plus die $1.048.576_d$

Positionen von 10000_h bis $10FFFF_h$ und der für ISO/IEC 10646 die $2.147.483.648_d$ Positionen von 0 bis $7FFFFFFF_h$.

Kodierung	Code-Raum	Zahl der Codeposition
US-ASCII	0-7F	128
ISO 8859-X	0-FF	256
Unicode	0-10FFFF	65.536+1.048.576
ISO / IEC 10646	0-7FFFFFFF	2.147.483.648

Tab.: 1.4 Code-Räume für verschiedene Kodierungen

Die Code-Räume können weiter strukturiert sein:

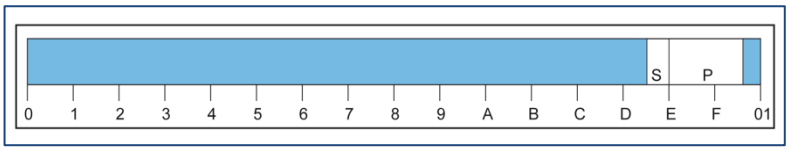


Abb.: 1.7 Unicode-Code-Raum mit Surrogatpositionen und privatem Bereich

Die Positionen von 0 bis $1F_h$ und die Position $7F_h$ von US-ASCII sind Kontrollzwecken vorbehalten, ebenso die Positionen 80_h bis $9F_h$ von ISO 8859-X. Beide Code-Räume können durch ein Byte repräsentiert werden.

Der Unicode-Code-Raum in Abb.: 1.7 enthält Lücken an den 2.048_d *Surrogatpositionen* von $D800_h$ bis $DFFF_h$ und an den Positionen $FFFE_h$ sowie $FFFF_h$. Diese Positionen sind für spezielle Funktionen vorgesehen und es werden ihnen auch künftig keine Unicode-Zeichen zugeordnet werden.

Die 6.400 Unicode-Codepositionen von $E000_h$ bis $F8FF_h$ – also knapp 10 Prozent des Code-Raums von 0 bis $FFFF_h$ – sind für private Zwecke reserviert. Die Positionen sind etwa für Logos, für Icons oder für in speziellen Notationen benötigte Zeichen etwa wie bei Musik und Tanz außerhalb des Unicode-Zeichensatzes freigegeben. Die Codepositionen in diesem Bereich werden auch in späteren Unicode-Ergänzungen für private Verwendung freigehalten.

Selbsttestaufgabe 1.6

Berechnen Sie den prozentualen Anteil der Codepositionen von $E000_h$ bis $F8FF_h$ am Code-Raum von 0 bis $FFFF_h$.

Von den übrigen 57.086_d Unicode-Positionen im Bereich von 0 bis $FFFF_h$ sind 65_d mit Kontrollzeichen belegt, ist 49.194_d Positionen ein druckbares Zeichen zugeordnet und sind 7.827_d Positionen noch frei für Erweiterungen von Unicode. Im nächsten Abschnitt bei der Diskussion von Kodierungsformaten wird

erläutert, wie der Unicode-Code-Raum im Bereich von 0 bis $FFFF_h$ durch zwei Bytes und im Bereich von 10000_h bis $10FFFF_h$ durch vier Bytes repräsentiert wird. Derzeit, d. h. bis zur Version 3.0, enthält die Unicode-Kodetabelle nur Werte im Bereich von 0_h bis $FFFF_h$.

BMP

Der Code-Raum von ISO/IEC 10646 ist konzeptuell unterteilt in 128 Gruppen von 256 Ebenen, wobei jede Ebene 256 Spalten mit je 256 Seiten, also insgesamt 65.536 Zeichen, enthält. Derzeit weist ISO/IEC 10646-1:2000 nur den Codepositionen von 0 bis $FFFF_h$, die in Ebene 0 von Gruppe 0 liegen Werte zu. Diese Ebene heißt auch *B*asic *M*ultilingual *P*lane. Die Kodetabellen von Unicode Version 3.0 und ISO/IEC 10646-1:2000 sind Position für Position identisch. Das Unicode-Konsortium und die ISO haben sich verpflichtet, diese Übereinstimmung auch bei späteren Ergänzungen der Kodetabelle aufrechtzuerhalten; sie haben eine Organisationsform wechselseitiger Liaisonen geschaffen, die dies garantieren kann. Unicode beinhaltet für jedes in seiner Codetabelle kodierte Zeichen die folgenden Daten:

- die Codeposition des Zeichens
- ein typisches Glyphenbild für das Zeichen
- einen Namen
- semantische Information

Die semantische Information ist eine Spezialität bei Unicode. ISO/IEC 10646 stellt diese Information nicht zur Verfügung. Für die Codeposition $00AF_h$ steht in Unicode die folgende Information zur Verfügung:
Das Zeichen hat (1) den Unicode-Namen „Macron", (2) alternative Namen „Overline" und „APL Overline" und ist (3) grafisch ein hochgestellter waagerechter Strich. Das Macron bewirkt (4) einen horizontalen Vorschub (*Spacing Character*) und steht (5) in Beziehung zu weiteren Unicode-Zeichen, u. a. dem Zeichen 0304 „Combining Macron", das genauso aussieht wie das Macron selbst, aber beim Formatieren keinen Vorschub bewirkt (*Nonspacing Character*), sondern sich über das vorangehende Basiszeichen positioniert. Das Makron kann (6) zerlegt werden in das Leerzeichen 0020, gefolgt von dem Combining Macron 0304. Das Macron 00AF ist (7) im Unterschied zum Combining Macron kein Kombinationszeichen sondern ein Basiszeichen.

Das Macron wurde, wie viele andere Unicode-Zeichen, aus Kompatibilitätsgründen in die Codetabelle aufgenommen. Hier wirkt sich das Designziel der Konvertierbarkeit von existierenden Standards nach Unicode und zurück aus. Ein Zeichen, das nur aus Kompatibilitätsgründen in der Codetabelle steht, lässt sich gemäß einer Unicode-Vorgabe immer zerlegen in eine Sequenz aus so genannten Primärzeichen.

Äquivalenz von Zeichen

Für Basiszeichen, die von Kombinationszeichen gefolgt sind, wird bei Unicode ein Äquivalenzbegriff definiert. Kombinationszeichen, die grafisch an unterschiedlichen Stellen am Basiszeichen positioniert werden, können miteinander ausgetauscht werden. Beispielsweise können für die beiden Kombinationszeichen 0307_h „*Combining Dot Above*" und 0323_h „*Combining Dot Below*" die Plätze getauscht werden, sodass die beiden Kodierungen 006F:0307:0323_h und 006F:0323:0307_h für ein „o" mit einem Punkt über sich und einem Punkt unter sich miteinander äquivalent sind.

Auf dem Äquivalenzbegriff für Zeichenfolgen und der Zerlegung von Kompatibilitätszeichen in Folgen von Primärzeichen und der Komposition von Kompatibilitätszeichen aus Folgen von Primärzeichen beruhen auch zwei Normalisierungen von Unicode Zeichenketten, nämlich die *Precomposed* Form und die *Decomposed* Form.

Bei Web-Texten sollten stets Zeichen benutzt werden, die schon so weit wie möglich zusammengesetzt sind. Verbleibende Kombinationszeichen sollten in normierter Reihenfolge angegeben werden. Web-Anwendungen sollten bei Texten, die eingelesen werden so liberal wie möglich programmiert sein. Im Gegenzug dazu sollten sie so konservativ filtern bei Texten, die sie ausgeben. Dies soll Kompatibilitätsprobleme weitgehend vermeiden.

Von den zehn Designprinzipien, die Unicode anführt, lassen sich fünf auf der Ebene der Codetabelle erklären:

Kodierung/Kodierungsformat	Code-Raum/Code-Einheit/Länge
US-ASCII / kanonisch	0-7F / 1Byte / fest (1)
ISO 8859-X / kanonisch	0-FF / 1 Byte / fest (1)
ISO / IEC 10646, Ebene 0 / UCS-2	0-FFFF / 1 Word / fest (1)
ISO / IEC 10646 / UCS-4	0-7FFFFFFF / 2 Word / fest (1)
Unicode 3.0 / UTF8	0-10FFFF / 1 Byte / variabel (1-4)
Unicode 3.0 / UTF16	0-10FFFF / 1 Word / variabel (1-2)

Tab.: 1.5 Kodierungsformate für verschiedene Kodierungen

1. **Unifikation:** Unicode repräsentiert Zeichen, die in verschiedenen Alphabeten vorkommen, aber vom Aussehen her ähnlich sind, nur einmal.
2. **Konvertierbarkeit zwischen etablierten Standards:** Eine eineindeutige Abbildung aus einem etablierten Standard nach Unicode soll ermöglicht werden. Zeichenpositionen aus einem einzelnen international verbreiteten Zeichensatz, die nach dem Designprinzip der Unifikation eigentlich miteinander identifiziert werden müssten, erhalten trotzdem getrennte Unicode-Codepositionen.

3. **Semantik:** Unicode definiert semantische Eigenschaften für Zeichen.
4. **Dynamische Komposition:** Jedes Basiszeichen kann mit beliebig vielen Kombinationszeichen gleichzeitig gepaart werden.
5. **Charakterisierung äquivalenter Kodierungen:** Für die verschiedenen Kodierungen eines Basiszeichens mit Kombinationszeichen oder eines primären und eines aus Kompatibilitätsgründen in den Zeichensatz aufgenommenen Zeichens kann eine normalisierte Kodierung gefunden werden.

1.4.5 Kodierungsformat

Mithilfe des Kodierungsformats für eine Codetabelle werden die Bitrepräsentationen für die Codeposition festgelegt. Dazu gibt es eine Code-Einheit, die in der Regel aus acht oder sechzehn Bits besteht. Durch das Kodierungsformat werden dann die Positionen eines Code-Raums in Sequenzen von Code-Einheiten und somit in Bitmuster abgebildet. Wird jede Codeposition einer Codetabelle auf die gleiche Anzahl von Code-Einheiten abgebildet, sprechen wir von einem Kodierungsformat fester Länge; andernfalls hat das Kodierungsformat variable Länge.

