Entwurf von benutzergerechten Informationssystemen im WWW

Peter Kaul

Diplomarbeit
Fachbereich Informatik der Universität Hamburg
November 2000

Betreuer
Prof. Dr. Horst Oberquelle
Prof. Dr. Heinz Züllighoven

Inhaltsverzeichnis

1. Einleitung

1.1. Problemstellung

Innerhalb kurzer Zeit hat sich das World Wide Web (WWW) von einem unbekannten Hilfsmittel der Wissenschaft zu einem populären Massenmedium entwickelt. Während noch vor wenigen Jahren der Begriff und die Funktionsweise des „Internets“ nur einem kleinen Kreis von Personen bekannt war, so ist es heutzutage in aller Munde und wird von einem zunehmend größeren Kreis an Personen auch genutzt. Dabei handelt es sich in keiner Weise mehr nur um technisch versierte Benutzer, sondern breite Kreise der Bevölkerung mit unterschiedlichem Wissen und Bedürfnissen nutzen dieses neue Medium.
Die Anwendungen, die sich aus diesem Medium erschließen, sind vielfältiger Natur: Beispielsweise werden unter dem Schlagwort „eCommerce“ webbasierte Systeme verstanden, mit denen online Produkte gekauft und auch bezahlt werden können. Ein anderer Anwendungsbereich versteht das WWW als ein Informationssystem. Dabei werden verschiedenste Daten bereitgestellt, auf die die Benutzer mittels des WWW Zugriff bekommen, wodurch sie grundsätzlich erwünschten Informationen erhalten können. Dieser Zugriff beinhaltet allerdings eine Menge an Problempotential: Neben den technischen Unzulänglichkeiten des WWW, z.B. in Form der Infrastruktur des Internets oder der verwendeten Software, ist insbesondere auch das hinter dem WWW stehende Konzept des „Hypertext“ nicht ohne Tücken: Ein ungünstig gestalteter Hypertext führt mitunter zu Verwirrung und Frustration beim Benutzer, wodurch oftmals der Zweck und die Sinnhaftigkeit einzelner Hypertexte in Frage gestellt werden muss.

1.2. Aufgabenstellung

In dieser Diplomarbeit soll – grob gesehen – aufgezeigt werden, welche Schritte notwendig sind, um ein benutzungsgerechtes Informationssystem in Form eines Hypertextes für das WWW zu gestalten. Natürlich sind derartige Ansätze nicht grundsätzlich unbekannt, wie die allgemein verfügbare Literatur (z.B. [Rosenfeld & Morville 98] oder [Fleming 98]) zeigt, dennoch kann diese Arbeit einen zusätzlichen Beitrag leisten: Zum einen wurde in keiner einzelnen Quelle eine ganzheitliche Sicht auf die Problematik geliefert, sondern immer nur Teilaspekte betrachtet und zum anderen erscheint das jeweils geschilderte Vorgehen oftmals als zu grobgranular, als dass sich daraus ein konkretes Vorgehen ableiten lassen könnte. Dieses Manko soll in diesem Rahmen behoben werden: Es sollen die verschiedenen Aspekte zusammengetragen, in Beziehung gesetzt und zusätzlich durch ein konkretes Vorgehensmodell ergänzt werden. Ziel soll es sein, dass der Leser dieser Arbeit die Zusammenhänge erkennt und in die Lage versetzt wird, ein webbasiertes Informationssystem zu gestalten. Dazu werden u.a. die Schritte aufgezeigt, die ein Entwickler durchlaufen muss, um bestimmte Aspekte einer Website festzulegen. Dazu gehören insbesondere auch die Möglichkeiten, mit denen derartige Entwurfsergebnisse dokumentiert werden können.
Ein besonderes Augenmerk soll auf die Benutzungsgerechtheit des Zielsystems gelegt werden, was im Wesentlichen eine Orientierung am Benutzer impliziert. Dazu werden Mittel und Wege aufgezeigt, mit denen Benutzer am Entwurfsprozess beteiligt werden und die Entwurfsergebnisse in Bezug auf die Zielsetzung bewertet werden können.

Als eine weitere Zielsetzung soll ein Werkzeug entwickelt werden, mit dem nach Möglichkeit der Entwurfsprozess unterstützen werden kann. Diese Entwicklung soll sich an den Ergebnissen der übrigen Ausführungen orientieren, so dass das Werkzeug in den Entwurfsprozess integriert werden kann. Als Grundidee für das Werkzeug soll die Möglichkeit dienen, eine Website auf Basis von entworfenen Teilaspekten automatisiert generieren zu lassen.

Aufgrund des sehr begrenzten Rahmens einer Diplomarbeit soll der Fokus auf dem „Entwurf“ einer Website liegen, womit prinzipiell die Spezifikation der „Website“ gemeint ist, auf dessen Basis die eigentliche „Implementation“ stattfinden kann. Wie in den folgenden Ausführungen deutlich werden wird, ist eine derartige Trennung von „Entwurf“ und „Implementation“ nicht klar zu leisten, sondern die Grenze ist vergleichsweise willkürlich zu setzen. Im Grunde soll damit auch nur festgelegt sein, dass bestimmte Aspekte, wie z.B. das Aufsetzen einer technischen Infrastruktur, nicht behandelt werden. Ebenfalls sollen alle Themen des Projektmanagement soweit wie möglich unberücksichtigt bleiben, was bedeutet, dass im Wesentlichen die produktbezogenen und weniger die prozessbezogenen Aktivitäten behandelt werden. Somit scheiden z.B. Fragen zum Zeitmanagement aus.

1.3. Aufbau der Arbeit

Zunächst soll in Kapitel 2 auf das grundsätzliche Wesen eines webbasierten Informationssystems eingegangen werden. Dazu wird auf vergleichsweise abstraktem Niveau auf das Grundkonzept des Hypertextes und dessen interaktive Komponente „Navigation“ eingegangen. Zusätzlich soll mittels der Behandlung der Themen „Kommunikation“ und „Information“ ein Verständnis darüber vermittelt werden, was in Bezug auf den Entwurf eines Informationssystems beachtet werden sollte. In diesem Zusammenhang wird deutlich werden, dass der „Entwurf von Websites“ im Wesentlichen den „Entwurf von Informationen“ meint.

In Kapitel 3 werden Aspekte der Benutzergerechtheit in Form der „Benutzerorientierung“ behandelt und allgemeine Methoden in Bezug auf ein Vorgehen aufgezeigt. Dazu wird auf Erkenntnisse der Software-Ergonomie bzw. des Software Engineering zurückgegriffen und konkrete Möglichkeiten in Bezug auf das WWW dargestellt. Als Ergebnis liegen dem Leser ein Satz von Methoden vor, die bei Einbeziehung in den Entwicklungsprozess für eine benutzungsgerechte Systemgestaltung sorgen.

Das Kapitel 4 geht auf Umstände der Website-Entwicklung ein und skizziert einerseits wichtige Rahmenbedingungen für ein Vorgehen und untersucht zusätzlich, inwieweit fertige Lösungen in Bezug auf ein Vorgehen angewandt werden können. Dazu wird insbesondere ein Augenmerk auf Prozessmodelle geworfen und deren Anwendbarkeit in Bezug auf die hier behandelte Problemstellung reflektiert. Ziel ist es, grundsätzliche Lösungsansätze zu bekommen, die in den Entwurf eines konkreten Prozessmodells mit einfließen können. Als Ergebnis soll zudem solch ein Prozess grob skizziert werden und als Grundlage für ein detailliertes Vorgehen an späterer Stelle dienen.

In Kapitel 5 wird ein Modell einer Website definiert, welches insbesondere die informations- und WWW-spezifischen Aspekte aus Kapitel 2 berücksichtigt und somit eine ganz besondere Sicht auf das Medium erlaubt. Ziel ist es, dass mittels dieses Modells die Bausteine festgelegt werden, aus denen im Entwurfsprozess die Website modelliert wird.

In den Kapiteln 6 und 7 wird letztendlich ein konkretes Vorgehen auf Basis des Entwicklungsprozesses, des Website-Modells und den Ergebnissen aus Kapitel 3 beschrieben. Das bedeutet, dass die einzelnen Schritte in Bezug auf Entwurf, Dokumentation und Bewertung dargestellt werden und somit als eine Anleitung für den Entwickler dienen. Soweit dies notwendig erscheint, wird dabei auch auf bisher nicht behandelte Aspekte des Entwurfs eingegangen.

In Kapitel 8 wird ein im Rahmen dieser Arbeit entwickeltes Werkzeug vorgestellt, mit dessen Hilfe eine Unterstützung beim Entwurfsprozess stattfindet. Dort lassen sich alle in den vorherigen beiden Kapiteln geschilderten Teilentwürfe auf einfache Weise zu einem Gesamtentwurf integrieren, so dass eine benutzbare Website entsteht. Mit Hilfe dieser Möglichkeit können Entwurfsergebnisse zu einem frühen Zeitpunkt bewertet und ggf. modifiziert werden, ohne dass eine mitunter zeitaufwendige Implementation stattfinden muss.

Letztendlich werden in Kapitel 9 die Ergebnisse der Diplomarbeit in Bezug auf die Zielsetzung resümiert und zusammengefasst.

2. Websites als Informationssysteme

Nachdem im vorherigen Kapitel die grundsätzliche Problemstellung für diese Diplomarbeit skizziert wurde, soll in diesem Kapitel eine Einführung in die Materie der Websites stattfinden. Dazu werden Aspekte geschildert, die das Wesen einer Website ausmachen und insbesondere Bezug auf die spezielle Verwendung als Informationssystem nehmen.

2.1. Hypertext

Diese Diplomarbeit beschäftigt sich mit der Erstellung von Websites. Bevor der Begriff der Website genauer durchleuchtet wird, soll zunächst das dahinterstehende Prinzip des Hypertextes behandelt werden, was in starker Anlehnung an die Ausführungen der Studienarbeit des Autors [Kaul 98] geschehen ist.

Unter einem Hypertext versteht man eine Sammlung von einzelnen Textdokumenten, sogenannten „Knoten“ oder „Nodes“, die auf bestimmte Art und Weise miteinander verknüpft sind. Jedes dieser Dokumente kann nämlich sogenannte „Hyperlinks“ zu anderen Dokumenten beinhalten, mit denen der Leser des Hypertextes die Möglichkeit hat, von einem Dokument zum nächsten zu springen.
Bei einer erweiterten Sicht des Begriffs Hypertext versteht man unter einem Knoten nicht mehr ausschließlich Textdokumente, sondern allgemeiner andere Medientypen, wie z.B. Bilder, Videos oder Musikstücke. In diesem Zusammenhang ist dann aber eher von Hypermedia statt Hypertext die Rede. In dieser Diplomarbeit soll der Begriff des Hypertextes verwendet werden, ohne damit die Einbeziehung anderer Medien ausschließen zu wollen. Des weiteren werde ich bei Hypertexten den Begriff Benutzer anstatt Leser verwenden, um damit der besonderen Rolle des Hypertextes gerecht zu werden.