Ein kanonisches Kodierungsformat fester Länge ergibt sich aus der Dualzahldarstellung von Codepositionen. Die Anzahl der jeweils benötigten Code-Einheiten ergibt sich aus der Größe des Code-Raum und der Zahl der Bits in einer Code-Einheit. In Tab.: 1.5 gibt es eine Übersicht über die verschiedenen Kodierungsformate.

Für US-ASCII bzw. seine ISO 646-Varianten sowie für die ISO 8859-XKodetabellen ist das kanonische Kodierungsformat das einzig gebräuchliche. Für die höchstens 256_d vielen Positionen genügen eine Code-Einheit von acht Bits sowie eine Code-Einheit pro Zeichenposition.

Die beiden kanonischen Kodierungsformate für die Code-Räume von 0 bis $FFFF_h$ bzw. von 0 bis $7FFFFFFF_h$ heißen UCS2 und UCS4. Beide Kodierungsformate repräsentieren eine Codeposition mit genau einer Code-Einheit. Im Falle von UCS2 ist die Code-Einheit 16 Bit lang und im Falle von UCS4 ist sie 32 Bits lang.

Prinzipiell kann eine Kodierung mehrere Kodierungsformate definieren und das kanonische Kodierungsformat muss nicht darunter sein. Unicode 3.0 beispielsweise definiert nur zwei Kodierungsformate, nämlich UTF8 und UTF16, wobei UTF für Unicode Transfer Format steht. UTF8 hat eine Code-Einheit von 8 Bits Länge und repräsentiert Codepositionen mit einer bis vier Code-Einheiten.

UTF16 hat eine Code-Einheit von 16 Bits Länge und repräsentiert Codepositionen mit einer bis zwei Code-Einheiten.

Ursprünglich war Unicode für einen Code-Raum von 0 bis $FFFF_h$ ausgelegt, sodass sechzehn Bits genügt hätten, um Codepositionen darzustellen. Tatsächlich hätte dieser Code-Raum ja auch mindestens bis Version 3.0 ausgereicht. Als jedoch klar wurde, dass zumindest für alle historisch interessanten Sprachen und Alphabete der Code-Raum nicht ausreichen würde und weil man mit ISO/IEC 10646 und seinem größeren Code-Raum kompatibel bleiben wollte, wurde das Konzept der Surrogat-Codepositionen eingeführt und der Code-Raum von Unicode bis zur Position $10FFFF_h$ erweitert. Das Kodierungsformat UTF16 benutzt jeweils ein Paar von Surrogatpositionen, um Positionen jenseits von FFFF darzustellen.

UTF16 stellt wie die kanonische Kodierung jede gültige Codeposition im Bereich von 0 bis $FFFF_h$dar. Die ungültigen Codepositionen in dem so genannten hohen Surrogatbereich von $D800_h$ bis $DBFF_h$ und in dem niedrigen Surrogatbereich von DC00 bis DFFF entsprechen den hohen und niedrigen Surrogat-Bitfolgen der Form $110110xxxxxxxxxx_b$ und $110111xxxxxxxxxx_b$, in denen jeweils zehn Bitpositionen für 2^{10} verschiedene Werte frei wählbar sind. Die Gegenrechnungen:

$$DBFF_h + 1 - D800_h = DC00_h - D800_h = 400_h = 4 \times 16^2{}_d = 2^{10}{}_d$$
$$DFFF_h + 1 - DC00_h = E000_h - DC00_h = 400_h = 2^{10}{}_d$$

bestätigen, dass in jedem der beiden Surrogatbereiche 2^{10}, also 1024_d ungültige Codepositionen liegen. Ein Paar von Surrogatpositionen, von denen die erste im hohen und die zweite im niedrigen Bereich liegt, ist somit eindeutig charakterisiert durch zwanzig Bits, die wiederum die 2^{20} Werte von 10000_h bis $10FFFF_h$ repräsentieren können:

$$10FFFF_h + 1 - 10000_h = 110000_h - 10000_h = 100000_h = 16^5{}_d = 2^{20}{}_d$$

Es ist deshalb stimmig, wenn bei UTF16 eine Codeposition P im Bereich von 10000_h bis $10FFFF_h$ durch die kanonische Bitdarstellung der beiden hohen und niedrigen Surrogatpositionen H und L repräsentiert, wobei sich H und L wie folgt berechnen:

$$H = (P - 10000_h) \, DIV \, 400_h + D800_h$$
$$L = (P - 10000_h) \, MOD \, 400_h + DC00_h$$

Einheiten	Darstellungsform	freie Bits	Maximum
1	0xxxxxxx	7	7F
2	110xxxxx 10xxxxxx	11	7FF
3	1110xxxx (10xxxxxx)^2	16	FFFF
4	11110xxx (10xxxxxx)^3	21	1FFFFF
5	111110xx (10xxxxxx)^4	26	3FFFFFF
6	1111110x (10xxxxxx)^5	31	7FFFFFFF

Tab. 1.6: Grunddaten zu Kodierungseinheiten

Umgekehrt bestimmen eine hohe und eine niedrige Surrogatposition H und L eine Unicode-Position P im Bereich von 10000 bis 10FFFF wie folgt:

$$P = (H - D800_h) \cdot 400_h + (L - DC00_h) + 10000_h$$

Selbsttestaufgabe 1.7

Wie wird die Unicode-Position F0000 unter UTF16 dargestellt?

Selbsttestaufgabe 1.8

Welche Unicode-Position wird unter UTF16 durch die beiden Surrogat-Words DAFF und DEFF dargestellt?

Das zweite Unicode-Kodierungsformat, nämlich UTF8, hat die besondere Eigenschaft, die ersten 128 Codepositionen kanonisch mit einem Byte darzustellen. UTF8 ist damit US-ASCII-transparent.

Die Kodierungseinheit von UTF8 ist 8 Bits lang. UTF8 stellt jede Position im Bereich von 0 bis $FFFF_h$ mit ein bis drei Kodierungseinheiten, im Bereich von 0 bis $10FFFF_h$ mit ein bis vier Kodierungseinheiten und im Bereich von 0 bis $7FFFFFFF_h$ mit ein bis sechs Kodierungseinheiten dar.

Die UTF8-Darstellung einer Position mit einer einzigen Kodierungseinheit hat die Form $0xxxxxxx_b$ mit einer führenden 0 und beliebigen Bits, die anstelle des Platzhalters x stehen. Die UTF8-Darstellung einer Position mit n Kodierungseinheiten hat die Form $1^n0x^{7-n}{}_b$ gefolgt von n – 1 Kodierungseinheiten der Form $10xxxxxx_b$. Bei drei Kodierungseinheiten sind also genau sechzehn Bits frei wählbar, sodass mit drei Kodierungseinheiten unter UTF8 Code-Positionen bis FFFF darstellbar sind.

Die mit n Kodierungseinheiten darstellbaren Positionsbereiche lassen sich aus der Tab. 1.6: Grunddaten zu Kodierungseinheiten entnehmen.

Für die Darstellung einer Codeposition unter UTF8 muss immer die kleinste Zahl von Kodierungseinheiten gewählt werden. Eine Codeposition im Bereich von 800_h bis $FFFF_h$ muss also mit drei Kodierungseinheiten und darf nicht –obwohl das technisch möglich wäre – mit vier Kodierungseinheiten dargestellt werden.

Die Umkehrfunktion von UTF8, die aus einer Folge von Code-Einheiten eine Codeposition berechnet, darf jedoch auch zu lange Darstellungen korrekt umrechnen. Die zwei Bytes $C1BF_h$ dürfen also unter UTF8 in die Position $7F_h$ zurückgerechnet werden, obwohl die UTF8-Darstellung der Position $7F_h$ das Byte $7F_h$ ist.

Das Unicode-Designprinzip der Effizienz lässt sich auf Ebene von Kodierungsformaten erläutern. Für jedes der beiden Unicode-Kodierungsformate UTF8 und UTF16 lässt sich in einem Strom von Positionsdarstellungen von jeder Kodierungseinheit aus der Anfang der zugehörigen Positionsdarstellung mit beschränktem Backup finden. Im Falle von UTF16 muss höchstens um eine Kodierungseinheit zurückgegangen werden, im Falle von UTF8 höchstens um drei Positionen.

Selbsttestaufgabe 1.9

Kodieren Sie die ASCII-Zeichenfolge „<?xml" ohne die Anführungsstriche unter UTF8 und UTF16.

Selbsttestaufgabe 1.10

Zeigen Sie, dass unter UTF8 die Kodierungen der Positionen 192 bis 255 alle dasselbe führende Byte haben und zwar $C3_h$.

Selbsttestaufgabe 1.11

Ein Textsystem stellt das Zeichen „ü" als Kombination zweier Glyphen Ã und ¼ dar. Überlegen Sie, in welchem Kodierungsformat der Textstrom vorliegen könnte und welche Annahmen das Textsystem über das Kodierungsformat macht.

1.4.6 Kodierungsschema

Damit Daten über Netzwerke zuverlässig ausgetauscht werden können, müssen sie in eine lineare Folge von Bytes gebracht werden. Dieser Vorgang wird auch *Serialisierung* genannt. Mithilfe eines Kodierungsformats wird der Text als Folge von Kodierungseinheiten dargestellt. Ein Kodierungsschema wird nun dazu genutzt zu einem Kodierungsformat zusätzlich festzulegen, wie die Kodierungseinheiten in Bytefolgen zu serialisieren sind.

Ist die Kodierungseinheit eines Kodierungsformats selbst acht Bits lang, so ist nichts mehr festzulegen und das Kodierungsschema ist mit dem Kodierungsformat identisch. So kann also UTF8 sowohl als Kodierungsformat als auch als Kodierungsschema bezeichnet werden.