Das Wesen eines Hypertextes wird besonders im Vergleich zu einem gewöhnlichen Text, wie z.B. einem Buch deutlich. Ein Buch besteht ebenfalls aus einer Sammlung von einzelnen Dokumenten, nämlich den Seiten. Der Benutzer bzw. Leser eines Buches wird sich normalerweise in einer linearen Sequenz durch das Buch bewegen, d.h. er wird zunächst Seite 1 lesen, dann Seite 2 usw. In der Reihenfolge des Lesens liegt der entscheidende Unterschied zum Hypertext. Während beim Buch die Reihenfolge vorgegeben ist, entscheidet der Benutzer eines Hypertextes von Situation zu Situation, welches Dokument er als nächstes lesen möchte. Wenn er auf einen Hy-

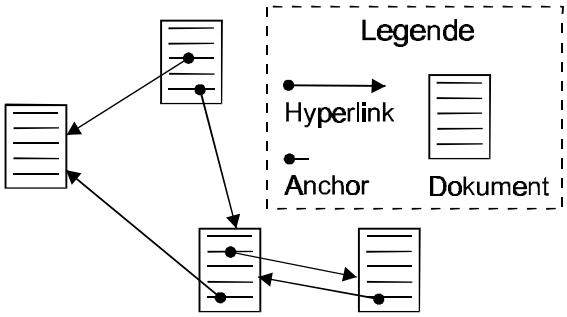

Abbildung 2.1: Ein Hypertext bestehend aus vier Knoten und fünf Hyperlinks.

perlink stößt, kann er entscheiden, ob er beim aktuellen Dokument bleibt oder aber dem Hyperlink zu einem anderen Dokument folgt. Ähnlich verhält es sich bei Fußno-

ten oder Verweisen in gewöhnlichen Texten, wie z.B. „siehe auch Kapitel 5.3“, wo der Leser eventuell seinen natürlichen Lesefluss unterbricht und an anderer Stelle weiterliest.

Eine typische Situation bei der Benutzung eines Hypertextes ist die folgende: Auf einem Dokument wird der Verfasser des Textes genannt und ein Hyperlink auf die Biographie dieses Menschen angeboten. Der Benutzer verwendet den Hyperlink und findet auf dem neuen Dokument wiederum Hyperlinks zu allen anderen Publikationen dieser Person. Er muss sich nun entscheiden, ob er zum ursprünglichen Text zurückkehrt oder aber, ob er eine der übrigen Publikationen lesen möchte. Auf diese Weise nimmt der Lesefluss des Benutzers immer neue Wendungen.

Wie erwähnt besteht ein Hypertext im Wesentlichen aus den beiden Komponenten „Knoten“ und „Hyperlink“. Die Position in einem Knoten, von der ein Hyperlink ausgeht, nennt man „Anchor“ (engl. für „Anker“). Dieser ist mit einem sogenannten „Label“ versehen, über das Hinweise vermittelt werden, was das Zieldokument beinhaltet. Dadurch kann der Leser abschätzen, was ihn auf dem jeweiligen Dokument für Inhalte erwarten.

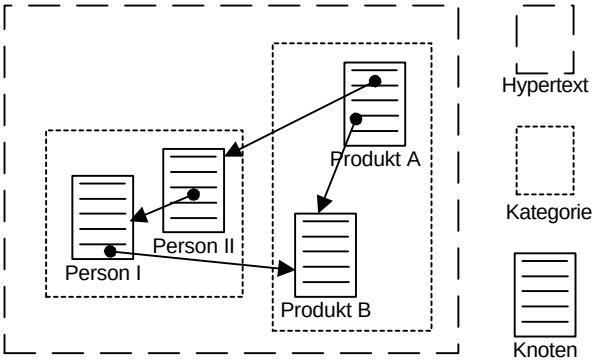

Abbildung 2.2: Ein Hypertext unter Betrachtung seiner verschiedenen Granularitäten „Knoten“, „Kategorie und „Hypertext“. Beispielsweise können die beiden in Knotenform vorliegenden Chunks „Person I“ und „Person II“ zu einer Kategorie zusammengefasst werden und somit als ein Chunk „Personen“ gesehen werden.

Ein Hypertext stellt einen Informationsraum dar, der als Summe der Informationen seiner einzelnen Knoten gesehen werden kann. Andersherum gesehen repräsentiert ein Knoten einen Ausschnitt aus dem gesamten Informationsraum und stellt für sich eine einzelne Informationseinheit dar. Diese Informationseinheit lässt sich in der Regel wiederum in kleinere Einheiten aufteilen, die sich ihrerseits wieder unterteilen lassen können: Beispielsweise wird ein Knoten, in dem Informationen über eine Person vermittelt werden, eine kleinere Informationseinheit „Name“ enthalten, die sich wiederum in „Vorname“ und „Nachname“ unterteilen lässt.
Eine solche Informationseinheit nennt sich jeweils „Chunk of Knowledge“ bzw. „Chunk“, wobei damit nichts über ihre Größe ausgesagt wird. Prinzipiell setzt sich jeder Chunk wiederum aus mehreren Chunks zusammen und ist selbst wieder Bestandteil eines übergeordneten Chunks. Auf diese Weise lassen sich Chunks von verschiedener „Granularität“ definieren, wie z.B. „Hypertext“, „Kategorie“, „Knoten“, usw.
Unter der „Granularität“ versteht man den Grad der Feinheit eines Chunks. Bei einem Hypertext ist die Frage interessant, wie die Granularität eines Knotens gewählt wer-

den sollte, d.h. z.B. ob er viele Informationseinheiten beinhalten soll, oder ob diese besser auf mehrere Knoten verteilt werden sollen. Diese Frage ist als „Granularitätsproblem“ bekannt und lässt sich nicht eindeutig beantworten, weil sowohl eine zu große Granularität, als auch eine zu kleine problematisch sind [Schulmeister 96, S. 267f].

2.2. WWW und Websites

Seit seiner „Erfindung“ im Jahr 1945 von Vannevar Bush [Nielsen 96c, S. 33] hat es verschiedene Hypertextsysteme gegeben, so z.B. das 1987 vorgestellte „HyperCard“ [Nielsen 96c, S. 58]. Diese Diplomarbeit soll sich aber ausschließlich mit der speziellen Ausprägung in Form einer „Website“ im WWW auseinandersetzen.