Ist die Kodierungseinheit ein Word, wie bei UTF16, so gibt es zwei Möglichkeiten der Serialisierung: Big Endian[1]und Little Endian. Dementsprechend gibt es zu UTF16 zwei Kodierungsschemata: UTF16-BE und UTF16-LE. Analoges gilt für UCS2 und UCS4.

BOM

Mit dem *Byte Order Mark* an Position $FEFF_h$ kann bei einem UTF16-repräsentierten Datenstrom signalisiert werden, welche der beiden Serialisierungen, Big Endian oder Little Endian, vorliegt. Da $FFFE_h$ keine zulässige Codeposition bei Unicode ist, lässt sich schließen, dass das Kodierungsschema UTF16-LE vorliegt. Dies funktioniert jedoch nur dann, wenn als die ersten zwei Bytes FF_h und FE_h gelesen werden. Ein BOM am Anfang eines UTF16-kodierten Datenstroms ist nicht Bestandteil der Daten und wird – abgesehen von seiner Signalwirkung für die Serialisierung – ignoriert. Tritt die Codeposition $FEFF_h$ dagegen an anderer Stelle im Datenstrom auf, so wird sie als das entsprechende Unicode-Zeichen *Zero Width No-Break Space* interpretiert, das dieser Position zugewiesen ist.

Generell darf ein Unicode-Zeichenstrom am Anfang mit dem zusätzlichen Zeichen BOM versehen werden. Anwendungen, die über keine Metainformation über die Natur des Datenstroms verfügen, erhalten so das Signal, dass es sich im Folgenden um Unicode-Daten, die in einem bestimmten Kodierungsschema vorliegen, handelt. Das erste BOM eines Zeichenstroms bildet also keinen Teil der eigentlichen Daten.

Selbsttestaufgabe 1.12 Was ist die Kodierung des BOM FEFF unter dem Kodierungsschema UTF8? Überprüfen Sie, ob das Vorausschicken eines BOM

[1] Das höherwertige Bit kommt zuerst

ausreicht, um zwischen den bisher besprochenen Kodierungsschemata zu differenzieren.

1.4.7 Jenseits von glattem Text – eine Übertragungssyntax

Mit glattem Text, der als Folge von Unicode-Zeichen kodiert ist, kann der reine Inhalt von strukturierten Dokumenten gehandhabt werden. Da auch eingebettetes Markup – wie etwa bei den in Abschnitt 1.3.7 eingeführten Tags der Form `<XXX>` und `</XXX>` – eine Folge von Zeichen ist, scheint es auf den ersten Blick als wäre das Problem, strukturierte Dokumente zu kodieren, bereits vollständig gelöst.

Es ergibt sich jedoch eine zusätzliche Komplikation dadurch, dass der Markuptext vom Inhaltstext syntaktisch voneinander unterscheidbar sein muss. Zur Abgrenzung von Markup und Inhalt werden bestimmte Funktionszeichen eingesetzt, die dann außerhalb ihrer syntaktischen Rolle weder im Markup selbst noch im Inhalt des Klartextes vorkommen dürfen. In XML ist „<" ein solches Funktionszeichen, das den Anfang eines Tags charakterisiert. Das Zeichen „<" darf deswegen im Inhaltstext eines XML-Dokuments nicht vorkommen.

Die klassische Methode Funktionszeichen in glattem Text unterzubringen, ist die Verwendung von so genannten *Escape-Zeichen*. Diese werden nicht selbst als Bestandteil des Textes interpretiert sondern signalisieren die Präsenz eines Zeichens, das eigentlich nicht im Text vorkommen darf. Im Falle von XML bezeichnet die Formel `&#<<Hexziffern>>;` oder als `&<<Dezimalziffern>>;` im Inhaltstext das Unicode- Zeichen an Position `<<Hexziffern>>` bzw. an Position `<<Dezimalziffern>>`. Wir können ein „<" im Inhaltstext als z. B. als < notieren und das zusätzliche Funktionszeichen „&" durch &.

Im Allgemeinen wird glatter Text nicht direkt als Folge von Zeichen auftreten. Vielmehr wird die Zeichenfolge, die einen Text ausmacht, konstruiert aus einer weiteren Folge von so genannten Eingabezeichen, von denen ganze Teilsequenzen einzelne Zeichen des Textes repräsentieren. In der Praxis existiert also eine zweistufige Kodierung von Texten. Zeichen entweder durch sich selbst oder durch spezielle Zeichenfolgen wie `&#<<Hexziffern>>;` zu kodieren, löst insgesamt drei bekannte Probleme:

1 In glattem Text lassen sich Funktionszeichen an Stellen unterbringen, an denen sie nicht im Klartext vorkommen dürfen.

2 Es können Zeichen in einen Text eingefügt werden, die das verwendete Eingabewerkzeug nicht unterstützt. So kann also in ein XML-Dokument

auch mithilfe einer amerikanischen Tastatur etwa der Umlaut „ä" als ä eingegeben werden.

3 Die Paare aus den beiden ASCII-Steuerzeichen *linefeed* bzw. *newline* und *carriage return* können als ein- und dasselbe Zeilenendezeichen interpretiert werden. Auf diese Weise werden die beiden gängigen verschiedenen Konventionen im Text ein Zeilenende zu signalisieren vereinheitlicht. Werkzeuge mit denen die Texte verarbeitet werden, verfügen damit über eine plattformunabhängige Methode zur Zeilen-Nummerierung. LF und CR

Die Technik, Zeichen durch Zeichenfolgen spezieller Syntax zu kodieren, ist Standard in allen Bereichen, in denen Texte erstellt werden. Insbesondere bei der Erstellung von Programmtexten spielt sie eine wichtige Rolle.

Java-Programme etwa dürfen beliebige Unicode-Zeichen der Basic Multilingual Plane enthalten. Dies gilt unter anderem für Namen, Literale für Zeichenketten oder bei Kommentaren. Grundsätzlich kann im Programmtext jedes Zeichen durch sich selbst oder durch eine Sequenz `\u<<xxxx>>` mit einer ungeraden Anzahl von Zeichen „\", einer positiven Anzahl von Zeichen „u" und einer Sequenz `<<xxxx>>` von genau vier Hexziffern dargestellt werden. Die einzige Ausnahme betrifft das Zeichen „\", das nur dann für sich selbst stehen darf, wenn es nicht in einer Sequenz der Form `\u<<xxxx>>` vorkommt.

Der Programm-Code in Java zu dem Klassiker "HelloWorld"

```
public class HelloWorld {
        public static void main(String[] args) {
                    System.out.println("Hello World");
        }
}
```

List.: 1.2 Ein typischer Java Programm-Code

`dürfte also auch in folgender Form erscheinen:`

```
\u0070ublic class HelloWorld {
    public static void main(String[] args) {
        System.out.println("\uuuu0048ello World");
    \\\u007D
}
```

List.: 1.3 Programm-Code mit Escape Zeichen

Die Java-Notation macht es also möglich, einen beliebigen Unicode-Text mit Zeichen im Bereich zwischen 0 und FFFF als einen 7-Bit-ASCII-Text zu kodieren.

1.4.8 Unicode-Werkzeuge

Bei der Diskussion von Werkzeugen zur Bearbeitung von Unicode-Texten gibt es den entscheidenden Unterschied zwischen der Eingabe und der Darstellung von Unicode-Zeichen. Wird ein genuiner Unicode-Editor gewünscht, so ist die Auswahl derzeit noch nicht allzu groß. Eine der wenigen Optionen ist ein Forschungsprototyp von der Duke University namens UniEdit,[1] der eine Vielzahl von Kodierungsschemata und Eingabeformaten unterstützt.

Ansonsten unterstützt jeder beliebige Editor für 7-Bit-ASCII den Unicode. Die so erstellten Texte können als Unicode-Texte in UTF8-Kodierung ausgegeben werden. Damit können auch nur 7-Bit-ASCII-Zeichen in eigenen Texten verwendet werden. Indirekt gibt es eine weitere Kodierungsstufe, über die der Zugriff zu dem vollen Unicode-Zeichenrepertoire gewährleistet ist. Im Falle der Java-Kodierung steht ein Werkzeug zur Verfügung, mit dem die Java-Notation wieder dekodiert werden kann. Sie wird so und in verschiedene Kodierungsformate übersetzt. Das Werkzeug heißt *native2ascii* und ist Bestandteil des Java Software Development Kit.

Selbsttestaufgabe 1.13

Installieren Sie, falls erforderlich, das Werkzeug native2ascii (http://java.sun.com) aus dem Java Software Development Kit. Erstellen Sie eine Textdatei mit Umlauten und anderen Zeichen außerhalb von US-ASCII und übersetzen Sie diese Datei mithilfe von native2ascii nach UTF8.

Selbsttestaufgabe 1.14

Erstellen Sie eine Textdatei, in der Sie in Java-Notation die Zeichen außerhalb des Code-Raumes von 0 bis FF_h referenzieren. Versuchen Sie, diese Datei mit native2ascii nach ISO Latin-1 bzw. Code Page 1252 zu Übersetzen. Was passiert?

1.5 Zusammenfassung und Ausblick

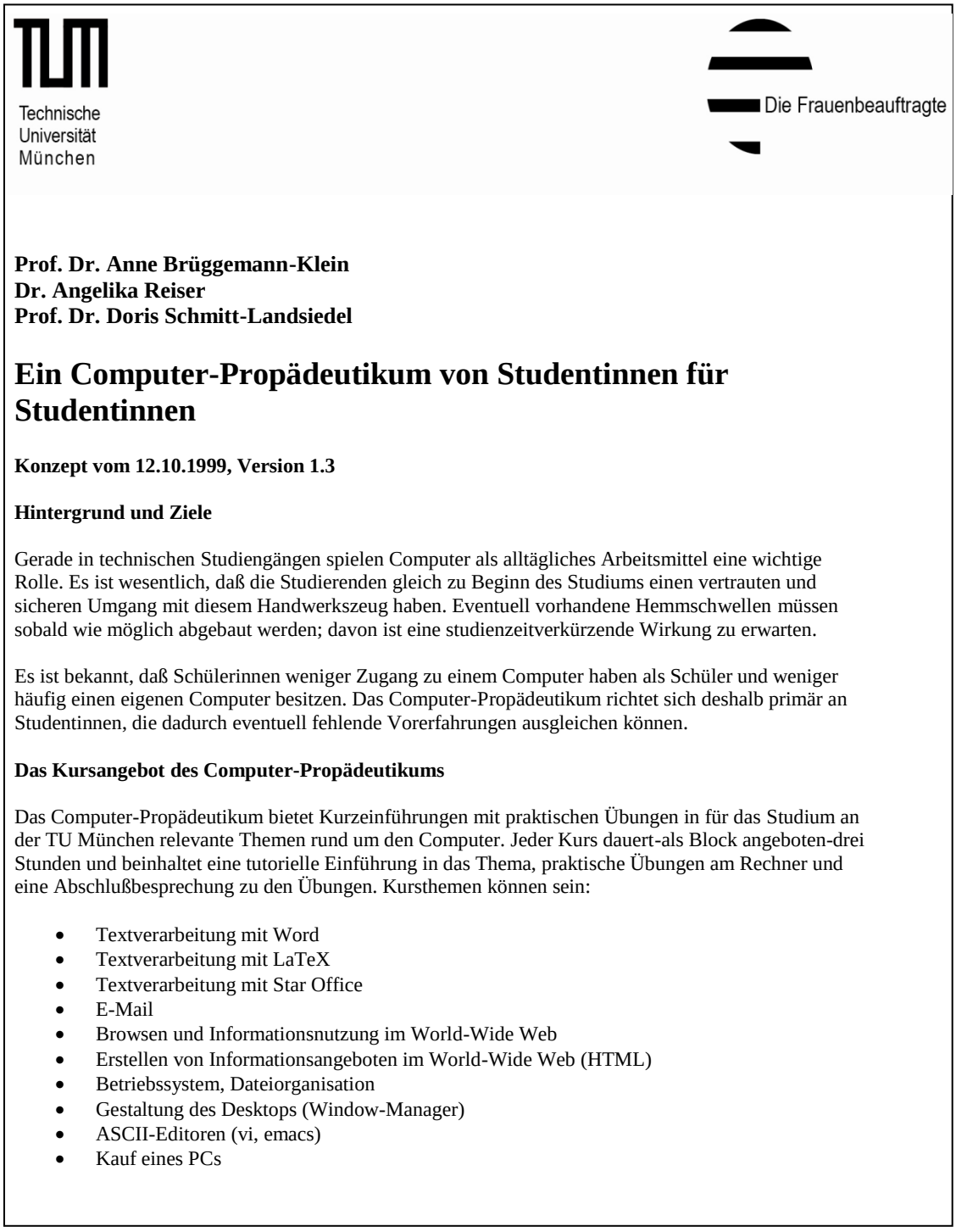
Zwei wichtige Faktoren machen digitale Dokumente zu modernen Informationsträgern, die vielfach verwendbar und der automatischen Verarbeitung zugänglich sind. Die beiden Faktoren sind das Modell der strukturierten Dokumente, dem solche digitalen Dokumente entsprechen sollen, und die Standardisierung der Textkodierung, die Textinhalte austauschbar macht. Beide

[1] http://www.humancomp.org/uniintro.htm

Faktoren wurden in dieser Kurseinheit besprochen. Als dritter Faktor steht die Standardisierung der Strukturinformation noch aus; damit werden die kommenden Kurseinheiten im Zusammenhang mit XML und den verwandten Sprachfamilien zu tun haben.

1.6 Anhang: Formate am Beispiel

1.6.1 Fertig formatiertes Dokument für ein Journal



Technische Universität München

Die Frauenbeauftragte

Prof. Dr. Anne Brüggemann-Klein
Dr. Angelika Reiser
Prof. Dr. Doris Schmitt-Landsiedel

Ein Computer-Propädeutikum von Studentinnen für Studentinnen

Konzept vom 12.10.1999, Version 1.3

Hintergrund und Ziele

Gerade in technischen Studiengängen spielen Computer als alltägliches Arbeitsmittel eine wichtige Rolle. Es ist wesentlich, daß die Studierenden gleich zu Beginn des Studiums einen vertrauten und sicheren Umgang mit diesem Handwerkszeug haben. Eventuell vorhandene Hemmschwellen müssen sobald wie möglich abgebaut werden; davon ist eine studienzeitverkürzende Wirkung zu erwarten.

Es ist bekannt, daß Schülerinnen weniger Zugang zu einem Computer haben als Schüler und weniger häufig einen eigenen Computer besitzen. Das Computer-Propädeutikum richtet sich deshalb primär an Studentinnen, die dadurch eventuell fehlende Vorerfahrungen ausgleichen können.

Das Kursangebot des Computer-Propädeutikums

Das Computer-Propädeutikum bietet Kurzeinführungen mit praktischen Übungen in für das Studium an der TU München relevante Themen rund um den Computer. Jeder Kurs dauert-als Block angeboten-drei Stunden und beinhaltet eine tutorielle Einführung in das Thema, praktische Übungen am Rechner und eine Abschlußbesprechung zu den Übungen. Kursthemen können sein:

- Textverarbeitung mit Word
- Textverarbeitung mit LaTeX
- Textverarbeitung mit Star Office
- E-Mail
- Browsen und Informationsnutzung im World-Wide Web
- Erstellen von Informationsangeboten im World-Wide Web (HTML)
- Betriebssystem, Dateiorganisation
- Gestaltung des Desktops (Window-Manager)
- ASCII-Editoren (vi, emacs)
- Kauf eines PCs

Abb.: 1.8 Erste Seite des formatierten Textes

- Datenverbindungen von zu Hause (analog, ISDN)
- Fakultätsspezifische Informationsinfrastrukturen

Jeder Kurs sollte sich auf ein enges Thema konzentrieren. Die Themenliste ist offen für Ergänzungen. Kriterium ist, daß der Kursinhalt propädeutischer Natur für das Studium an der TU ist und eine Form der Computernutzung zum Inhalt hat. Ein Kurs kann fakultätsübergreifend oder fakultätsspezifisch sein. Die Kurse sind in der Regel auf eine spezielle Rechnerplattform (PC/Windows, Unix) zugeschnitten.

Die Zielgruppe des Computer-Propädeutikums

Das Kursangebot richtet sich an Studentinnen im ersten Semester. Wird ein Angebot von dieser Zielgruppe nicht ausgeschöpft, steht es auch Studentinnen höherer Semester und, in dritter Priorität, männlichen Studierenden offen.

Die Leiterinnen und Leiter des Computer-Propädeutikums

Die Kurse werden vorzugsweise von Studentinnen höherer Semester konzipiert und durchgeführt. Zur Ergänzung des Kursangebots können auch Mitarbeiterinnen und Mitarbeiter oder Studenten höherer Semester Kurse anbieten. Nur von Studierenden angebotene Kurse können im Rahmen des Programms finanziert werden.
Die Kursleiterinnen und -leiter sind selbst für die Organisation von Seminar- und Rechnerräumen und von Zugangsmöglichkeiten zu den Räumen am Wochenende für ihren Kurs verantwortlich. Wir gehen davon aus, daß die Studiendekaninnen und -dekane sie bei der Organisation unterstützen.

Durchführung

Das Computer-Propädeutikum ist eine Initiative der Frauenbeauftragten der TU München und wird vom Frauenbüro (Kerstin Hansen, Anja Quindeau) organisiert. Die TU München finanziert das Programm aus dem Tutorienprogramm zur Verkürzung der Studiendauer. Für von Studierenden angebotene Kurse wird ein Werksvertrag abgeschlossen, der die Konzeption, Vorbereitung und Durchführung eines dreistündigen Kurses beinhaltet. Konzeption und Vorbereitung eines Kurses werden mit 200 DM und jede Durchführung mit 100 DM honoriert. Das Frauenbüro erhält Hilfskraftmittel aus dem Tutorienprogramm für die Organisation des Computer-Propädeutikums. Wir rechnen mit folgenden Kosten: 20 Kurse, jeder zweimal durchgeführt, zu 20 mal 200 DM plus 20 mal 2 mal 100 DM, also 8000 DM, plus 1000 DM für die Organisation im Frauenbüro.

Eine erste Ausschreibung des Programms ist bereits erfolgt. Interessierte Kursleiterinnen und -leiter können ihr Kursangebot bis zum 5.11.1999 abgeben. Bitte benutzen Sie dazu unser Formular. Die ersten Kurse sollen noch im November 1999 stattfinden. Bei Bedarf erfolgt eine zweite Ausschreibung, zu der neue Kursangebote abgegeben werden können, Anfang Dezember 1999. Die Ausschreibung des Programms, die Gestaltung des Kursangebots sowie die Ankündigung der Kurse erfolgt durch das Frauenbüro in Zusammenarbeit mit den Studierendenvertretungen, den Studiendekanen und den Frauenbeauftragten. Die Verträge werden nach Absprache mit dem Frauenbüro von der Hochschulverwaltung abgeschlossen.

Die Kurse sollen zu Randzeiten (Abends, Freitags Nachmittags oder am Wochenende) stattfinden.