Beim WWW handelt es sich um ein weltumspannendes Netzwerk von Systemen, bei denen es sich u.a. um „Web-Server“ (kurz: Server) und „Web-Clients“ handelt. Die „Web-Clients“ sind auch als „Browser“ bekannt und haben prominente Vertreter in Form des „Netscape Navigators“ oder „Microsoft Internet Explorer“.
Auf den Servern sind Dokumente abgelegt, die von den Benutzern mithilfe der Browser abgerufen und dargestellt werden können. Bei den Dokumenten kann es sich um Texte, Bilder, Videos usw. handeln, wobei aber vor allem solche interessant sind, die im sog. HTML-Format vorliegen. HTML (Hypertext Markup Language) ist eine Sprache zur Beschreibung von Hypertextknoten und ist damit zur Erstellung von Hypertexten geeignet. Neben dem Umstand dort z.B. Bilder und Texte zu kombinieren zu können, sind insbesondere die Möglichkeiten zur Definition von Hyperlinks entscheidend. Durch das Erstellen einzelner Knoten („Seiten“) mittels HTML und der Verknüpfung untereinander über Hyperlinks lassen sich beliebige Hypertexte, sog. „Websites“[1] definieren.

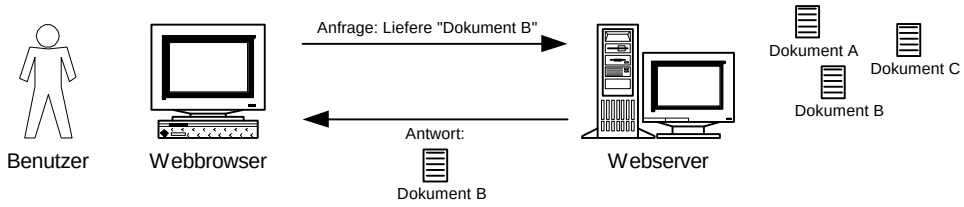

Abbildung 2.3: Ein Benutzer fordert mit einem Webbrowser (z.B. über einen PC) ein Dokument mittels Anfrage vom Webserver an, das ihm dann als Antwort zurückgeliefert wird.

Einzelne Seiten werden über den Browser mittels einer „Anfrage“ („Request“) vom Server angefordert und von diesem mittels einer „Antwort“ („Response“) übertragen. Der Browser stellt dem Benutzer diese Seite dann inklusive der Hyperlinks dar, wobei die Benutzung eines Hyperlinks wiederum zu einem Abruf mittels Anfrage/Antwort führt. Auf diese Weise kann der Benutzer durch das WWW wandern (auch bekannt als „surfen“, „browsen“ oder „navigieren“). Eine Website beinhaltet in der Regel eine bestimmte Seite, die sog. „Homepage“, die explizit als Einstiegspunkt dient und von der aus der Benutzer seine Navigation durch die Website beginnt.

Jedes Dokument im WWW wird über eine Adresse, der sog. URL (Uniform Resource Locator) [Berners-Lee et al. 94] angesprochen, die bezogen auf das WWW jeweils eindeutig ist. Somit besteht die Möglichkeit, dass auch Hyperlinks auf Dokumente außerhalb einer Website gesetzt werden können. So gesehen besteht das WWW

[1] „Site“ (engl.) für „Ort“.

aus mehreren gigantischen Hypertexten, die sich ihrerseits aus einzelnen oftmals miteinander verbundenen Websites zusammensetzen. Diese Definition eines Hypertextes im WWW ist in diesem Kontext nicht befriedigend, weshalb hier als Hypertext bzw. Website eine Sammlung von Dokumenten verstanden werden soll, die miteinander verknüpft sind und explizit als zusammengehörig erklärt werden. Obwohl es sich bei einer Website damit jeweils nur um einen Bruchteil der im gesamten WWW vorhandenen Dokumente handelt, soll dies aber nicht bedeuten, dass nicht auch Hyperlinks zu Dokumenten außerhalb verwendet werden.

Bei den in einer Website gesammelten Dokumenten handelt es sich wie erwähnt um alle möglichen Medientypen (Texte, Bilder, Videos, ...), wobei die HTML-Seiten als die wichtigsten angesehen werden sollen. Zwar können auch die übrigen Dokumente Hypertextknoten darstellen, wobei aber deren mangelnde Möglichkeit Hyperlinks zu beinhalten dazu führt, dass sie innerhalb einer Website eine „Sackgasse“ darstellen. Stattdessen werden sie häufig in eine Seite „eingebettet“, d.h. sie stellen dann aus Sicht des Benutzers einen integralen Bestandteil dieser Seite dar. Technisch gesehen, wird in der Seite allerdings nur ein Verweis auf das jeweilige Dokument gesetzt, wodurch der Browser aufgefordert wird, diese Dokument abzurufen und zusätzlich darzustellen.

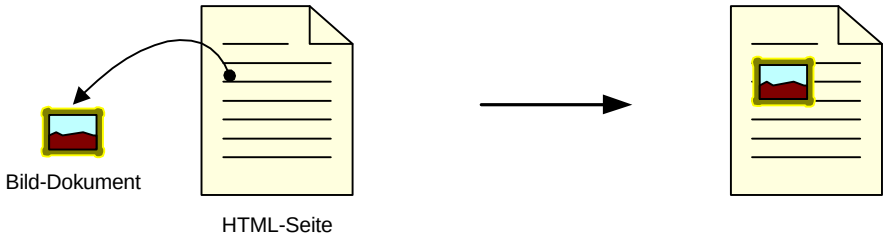

Abbildung 2.4: Von einer HTML-Seite aus ist ein Verweis auf ein Bild-Dokument gesetzt (links), was zu einer eingebetteten Darstellung im Webbrowser führt (rechts).

Dokumente einer Website und insbesondere HTML-Seiten werden in statische und dynamische Dokumente unterteilt. Als dynamisch sind solche zu verstehen, die erst zum Zeitpunkt der Anfrage generiert werden und somit Bezug auf zusätzliche, in der Anfrage übermittelte, Parameter nehmen können. Statisch hingegen sind solche, die schon vor der Anfrage komplett bereitliegen. Mithilfe dynamischer Seiten wird es z.B. möglich, eine Website für transaktionsorientierte Zwecke zu verwenden.