Abb.: 1.9 Zweite Seite des formatierten Textes

1.6.2 Unformatiertes Dokument mit reinem Text

Prof. Dr. Anne Brueggemann-Klein Dr. Angelika Reiser Prof. Dr. Doris Schmitt-Landsiedel Ein Computer-Propaedeutikum von Studentinnenfuer Studentinnen Konzept vom 12.10.1999, Version 1.3 Hintergrund und Ziele Gerade in technischen Studiengaengen spielen Computer alsalltaegliches Arbeitsmittel eine wichtige Rolle. Es ist wesentlich, dass die Studierenden gleich zu Beginn des Studiums einen vertrauten und sicheren Umgang mit diesem Handwerkszeug haben. Eventuell vorhandene Hemmschwellen muessen sobald wie moeglich abgebaut werden; davon ist eine studienzeitverkuerzende Wirkung zu erwarten. Es ist bekannt, dass Schuelerinnen weniger Zugang zu einem Computer haben als Schueler und weniger haeufig einen eigenen Computer besitzen. Das Computer-Propaedeutikum richtet sich deshalb primaer an Studentinnen, die dadurch eventuell fehlende Vorerfahrungen ausgleichen koennen. Das Kursangebot des Computer-Propaedeutikums Das Computer-Propaedeutikum bietet Kurzeinfuehrungen mit praktischen Uebungen in fuer das Studium an der TU Muenchen relevante Themen rund um den Computer. Jeder Kurs dauert-als Block angeboten-drei Stunden und beinhaltet eine tutorielle Einfuehrung in das Thema, praktische Uebungen am Rechner und eine Abschlussbesprechung zu den Uebungen. Kursthemen koennen sein: Textverarbeitung mit Word Textverarbeitung mit LaTeX Textverarbeitung mit Star Office E-Mail Browsen und Informationsnutzung im World-Wide Web Erstellen von Informationsangeboten im World-Wide Web (HTML) Betriebssystem, Dateiorganisation Gestaltung des Desktops (Window-Manager) ASCII-Editoren (vi, emacs) Kauf eines PCs Datenverbindungen von zu Hause (analog, ISDN) Fakultaetsspezifische Informationsinfrastrukturen Jeder Kurs sollte sich auf ein enges Thema konzentrieren. Die Themenliste ist offen fuer Ergaenzungen. Kriterium ist, dass der Kursinhalt propaedeutischer Natur fuer das Studium an der TU ist und eine Form der Computernutzung zum Inhalt hat. Ein Kurs kann fakultaetsuebergreifend oder fakultaetsspezifisch sein. Die Kurse sind in der Regel auf eine spezielle Rechnerplattform (PC/Windows, Unix) zugeschnitten. Die Zielgruppe des Computer-Propaedeutikums Das Kursangebot richtet sich an Studentinnen im ersten Semester. Wird ein Angebot von

dieser Zielgruppe nicht ausgeschoepft, steht es auch Studentinnen hoeherer Semester und, in dritter Prioritaet, maennlichen Studierenden offen. Die Leiterinnen und Leiter des Computer-Propaedeutikums Die Kurse werden vorzugsweise von Studentinnen hoeherer Semester konzipiert und durchgefuehrt. Zur Ergaenzung des Kursangebots koennen auch Mitarbeiterinnen und Mitarbeiter oder Studenten hoeherer Semester Kurse anbieten. Nur von Studierenden angebotene Kurse koennen im Rahmen des Programms finanziert werden. Die Kursleiterinnen und -leiter sind selbst fuer die Organisation von Seminar- und Rechnerraeumen und von Zugangsmoeglichkeiten zu den Raeumen am Wochenende fuer ihren Kurs verantwortlich. Wir gehen davon aus, dass die Studiendekaninnen und -dekane sie bei der Organisation unterstuetzen. Durchfuehrung Das Computer-Propaedeutikum ist eine Initiative der Frauenbeauftragten der TU Muenchen und wird vom Frauenbuero (Kerstin Hansen, Anja Quindeau) organisiert. Die TU Muenchen finanziert das Programm aus dem Tutorienprogramm zur Verkuerzung der Studiendauer. Fuer von Studierenden angebotene Kurse wird ein Werksvertrag abgeschlossen, der die Konzeption, Vorbereitung und Durchfuehrung eines dreistuendigen Kurses beinhaltet. Konzeption und Vorbereitung eines Kurses werden mit 200 DM und jede Durchfuehrung mit 100 DM honoriert. Das Frauenbuero erhaelt Hilfskraftmittel aus dem Tutorienprogramm fuer die Organisation des Computer-Propaedeutikums. Wir rechnen mit folgenden Kosten: 20 Kurse, jeder zweimal durchgefuehrt, zu 20 mal 200 DM plus 20 mal 2 mal 100 DM, also 8000 DM, plus 1000 DM fuer die Organisation im Frauenbuero. Eine erste Ausschreibung des Programms ist bereits erfolgt. Interessierte Kursleiterinnen und -leiter koennen ihr Kursangebot bis zum 5.11.1999 abgeben. Bitte benutzen Sie dazu unser Formular. Die ersten Kurse sollen noch im November 1999 stattfinden. Bei Bedarf erfolgt eine zweite Ausschreibung, zu der neue Kursangebote abgegeben werden koennen, Anfang Dezember 1999. Die Ausschreibung des Programms, die Gestaltung des Kursangebots sowie die Ankuendigung der Kurse erfolgt durch das Frauenbuero in Zusammenarbeit mit den Studierendenvertretungen, den Studiendekanen

```
und den Frauenbeauftragten. Die Vertraege werden nach
Absprache mit dem Frauenbuero von der
Hochschulverwaltung abgeschlossen. Die Kurse sollen zu
Randzeiten (Abends, Freitags Nachmittags oder am
Wochenende) stattfinden
```

1.6.3 Dokumentdarstellung mit optischer Gliederung

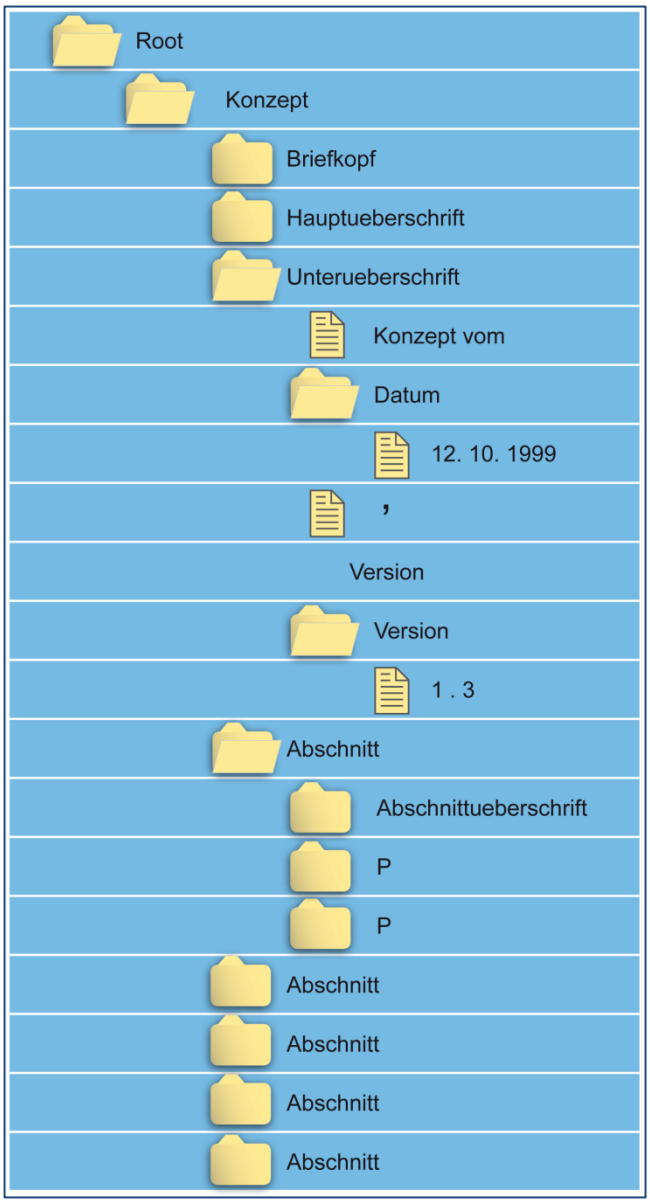


Abb.: 1.10:Dokumentdarstellung mit Ikonen

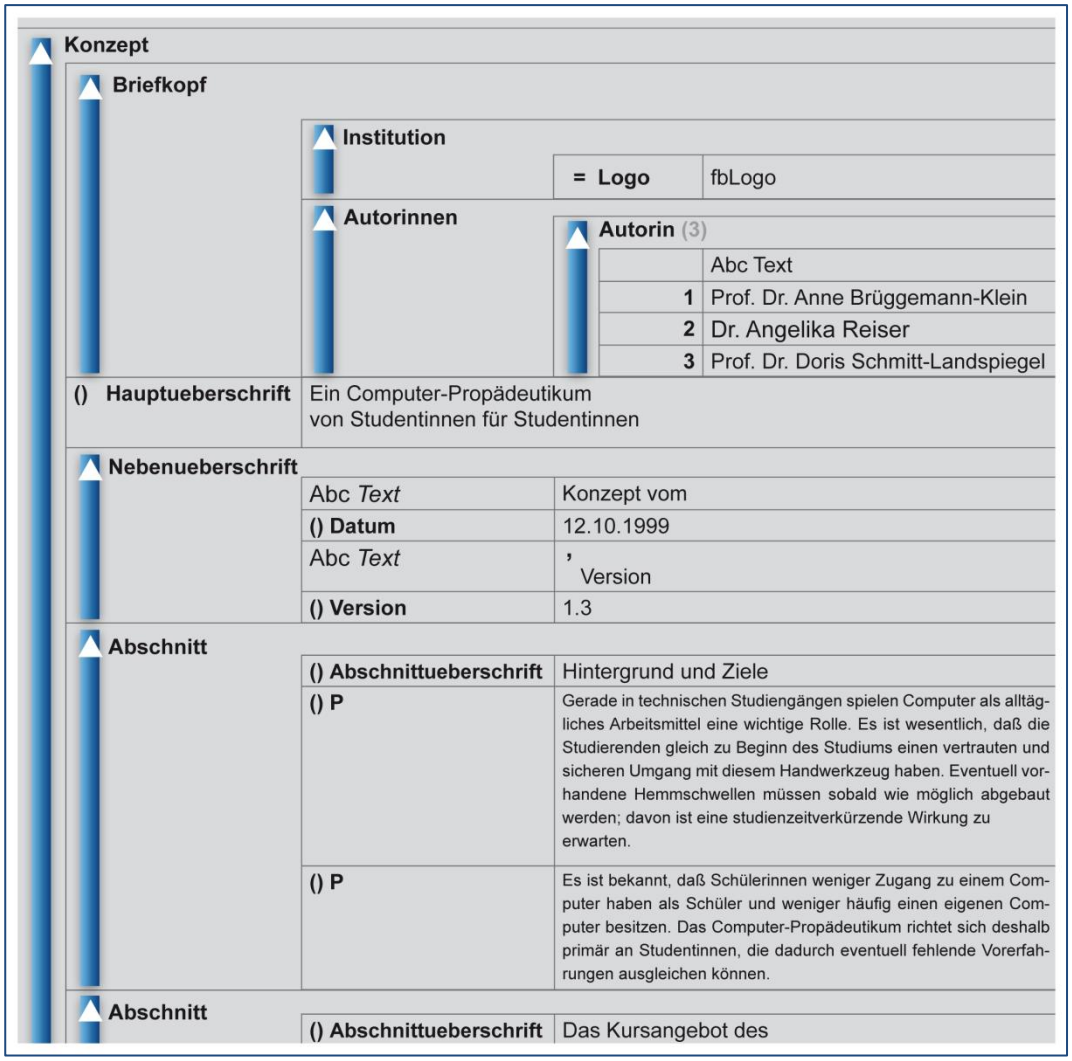


Abb.: 1.11 Dokumentdarstellung mit verschachtelten Boxen

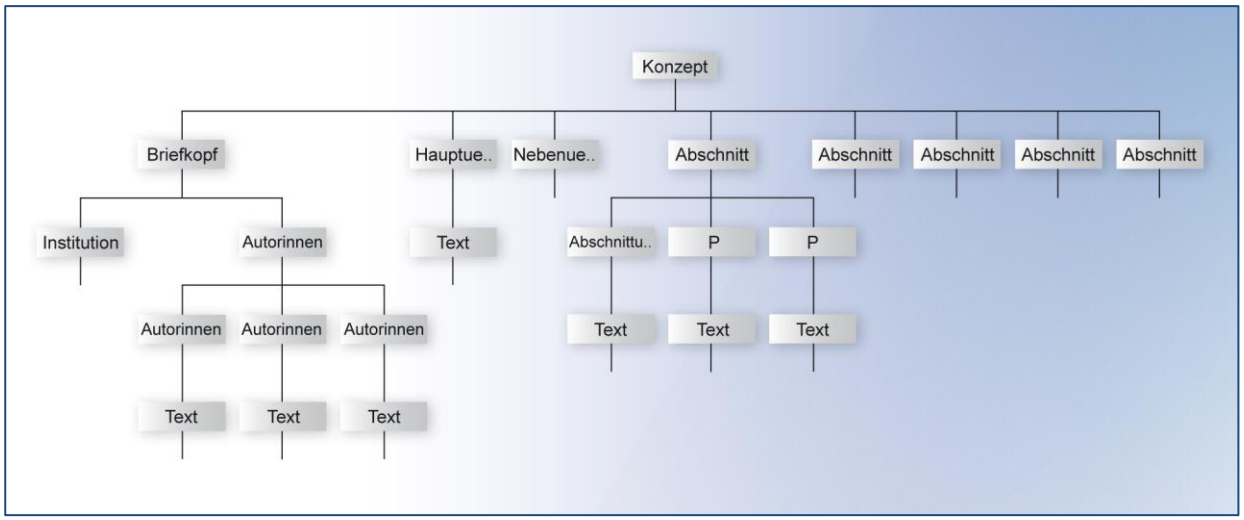


Abb.: 1.12 Dokumentdarstellung als Baum

1.6.4 Struktursicht mithilfe von eingebettetem Markup

```
<?xml version="1.0"?>
<!DOCTYPE Konzept SYSTEM "Konzept.dtd">

<Konzept>
<Briefkopf>
<Institution Logo="fbLogo"/>
<Autorinnen>
<Autorin>Prof. Dr. Anne Br&uuml;ggemann-Klein</Autorin>
<Autorin>Dr. Angelika Reiser</Autorin>
<Autorin>Prof. Dr. Doris Schmitt-Landsiedel</Autorin>
</Autorinnen>
</Briefkopf>
<Hauptueberschrift>
Ein Computer-Prop&auml;deutikum von Studentinnen f&uuml;r
Studentinnen
</Hauptueberschrift>
<Nebenueberschrift>
Konzept vom
<Datum>12.10.1999</Datum>
,Version
<Version>1.3</Version>
</Nebenueberschrift>
<Abschnitt>
<Abschnittueberschrift>
Hintergrund und Ziele
</Abschnittueberschrift>
<P>
Gerade in technischen Studieng&auml;ngen spielen
Computer als allt&auml;gliches Arbeitsmittel eine
wichtige Rolle.Es ist wesentlich, da&szlig; die
Studierendengleich zu Beginn des Studiums einen
vertrauten und sicheren Umgangmit diesem
Handwerkszeug haben. Eventuell vorhandene
Hemmschwellen m&uuml;ssen sobald wiem&ouml;glich
    abgebaut werden; davon ist eine
studienzeitverk&uuml;rzende Wirkung zu erwarten.
```

```
</P>
<P>
Es ist bekannt, da&szlig; Sch&uuml;lerinnen weniger
Zugangzu einem Computer haben als Sch&uuml;ler und
weniger h&auml;ufigeinen eigenen Computer besitzen.
Das Computer-Prop&auml;deutikum richtet sich deshalb
prim&auml;r anStudentinnen, die dadurch eventuell
fehlende Vorerfahrungenausgleichen k&ouml;nnen.
</P>
</Abschnitt>
<Abschnitt>
<Abschnittueberschrift>
   Das Kursangebot des Computer
Prop&auml;deutikums
</Abschnittueberschrift>

<P>Das Computer-Prop&auml;deutikum bietet Kurzeinf&uuml;hrungen
mit praktischen &Uuml;bungen in f&uuml;r das Studium an
der TU M&uuml;nchen relevante Themen rund um den Computer.
Jeder Kurs dauert-als Block angeboten-drei Stunden
und beinhaltet eine tutorielle Einf&uuml;hrung in das Thema,
praktische &Uuml;bungen am Rechner und eine
Abschlu&szlig;besprechung zu den &Uuml;bungen.
Kursthemen k&ouml;nnen sein:</P>

<UL>
<LI>Textverarbeitung mit Word</LI>
<LI>Textverarbeitung mit LaTeX</LI>
<LI>Textverarbeitung mit Star Office</LI>
<LI>E-Mail</LI>
<LI>Browsen und Informationsnutzung im World-Wide Web</LI>
<LI>Erstellen von Informationsangeboten im World-Wide Web (HTML)</LI>
<LI>Betriebssystem, Dateiorganisation</LI>
<LI>Gestaltung des Desktops (Window-Manager)</LI>
<LI>ASCII-Editoren (vi, emacs)</LI>
<LI>Kauf eines PCs</LI>
<LI>Datenverbindungen von zu Hause (analog, ISDN)</LI>
<LI>Fakult&auml;tsspezifische Informationsinfrastrukturen</LI>
</UL>

<P>Jeder Kurs sollte sich auf ein enges Thema konzentrieren.
Die Themenliste ist offen f&uuml;r Erg&auml;nzungen.
Kriterium ist, da&szlig; der
Kursinhalt prop&auml;deutischer Natur f&uuml;r das Studium an der TU ist
und eine Form der Computernutzung zum Inhalt hat.
Ein Kurs kann fakult&auml;ts&uuml;bergreifend
oder fakult&auml;tsspezifisch sein. Die Kurse sind in der Regel auf
eine spezielle Rechnerplattform (PC/Windows, Unix) zugeschnitten.</P>
</Abschnitt>

<Abschnitt>
<Abschnittueberschrift>Die Zielgruppe des
Computer-Prop&auml;deutikums</Abschnittueberschrift>

<P>Das Kursangebot richtet sich an Studentinnen im ersten Semester.
Wird ein Angebot von dieser Zielgruppe nicht ausgesch&ouml;pft, steht es
auch Studentinnen h&ouml;herer Semester und, in dritter Priorit&auml;t,
m&auml;nnlichen Studierenden offen.</P>
</Abschnitt>
```

```
<Abschnitt>
<Abschnittueberschrift>Die Leiterinnen und Leiter des
Computer-Prop&auml;deutikums</Abschnittueberschrift>

<P>Die Kurse werden vorzugsweise von Studentinnen h&ouml;herer Semester
konzipiert und durchgef&uuml;hrt. Zur Erg&auml;nzung des Kursangebots
k&ouml;nnen auch Mitarbeiterinnen und Mitarbeiter
oder Studenten h&ouml;herer Semester Kurse anbieten.
Nur von Studierenden angebotene Kurse k&ouml;nnen im Rahmen
des Programms finanziert werden.</P>

<P>Die Kursleiterinnen und -leiter sind selbst f&uuml;r die
Organisation von Seminar- und Rechnerr&auml;umen und von
Zugangsm&ouml;glichkeiten zu den R&auml;umen am Wochenende
f&uuml;r ihren Kurs verantwortlich. Wir gehen davon aus,
da&szlig; die Studiendekaninnen und -dekane sie bei der
Organisation unterst&uuml;tzen.</P>
</Abschnitt>

<Abschnitt>
<Abschnittueberschrift>Durchf&uuml;hrung</Abschnittueberschrift>

<P>Das Computer-Prop&auml;deutikum ist eine Initiative der
Frauenbeauftragten der TU M&uuml;nchen und wird vom
Frauenb&uuml;ro (Kerstin Hansen, Anja Quindeau) organisiert.
Die TU M&uuml;nchen finanziert das Programm aus dem Tutorienprogramm
zur Verk&uuml;rzung der Studiendauer.
F&uuml;r von Studierenden angebotene Kurse
wird ein Werksvertrag abgeschlossen, der
die Konzeption, Vorbereitung und Durchf&uuml;hrung
eines dreist&uuml;ndigen Kurses beinhaltet.
Konzeption und Vorbereitung eines Kurses werden mit 200 DM und jede
Durchf&uuml;hrung mit 100 DM honoriert.
Das Frauenb&uuml;ro erh&auml;lt Hilfskraftmittel
aus dem Tutorienprogramm f&uuml;r die
Organisation des Computer-Prop&auml;deutikums.
Wir rechnen mit folgenden Kosten: 20 Kurse,
jeder zweimal durchgef&uuml;hrt,
zu 20 mal 200 DM plus 20 mal 2 mal 100 DM, also 8000 DM,
plus 1000 DM f&uuml;r die Organisation im Frauenb&uuml;ro.</P>

<P>Eine erste <A HREF="Ausschreibung.html">Ausschreibung</A>
des Programms ist bereits erfolgt.
Interessierte Kursleiterinnen und -leiter
k&ouml;nnen ihr Kursangebot bis zum 5.11.1999 abgeben.
Bitte benutzen Sie dazu unser
<A HREF="Formular.html">Formular</A>.
Die ersten Kurse sollen noch im November 1999
stattfinden. Bei Bedarf erfolgt eine zweite Ausschreibung,
zu der neue Kursangebote abgegeben werden k&ouml;nnen,
Anfang Dezember 1999.
Die Ausschreibung des Programms, die Gestaltung des Kursangebots
sowie die Ank&uuml;ndigung der Kurse erfolgt durch das Frauenb&uuml;ro
in Zusammenarbeit mit den Studierendenvertretungen, den Studiendekanen
und den Frauenbeauftragten. Die Vertr&auml;ge werden nach Absprache
mit dem Frauenb&uuml;ro von der Hochschulverwaltung abgeschlossen.</P>

<P>Die Kurse sollen zu Randzeiten (Abends, Freitags Nachmittags
oder am Wochenende) stattfinden.</P>
</Abschnitt>
</Konzept>
```