Es ist möglich, Websites für gänzlich unterschiedliche Zwecke und Zielrichtungen zu erstellen. Zum einen ergeben sich Möglichkeiten in Bezug auf „eCommerce“, oder aber auch zur reinen Übermittlung von Informationen. Zum anderen spricht man bei der Nutzung einer Website als Informationsträger auch von einem „Web Information System“ (WIS). Wie einleitend erklärt wurde, sollen Websites dieser Gattung der ausschließliche Gegenstand der Diplomarbeit sein, weshalb im Folgenden die Begriffe Website und WIS als Synonyme verwendet werden. Als zusätzliche Einschränkung soll hier gelten, dass ein derartiges Informationssystem ausschließlich statisch vorliegt, d.h. dass die Inhalte im Vorfeld fest definiert sind und nicht erst bei der Abfrage erzeugt werden.

2.3. Kommunikation

Wie im vorherigen Kapitel dargestellt wurde, sollen Websites in dieser Arbeit als Träger für Informationen gesehen werden. Diese Informationen werden von bestimmten

Personen zu einem bestimmten Zweck bereitgestellt und von Benutzern aus jeweils einem bestimmten Grund abgerufen. So gesehen findet indirekt eine Übertragung von Informationen von ersteren Personen zu den Benutzern statt, was als Vorgang als „Kommunikation“ bezeichnet wird (vgl. [Engesser 93, S. 339]).

Abstrakt gesehen werden bei einer Kommunikation Informationen von einer sendenden Person mithilfe eines Mediums bzw. Kanals (z.B. der Website) an einen Empfänger (z.B. den Benutzer) übermittelt. Vor dem Transport muss die Information allerdings noch kodiert, d.h. so umgewandelt werden, dass sie dem Medium gerecht wird. Nach dem Transport findet dann eine Dekodierung statt, d.h. sie wird in eine Form gebracht, die dem Empfänger gerecht wird.
Beispiel: Ein Mensch (Sender) möchte einem anderen Menschen (Empfänger) seinen Namen (Information) mitteilen. Dabei bedient er sich seiner Sprache (Medium) und wandelt die Information in Laute um (Kodierung), die zum Empfänger transportiert und dort über dessen Wahrnehmungsapparat empfangen und dekodiert werden. Wenn der Empfänger nicht taub ist und zudem die Sprache des Senders spricht, so besteht die Möglichkeit, dass er das Empfangene tatsächlich als die intendierte Information versteht. Andererseits können Störungen im Kommunikationsprozess, wie z.B. das Sprechen einer fremden Sprache, dazu führen, dass das Empfangene falsch bzw. ohne Wert aufgefasst wird.

Um den Kommunikationsprozess noch besser zu verstehen, soll hier der Begriff des Zeichens eingeführt werden: Unter einem Zeichen versteht man ein physisches Phänomen, welches von einem Menschen als solches erkannt und somit interpretiert wird. Eine Interpretation setzt einen sog. „Kode“ voraus, der entweder von Sender und Empfänger vereinbart ist, oder aber als allgemeine soziale Konvention existiert. Wenn ein gemeinsamer Kode zwischen Sender und Empfänger nicht existiert, so kann eine Information nicht oder nicht richtig übermittelt („kommuniziert“) werden [Züllighoven 93, S. 6ff]. Zeichen existieren in den verschiedensten Ausprägungen: So können Töne und Berührungen genauso wie Buchstaben unseres Alphabets als Zeichen empfunden werden[2].

Beispiel: Das in unserer Gesellschaft verwendete lateinische Alphabet (A,B,C, ...) stellt eine Sammlung von visuellen Zeichen dar, die von vielen Menschen interpretiert werden können. Allerdings kann erst über die Verknüpfung von Zeichen aus dieser Menge eine sinnvolle Information transportiert werden. Solche Verknüpfungen können Strukturen ergeben, die wir als Worte, Sätze, Kapitel, Text usw. kennen.
Aus dem gemeinsamen sozialen Umfeld ergibt sich ein „Kode“ (z.B. durch das gemeinsame Alphabet bzw. die gemeinsame Sprache), der es beispielsweise erlaubt, die gedruckte Fassung dieser Diplomarbeit als Information zu interpretieren, anstatt sie nur als schwarz-weiße Reize auf der Netzhaut wahrzunehmen.

Dass Kommunikation nicht zufällig zustande kommt, stellt [Kornatzki 89, S. 194] fest: „Jede Kommunikation zielt auf Wirkung: verbale und visuelle Information ist weder Selbstzweck, noch dient sie nur dazu, Unkenntnis zu beseitigen; vielmehr soll sie Meinungen und Verhalten in gewünschter Weise beeinflussen“. Dies bedeutet logischerweise, dass auch Websites nicht zweckfrei entstehen, sondern dass das Bereitstellen von Informationen durchaus mit einer Absicht geschieht.

[2] Dies ist Thema der „Semiotik“ und wird u.a. in [Eco 77] behandelt.

2.4. Information

Um eine Website in ihrer Eigenschaft als Kommunikationsmedium besser zu verstehen, ist es notwendig, sich genauer mit dem Begriff der Information zu befassen. Zwar wird dieser Begriff tagtäglich verwendet und die meisten Menschen haben eine ungefähre Ahnung, was sich hinter diesem Begriff verbirgt, aber „[...] trotzdem ist er bisher kaum präzisiert worden“ [Engesser 93, S. 315]. Dennoch sollen an dieser Stelle einige Erklärungsversuche wiedergegeben werden:

Nathan Shedroff bettet den Begriff „Information“ in einen größeren Kontext ein, in der „Daten“ die Basis von Informationen sind, die wiederum die Basis von „Wissen“ und „Weisheit“ darstellen. Dazu erklärt er folgendes: „Data is the product of research, creation, collection, and discovery. It is the raw material we find or create that we use to build our communication” [Shedroff 99, S. 271f]. In [Züllighoven 93, S. 12] wird zudem festgestellt, dass „Information“ etwas persönliches ist, während Daten nur Repräsentanten der Information sind. Daten stellen also nüchterne Sachverhalte da, die erst beim einzelnen Menschen Bedeutung erlangen und somit zu einer Information werden können.
Beispiel: Der Satz „Peter Kaul ist Student an der Universität Hamburg“ stellt ohne weiteres keine Information dar, sondern ist zunächst nur eine kontextfreie Aussage. Dem Leser dieser Diplomarbeit wird möglicherweise aufgefallen sein, dass „Peter Kaul“ der Name des Autors ist. Somit bekommt dieses Datum im „Kontext“ dieser Diplomarbeit eine „Bedeutung“, womit es für manche Leser auch eine Information darstellt.