List.: 1.4 Der Text als XML-Datei

1.6.5 Das Dokument im LaTeX-Format

```
\documentclass{article}
\usepackage[german]{babel}
\usepackage{myPage}

\def\Datum{\today}
\def\Version#1{{\bf #1}}

\begin{document}

\author{Prof. Dr. Anne Br"uggemann-Klein \and
     Dr. Angelika Reiser \and
     Prof. Dr. Doris Schmitt-Landsiedel}

\title{Ein Computer-Prop"adeutikum
     von Studentinnen f"ur Studentinnen
     (Konzept vom \Datum, Version \Version{1.3})}

\maketitle

\section{Hintergrund und Ziele}

Gerade in technischen Studieng"angen spielen
Computer als allt"agliches Arbeitsmittel eine wichtige Rolle.
Es ist wesentlich, da"s die Studierenden
gleich zu Beginn des Studiums einen vertrauten und sicheren Umgang
mit diesem Handwerkszeug haben. Eventuell vorhandene
Hemmschwellen m"ussen sobald wie m"oglich abgebaut werden;
davon ist eine studienzeitverk"urzende Wirkung zu erwarten.

Es ist bekannt, da"s Sch"ulerinnen weniger Zugang zu einem Computer
haben als Sch"uler und weniger h"aufig einen eigenen Computer besitzen.
Das Computer-Prop"adeutikum richtet sich deshalb prim"ar an
Studentinnen, die dadurch eventuell fehlende Vorerfahrungen
ausgleichen k"onnen.

\section{Das Kursangebot des Computer-Prop"adeutikums}

Das Computer-Prop"adeutikum bietet Kurzeinf"uhrungen
mit praktischen "Ubungen in f"ur das Studium an
der TU M"unchen relevante Themen rund um den Computer.
Jeder Kurs dauert-als Block angeboten-drei Stunden
und beinhaltet eine tutorielle Einf"uhrung in das Thema,
praktische "Ubungen am Rechner und eine
Abschlu"sbesprechung zu den "Ubungen. Kursthemen k"onnen sein:

\begin{itemize}
\item Textverarbeitung mit Word
\item Textverarbeitung mit LaTeX
\item Textverarbeitung mit Star Office
\item E-Mail
\item Browsen und Informationsnutzung im World-Wide Web
```

```
\item Erstellen von Informationsangeboten im World-Wide Web (HTML)
\item Betriebssystem, Dateiorganisation
\item Gestaltung des Desktops (Window-Manager)
\item ASCII-Editoren (vi, emacs)
\item Kauf eines PCs
\item Datenverbindungen von zu Hause (analog, ISDN)
\item Fakult"atsspezifische Informationsinfrastrukturen
\end{itemize}

Jeder Kurs sollte sich auf ein enges Thema konzentrieren.
Die Themenliste ist offen f"ur Erg"anzungen.
Kriterium ist, da"s der
Kursinhalt prop"adeutischer Natur f"ur das Studium an der TU ist
und eine Form der Computernutzung zum Inhalt hat.
Ein Kurs kann fakult"ats"ubergreifend
oder fakult"atsspezifisch sein. Die Kurse sind in der Regel auf
eine spezielle Rechnerplattform (PC/Windows, Unix) zugeschnitten.

\section{Die Zielgruppe des Computer-Prop"adeutikums}

Das Kursangebot richtet sich an Studentinnen im ersten Semester.
Wird ein Angebot von dieser Zielgruppe nicht ausgesch"opft, steht es
auch Studentinnen h"oherer Semester und, in dritter Priorit"at,
m"annlichen Studierenden offen.

\section{Die Leiterinnen und Leiter des Computer-Prop"adeutikums}

Die Kurse werden vorzugsweise von Studentinnen h"oherer Semester
konzipiert und durchgef"uhrt. Zur Erg"anzung des Kursangebots
k"onnen auch Mitarbeiterinnen und Mitarbeiter
oder Studenten h"oherer Semester Kurse anbieten.
Nur von Studierenden angebotene Kurse k"onnen im Rahmen
des Programms finanziert werden.

Die Kursleiterinnen und -leiter sind selbst f"ur die
Organisation von Seminar- und Rechnerr"aumen und von
Zugangsm"oglichkeiten zu den R"aumen am Wochenende
f"ur ihren Kurs verantwortlich. Wir gehen davon aus,
da"s die Studiendekaninnen und -dekane sie bei der
Organisation unterst"utzen.

\section{Durchf"uhrung}

Das Computer-Prop"adeutikum ist eine Initiative der
Frauenbeauftragten der TU M"unchen und wird vom
Frauenb"uro (Kerstin Hansen, Anja Quindeau) organisiert.
Die TU M"unchen finanziert das Programm aus dem Tutorienprogramm
zur Verk"urzung der Studiendauer. F"ur von Studierenden angebotene Kurse
wird ein Werksvertrag abgeschlossen, der
die Konzeption, Vorbereitung und Durchf"uhrung
eines dreist"undigen Kurses beinhaltet.
```

```
Konzeption und Vorbereitung eines Kurses werden mit 200 DM und jede
Durchf"uhrung mit 100 DM honoriert.
Das Frauenb"uro erh"alt Hilfskraftmittel
aus dem Tutorienprogramm f"ur die
Organisation des Computer-Prop"adeutikums.
Wir rechnen mit folgenden Kosten: 20 Kurse,
jeder zweimal durchgef"uhrt,
zu 20 mal 200 DM plus 20 mal 2 mal 100 DM, also 8000 DM,
plus 1000 DM f"ur die Organisation im Frauenb"uro.

Eine erste Ausschreibung des Programms ist bereits erfolgt.
Interessierte Kursleiterinnen und -leiter
k"onnen ihr Kursangebot bis zum 5.11.1999 abgeben.
Bitte benutzen Sie dazu unser
Formular. Die ersten Kurse sollen noch im November 1999
stattfinden. Bei Bedarf erfolgt eine zweite Ausschreibung,
zu der neue Kursangebote abgegeben werden k"onnen,
Anfang Dezember 1999.
Die Ausschreibung des Programms, die Gestaltung des Kursangebots
sowie die Ank"undigung der Kurse erfolgt durch das Frauenb"uro
in Zusammenarbeit mit den Studierendenvertretungen,
den Studiendekanen und den Frauenbeauftragten.
Die Vertr"age werden nach Absprache mit dem Frauenb"uro
von der Hochschulverwaltung abgeschlossen.

Die Kurse sollen zu Randzeiten
(Abends, Freitags Nachmittags oder am Wochenende) stattfinden.

\end{document}
```

List.: 1.5Auschnitt aus der LaTex Datei

1.6.6 Prinzipien aus der Informatik

Das Studium dieses Kurses setzt nicht voraus, dass andere Kurse aus dem Bereich der Informatik bereits belegt und erfolgreich absolviert wurden. Bei einem so spezialisierten Wissensgebiet wie es das Daten und Dokument-management darstellt, ist es jedoch unumgänglich, einige grundlegende Prinzipien zu verstehen. Die nötigen Basisinformationen folgen diesem Abschnitt.

Late Binding

Late Binding bedeutet, Entscheidungen so lange hinauszuzögern, bis sie unumgänglich sind. Das Prinzip des Late Binding bringt oft einen Gewinn an Flexibilität und wird in ganz unterschiedlichen Teilgebieten der Informatik angewendet. Polymorphie in objektorientierten Programmiersprachen ist ein Beispiel. Polymorphie bedeutet, dass in einer Klassenhierarchie spezialisierte Klassen die Methoden allgemeinerer Klassen, von denen sie erben, umdefinieren dürfen. Erst zur Laufzeit steht für ein jedes Objekt fest, wie es in

der Klassenhierarchie einzuordnen ist und welche der polymorph definierten Methoden für das Objekt Gültigkeit haben. Im Zusammenhang mit Stylesheets bedeutet der Begriff Late Binding folgendes: Erst wenn ein Dokument präsentiert werden soll, wird ein zum Anwendungszweck passendes Stylesheet zugeschaltet und das grafische Aussehen des Dokuments festgelegt.