Daten müssen also in einem Kontext stehen, damit aus ihnen überhaupt Informationen entstehen können. Kontext besteht z.B. durch persönliches Vorwissen oder auch über direkt umgebende Daten und Informationen. Je nach Kontext, werden Daten von Menschen unterschiedlich interpretiert, wodurch die daraus resultierenden Informationen unterschiedliche Bedeutungen haben können.

Im Verlauf dieser Diplomarbeit wird alternativ zu „Information“ und „Daten“ der Begriff des „Inhalts“ bzw. „Content“ verwendet. Damit sollen die Daten der Website gemeint sein, die potentiell eine Information darstellen.

2.5. Organisation

Wie im vorherigem Kapitel ausgeführt wurde, sind Daten und Information nicht gleichbedeutend. Damit aus Daten Informationen werden, müssen diese in einem Kontext auftreten, was aber andersherum die Möglichkeit ergibt, Informationen aktiv zu entwerfen. Dabei ist der Vorgang der Organisation ein Mittel, um Kontext zu erschaffen: „Transforming data into information is accomplished by organizing them into a meaningful form, presenting them in appropriate way, and communicating the context around them.“ [Shedroff 99, S. 272]. Ähnlich äußert sich auch [Turner 99]: "In order for it to become communicable, information needs to be organized into a meaningful form, in ways that make clear the relationship between its parts".
Dass es sich bei dem Organisieren von Dingen um etwas für den Menschen sehr Natürliches handelt, zeigt die Tatsache, dass wir Menschen nahezu alles in Kategorien sehen, sei es z.B., dass Musik nach Stilrichtungen und Menschen nach Beruf klassifiziert werden. Dahinter steckt nach [Rosenfeld & Morville 98, S. 22] ein Prinzip, das sie folgendermaßen erklären: „We organize to understand, to explain, and to control“.

Das Organisieren bedeutet, dass zwischen einzelnen Dingen Beziehungen hergestellt werden, z.B. indem jeweils gleichartige Dinge zu einer Gruppe zusammengefasst werden. Eine solche Organisation beinhaltet zwei Teilaspekte, nämlich zum einen ein „Schema“ und zum anderen eine „Struktur“. Ein Schema legt fest, welches Charakteristikum zur Organisation herangezogen wird (z.B. „alphabetisch“), während die Struktur beschreibt, in welcher Beziehung die schematisierten Dinge stehen [Rosenfeld & Morville 98, S. 26]. In Bezug auf Hypertexte wird die Struktur im Endeffekt über die Hyperlinks zwischen den Knoten ausgedrückt.

Schemata
Shedroff hat einige Basisschemata ausgemacht, wie z.B. „Alphabet“, „Ort“ und „Zeit“, nach denen sich seiner Meinung nach alle Dinge grundsätzlich organisieren lassen. Er geht davon aus, dass alle zu organisierenden Dinge mindestens ein gemeinsames Attribut haben, auf das sich mindestens eines der Schemata anwenden lässt [Shedroff 99, S. 275ff].

Strukturen
In [Ward & Codrai 98] werden eine Reihe von grundsätzlichen Strukturen dargestellt, die in Bezug auf Hypertexte eine Rolle spielen. Zwar gibt es theoretisch eine unendliche Anzahl möglicher Strukturen, wobei bestimmte, wie z.B. „Sequenz“, „Modul“, „Matrix“, „Hierarchie“ und „Netz“, bei der Gestaltung von Hypertexten offenbar häufiger Anwendung finden. Diese Strukturen sollen hier nicht grundsätzlich genauer vorgestellt werden, sondern es sei stattdessen auf o.g. Quelle verwiesen. Der Struktur „Hierarchie“ hingegen sei aufgrund ihrer herausragenden Bedeutung das nächste Kapitel gewidmet.

Organisationen können aufgrund unterschiedlicher Schemata und Strukturen höchst unterschiedlich ausfallen. Für Menschen, die auf solch eine Organisation zugreifen, ergeben sich daher mitunter Probleme das Organisationsschema bzw. die Struktur zu durchschauen. Gründe dafür liegen nach [Rosenfeld & Morville 98, S. 23ff] u.a. in der Mehrdeutigkeit bestimmter Begriffe, in der Heterogenität der Dinge oder unterschiedlichen Sichtweisen darauf. Aufgrund dieser Probleme kann es vorkommen, dass das Ergebnis einer Organisation unlogisch oder inkonsistent erscheint.

2.6. Hierarchische Struktur

Wie oben schon angedeutet, spielt die Struktur „Hierarchie“ in Bezug auf Hypertexte eine herausragende Rolle. So stellt beispielsweise [Scharl 99] fest: „The concept of hierarchy is essential for the architecture of any real-world WIS”. In ähnlicher Weise äußern sich beispielsweise auch [Rosenfeld & Morville 98, S. 37]. Aus diesem Grund soll der Hierarchie in dieser Diplomarbeit ebenfalls eine zentrale Rolle zukommen. Wenn im Folgenden von der Strukturierung eines Hypertextes gesprochen wird, soll damit eine „hierarchische Strukturierung“ gemeint sein. Auf eine genauere Behandlung in Bezug auf andere Strukturen wird somit verzichtet.

Bei einer Hierarchie werden die Knoten ausgehend von einem Wurzelknoten („Root“) nach einem Vorgänger-Nachkommen-Prinzip verknüpft, wodurch grundsätzlich eine baumartige Struktur entsteht. Das bedeutet, dass jeder Knoten (außer dem Wurzelknoten) einen Vorgänger hat und eventuell einen oder mehrere Nachkommen. Sofern ein Knoten keine Nachkommen hat, wird er „Blatt“ genannt.

In einem hierarchisch strukturierten Informationsraum findet das Prinzip der Chunks gemäß Kapitel 2.1 eine Anwendung: Dort stellt jeder Knoten einen Inhalt dar, der

über eventuelle Nachkommen verfeinert wird. Andersherum repräsentiert jeder Knoten den Teilaspekt eines übergeordneten Inhalts, sofern man einmal vom Wurzelknoten absieht. Jeder einzelne Knoten definiert somit wieder eine Teilhierarchie, deren Wurzelknoten er darstellt. Solch eine Teilhierarchie soll im Folgenden „Kategorie“ genannt werden und dabei soll auch auf das verwandte Konzept der „Sub-Site“ (vgl. [Nielsen 96a]) hingewiesen werden.

In [Ward & Codrai 98] wird eine Hierarchie in Bezug auf einen Hypertext gedeutet und dabei idealisiert hergeleitet, welche Funktion die einzelnen Knoten haben sollten: In den Blättern einer Hierarchie sind die einzelnen Inhalte untergebracht, die die Bedeutung des Informationsraumes ausmachen. Blätter haben jeweils einen Vorgänger in Form eines übergeordneten Knotens, der als inhaltliche Übersicht über alle seine Nachfolger dient. Analog lässt sich dies jeweils eine Ebene höher betrachten, bis dabei rekursiv der Wurzelknoten („Homepage“) erreicht ist, der als Übersicht über die gesamten übrigen Knoten dient.
Bei einer idealisiert betrachteten Navigation werden, ausgehend von der Homepage, schrittweise einige Übersichtsknoten durchlaufen, die jeweils eine Annäherung an die gesuchte Information in Form eines Blattes darstellen.

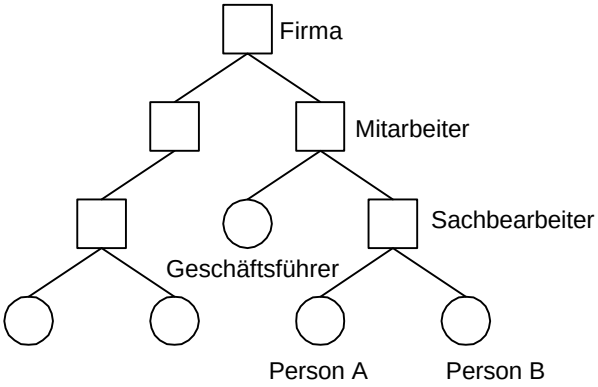

Abbildung 2.5: Ein hierarchischer Informationsraum, bei dem die Blätter (Geschäftsführer, Person A und Person B) die eigentlichen Inhalte darstellen. „Person A“ und „Person B“ werden mittels des Übersichtsknotens „Sachbearbeiter“ und dieser wiederum zusammen mit „Geschäftsführer“ vom Knoten „Mitarbeiter“ zusammengefasst. „Firma“ dient dabei als Wurzelknoten, der eine Übersicht über den gesamten Informationsraum, d.h. auch über die hier unbeschrifteten Knoten, darstellt.

Menschen erscheint die Art und Weise, mit der Dinge in einer Hierarchie organisiert werden, als grundsätzlich natürlich. Mittels rekursiver Zuordnung einzelner Dinge zu sich gegenseitig ausschließenden Kategorien bzw. Unterkategorien wird ein klares Prinzip beschrieben, nach dem sich andersherum auch einzelne Dinge wiederfinden lassen.
In Bezug auf die Bedeutung einzelner Inhalte innerhalb der Hierarchie wird in [Turner 99] darauf hingewiesen, dass neben der Vorgänger-Nachkommen-Beziehung auch die Reihenfolge der jeweiligen Nachkommen von Bedeutung ist. Einem Inhalt an erster Stelle wird zum Beispiel in Bezug auf die anderen die höchste Wichtigkeit beigemessen.

Beim Erstellen einer hierarchischen Struktur tritt mitunter das Problem auf, dass sich Dinge nicht eindeutig einer Kategorie zuordnen lassen, sondern thematisch an mehrere Stellen gehören. Eine Möglichkeit, diesem Problem zu begegnen, wäre das Duplizieren des jeweiligen Inhalts, so dass er als jeweils eigenständiger Inhalt an

mehreren Stellen auftritt. Gerade in Hinblick auf die dadurch beeinträchtigte Orientierung des Benutzers im Hypertext (vgl. Kapitel 2.7) ist dieses Vorgehen mehr als unbefriedigend. Eine bessere Möglichkeit stellt hierbei die Verwendung eines zusätzlichen Hyperlinks dar, so dass statt auf eine Dublette auf das einzigartige Original in einer anderen Kategorie verwiesen wird.

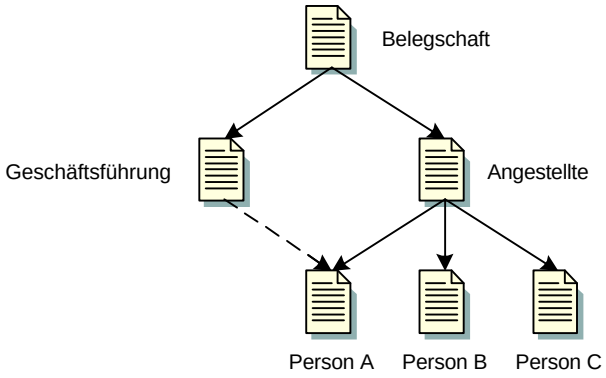

Abbildung 2.6: Anstatt den Inhalt „Person A“, der sowohl der Kategorie „Angestellte“ als auch zu „Geschäftsführung“ zugeordnet werden kann, zu duplizieren, kann auch ein assoziativer Hyperlink (gestrichelt) gesetzt werden. Die hierarchische Struktur, ausgedrückt durch die strukturellen (nicht-gestrichelten) Hyperlinks, bleibt dadurch erhalten.