Abstraktionsebenen

Levels of Abstraction einzuführen bedeutet, einen komplexen Sachverhalt ausgehend von einer hohen Abstraktionsstufe auf immer niedrigeren Abstraktionsstufen zu beschreiben und auf diese Weise einen konzeptuellen Rahmen herzustellen, in dem der Sachverhalt verstanden werden kann. Eine typische Anwendung des Prinzips ist die Rechnerarchitektur, in der ein Rechensystem auf den Ebenen Hardware, Maschinensprache, Betriebssystem und Anwendungssystem beschrieben wird. Bei DDM kommt das Prinzip der Levels of Abstraction im Zusammenhang mit Kodierungen zum Tragen, die im Rahmen des Kodierungsmodells auf immer konkreterer Ebene beschrieben werden.

Separation of Concerns

Die Trennung konzeptueller Dimensionen bedeutet, ein System so zu gestalten, dass unterschiedliche Aufgaben unabhängig voneinander durchgeführt werden können. Separation of Concerns erfordert entsprechende Modelle und Architekturen. Ein klassischer Fall von Separation of Concerns sind Datenbankanwendungen. Durch die Trennung in eine Anwendungsschicht, eine konzeptionelle Schicht und eine physikalische Schicht kann unabhängig voneinander an der Anwendungssoftware und an den Speicher- und Zugriffsstrukturen gearbeitet werden. Dies garantiert Datenunabhängigkeit. *Separation of Concerns* wird durch das Dokumentenmodell erzielt. Unabhängig voneinander können entsprechende Fachleute an der inhaltlichen Gestaltung eines Dokuments arbeiten.

Selbsttestaufgabe 1.15 Überlegen Sie, wo Ihnen die drei erwähnten Informatik-Prinzipien in Ihrem Studium schon einmal begegnet sind. Können Sie weitere grundlegende Prinzipien ausmachen?

1.7 Literatur

[1] J. André, R. Furuta, and V. Quint. Structured Documents. The Cambridge Series on Electronic Publishing. Cambridge University Press, Cambridge, 1989.

[2] B. Böhm.A spiral model of software development and enhancement, ACM, SIGSOFT, 1999. M. J. Dürst, F. Yergeau, R. Ishida, M. Wolf, A. Freytag, and T. Texin. Character model for the World Wide Web 1.0. www.w3.org/TR/charmod, February 2002. W3C Working Draft.

[3] M. J. Dürst, F. Yergeau, R. Ishida, M. Wolf, A. Freytag, and T. Texin. Character model for the World Wide Web 1.0. www.w3.org/TR/charmod, February 2002. W3C Working Draft.

[4] D. Engelbart. A conceptual framework for the augmentation of man's intellect. In P. W. Howerton and D. C. Weeks, editors, *Vistas in Information Handling*, volume 1, pages 1–29, Washington, DC, 1963. Spartan Books.

[5] T. Feuerstack, A. Hartmann, B. Vogeler, and J. Vieler. Einmal um die Erde und zurück: Unterwegs im Internet. Broschüre A/003/0811, Universitätsrechenzentrum FernUniversität Hagen, August 2011. www.fernuni-hagen.de/imperia/md/content/zmi_2010/a003_internet.pdf

[6] R. Furuta, J. Scofield, and A. Shaw. Document formatting: Survey, concepts, and issues. *Computing Surveys*, 14(3):417–172, September 1982.

[7] J. Korpela. A tutorial on character code issues, January 1999. Looking for a new home on the Internet.

[8] L. Lamport. LATEX: *A Document Preparation System, User's Guide & Reference Manual*. Addison-Wesley Publishing Company, Reading, MA, 1986.

[9] N. Meyrowitz and A. van Dam. Interactive editing systems (Parts I and II). *Computing Surveys*, 14(3):321–415, September 1982.

[10] T. H. Nelson. Embedded markup considered harmful, 2000. Published on XML.com

http://broadcast.oreilly.com/2009/08/microsoft-and-the-two-xml-pate-1.html

[11] D. R. Raymond, F. W. T. Tompa, and D. Wood. Markup reconsidered. Technical Report TR-356, Computer Science Department, University of Western Ontario, May 1993. http://www.csd.uwo.ca/tech-reports/356/tr356.abstract

[12] P. Weverka and D. A. Reid. *Word 2000: The Complete Reference*. Osborne/McGraw-Hill, Berkeley, 1999.

[13] K. Whistler and M. Davis. Character encoding model. Unicode Technical Report 17-3.1, Unicode, Inc., August 2000.

http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=IWS-Chapter03

[14] J. Wipper. Mathematik Online Kurs LaTeX.Darstellung mathematischer Ausdrücke: Pfeile, Universität Stuttgart, Februar 2009.

http://mo.mathematik.uni-stuttgart.de/kurse/kurs44/seite28.html

1.8 Lösungen der Selbsttestaufgaben

Selbsttestaufgabe 1.1 Bei Gemälden handelt es sich wegen des fehlenden sprachlichen Ausdrucks nicht um Dokumente. Personalausweise und Geldscheine sind eindeutig Dokumente ursprünglicher Art, für die das Trägermedium ebenso essentiell ist wie der sprachliche Informationsgehalt. Veranstaltungsplakate, Privatbriefe, Gedichtbände und Romane sind wohl in der Regel Dokumente neuerer Art, in denen neben dem Informationsgehalt auch die Formatierung eine Rolle spielt. Limitierte Auflagen und handschriftliche Briefe mit Unterschrift könnten wir jedoch genauso gut als Dokumente ursprünglicher Art einordnen. Memos, Bestellungen, Produktbeschreibungen und Fragebögen sehe ich je nach Kontext als Dokumente neuerer oder moderner Art an. Ein wichtiges Entscheidungskriterium ist, ob diese Dokumente digital oder auf Papier vorliegen. E-Mail-Nachrichten oder Nachrichten in einem Protokoll zum digitalen Zahlungsverkehr sind Nachrichten moderner Art, solange man den Protokollnachrichten den sprachlichen Ausdruck zugesteht.

Selbsttestaufgabe 1.2 Beispiele für sinnvolle Schachtelungen sind:

- Listen mit durchgängig nummerierten Aufzählungspunkten, die durch erläuternde Absätze unterbrochen werden können.

- Eine Gruppe von Zitaten mit Quellenangaben am Ende eines Kapitels.

Selbsttestaufgabe 1.3/4 Achten Sie auf Funktionen, die Struktur und Format trennen (Stylesheets) und auf Funktionen, die eine solche Trennung unterlaufen (explizite typografische Auszeichnung).

Selbsttestaufgabe 1.5 Tab.: 1.7gibt die Additionstabelle, Tab.: 1.8 gibt die Multiplikationstabelle für Hex-Ziffern an.

Addition:

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Tab.: 1.7 Die Additionstabelle für Hex-Ziffern

Multiplikationstabelle:

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	6A	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Tab.: 1.8 Die Multiplikationstabelle für Hex-Ziffern

Selbsttestaufgabe 1.6 Die Dezimalzahl 100 ist 64 in Hexadezimalnotation. Der prozentuale Anteil der Codepositionen von $E000_h$ bis $F8FF_h$ am Code-Raum von 0 bis $FFFF_h$ ist also in Hexadezimalnotation:

(F8FF – E000 + 1) · 64/(FFFF + 1), also 1900 · 64/10000, also 9C4/100. Knapp 10 Prozent der Unicode-Positionen im Bereich von 0 bis FFFF sind also für den privaten Gebrauch reserviert.

Selbsttestaufgabe 1.7 Die Unicode-Position F0000 wird unter UTF16 durch die beiden Surrogat-Words (F0000 – 10000) DIV 400 + D800 = DB80 und (F0000 – 10000) DIV 400 + DC00 = DC00 dargestellt.

Selbsttestaufgabe 1.8 Die Surrogat-Words DAFF und DEFF stellen die Unicode-Position (DAFF – D800) · 400 + DEFF – DC00 + 10000 = CFEFF dar.

Selbsttestaufgabe 1.9 Alle Zeichen der Zeichenfolge liegen im ASCII-Bereich von Unicode. Die Kodierung unter UTF8 und ASCII sind deshalb identisch; die Kodierung unter UTF16 ist um führende Nullen angereichert. Die Kodierung von „<?xml" unter UTF8 ist 3C3F786D6C, die unter UTF16 ist 003C003F00780060006C.

Selbsttestaufgabe 1.10 Die Positionen von 192 bis 255 werden unter UTF8 mit zwei Bytes kodiert nach dem Schema $110xxxxx_b10xxxxxx_b$. Da zur Kodierung der genannten Positionen nur acht Bits benötigt werden, die im Schema 110xxxxx 10xxxxxx von rechts aufgefüllt werden, hat das führende Byte des Kodifikats die Form 11000000, wobei xx die führenden beiden Bits der Hexzahlen C bis F sind, also 11000011, was C3 entspricht.

Selbsttestaufgabe 1.11 Das Zeichen „ü" hat die Position FC_h also 11111100_b. Die UTF8-Kodierung ist die Bitfolge 1100001110111100, bestehend aus den Unicode- oder ISO Latin-1-Positionen $C3_h$ für „Ã" und BC_h für „1/4". Es ist also denkbar, dass das Textsystem einen UTF8-kodierten Zeichenstrom als ISO-Latin-1-kodiert interpretiert.

Selbsttestaufgabe 1.12 Unter UTF8 wird $FEFF_h$ als Folge von drei Bytes dargestellt, nämlich EF_h, BB_h und BF_h.

Sowohl unter UTF16-BE als auch unter UCS2-BE wird BOM als dieselbe Folge von zwei Bytes dargestellt, nämlich FE_h und FF_h. Das BOM ist also kein hinreichendes Signal, um zwischen den bisher definierten Kodierungsschemata zu differenzieren.

Selbsttestaufgabe 1.13 native2ascii < test.ntv | native2ascii -reverse - encoding UTF8 > test.utf.

1.9 Index