Von [Isakowitz, Stohr & Balasubramanian 95, S. 41] wird ein Modell vorgestellt, bei dem Hyperlinks in zwei Arten klassifiziert werden: Die strukturellen und die assoziativen Hyperlinks. Unter den strukturellen Hyperlinks werden jene verstanden, die eine klare Struktur (beispielsweise eine Hierarchie) definieren, während die assoziativen Hyperlinks beliebig über diese Struktur hinausgehen.
Somit kann eine Struktur sauber definiert werden, ohne auf die Möglichkeit der „Querverweise“ verzichten zu müssen. Anders gesehen wird mit den assoziativen Hyperlinks eine alternative Struktur geschaffen, die parallel zu der „Hauptstruktur“ der strukturellen Hyperlinks existiert und der Navigation durch den Hypertext mehr Flexibilität verschafft.

2.7. Navigation

In den vorherigen Kapiteln ist häufiger der Begriff der „Navigation“ verwendet worden. Intuitiv sollte klar sein, dass es sich dabei um den interaktiven Aspekt von Hypertexten handelt, sich durch die einzelnen Knoten zu bewegen: Dadurch, dass der Benutzer die Möglichkeit hat, zu jedem Zeitpunkt den aktuellen Knoten mittels eines Hyperlinks zu verlassen, interagiert er mit dem Hypertextsystem.

Um diesen Vorgang besser zu verstehen, soll zunächst die Motivation des Benutzers hinterfragt werden. Man unterscheidet zwei grundsätzliche Vorgehensweisen bei der Navigation, nämlich die „erforschende“ und die „suchende“ [Nielsen 96c, S. 267]. Bei der erforschenden Navigation durchwandert der Benutzer den Hypertext ohne ein konkretes Ziel, sei es um sich einen Überblick zu verschaffen, oder aber einfach nur um vielleicht etwas Interessantes zu entdecken. Bei der suchenden Navigation hingegen steht das konkrete Bedürfnis im Vordergrund, eine bestimmte Information zu erlangen. Im Folgenden soll grundsätzlich davon ausgegangen werden, dass sich die Benutzer eines Hypertextes grundsätzlich auf der Suche nach einer Information befinden.

Aus Sicht des Benutzers stellt sich der Navigationsvorgang wie folgt dar: Als Ausgangspunkt befindet er sich auf einem Knoten und möchte zu einem weiteren gelangen. Dieser unbekannte Zielknoten beinhaltet die erwünschte Information und ist vermeintlich durch einen Pfad weiterer Knoten zu erreichen. Im Idealfall wird der Benutzer Schritt für Schritt die Knoten eines solchen Pfades durchlaufen, bis er sein Ziel erreicht hat. Im ungünstigen Fall hingegen wird er irgendwann auf dem Weg sein Vorhaben abbrechen, weil er z.B. die Orientierung verloren hat.
Abhängig von seiner derzeitigen Suchstrategie und den Informationen, die ihm augenblicklich zur Verfügung stehen, wird er sich bei jedem Schritt orientieren, eine Entscheidung treffen und dementsprechend handeln: Er wird entweder zum nächsten Knoten springen, oder aber sein Vorgehen modifizieren bzw. abbrechen [Jul & Furnas 97, S. 46f].

Der Vorgang der Navigation beinhaltet einiges an Problempotential, wobei als Grundproblem das Nichtauffinden einer gewünschten Information zu sehen ist. Dies kann zum einen den ganz banalen Grund haben, dass sie in dem jeweiligen Hypertext nicht existiert, zum anderen auch, dass der Benutzer an der falschen Stelle sucht, oder aber beim Suchen die Orientierung verliert („Lost-in-Hyperspace“).
Entgegen der These, dass Probleme dieser Art grundsätzlich im Wesen des Hypertextes verankert sind, merkt Kuhlen an, dass sie in „[...] einer fehlerhaften Konzeption bzw. einer unzureichenden Modellierung der betreffenden Hypertextbasis liegen.“ [Kuhlen 91, S. 134]. Mit anderen Worten: Mithilfe geeigneter Maßnahmen bei der Gestaltung von Hypertexten kann man diese Probleme zumindest verringern.

Um eine Information in einem Hypertext zu finden, ist es vorteilhaft zu wissen, an welcher Stelle es sich lohnt zu suchen. Andernfalls wäre man gezwungen systematisch jeden Knoten zu durchlaufen und auf Brauchbarkeit zu reflektieren. Eine natürliche Suchstrategie ist es, an einem Ort zu suchen, an dem sich ähnliche Informationen befinden. Von dort aus würde man versuchen sich dem Ziel Stück für Stück zu nähern. Dafür ist es unerlässlich, dass die Informationen durch ein sinnvolles Schema organisiert und durch eine entsprechende Struktur erreichbar sind.
In diesem Zusammenhang wird auch deutlich, weshalb sich die Hierarchie (s.o.) als Struktur für einem Hypertext besonders eignet: Durch sie kann der Benutzer seine Suche bei den groben Kategorien beginnen und Schritt für Schritt die Unterkategorien durchlaufen, bis er am Ziel angelangt ist.

Für die Orientierung in einem Hypertext ist es wichtig, dass der Benutzer jederzeit in der Lage ist, die dem „Ort-Modus-Weg“-Modell entnommenen Fragen „Wo bin ich?“, „Was kann ich hier tun?“, „Wie kam ich hierhin?“ und „Wo kann ich hin und wie komme ich dorthin?“ für sich selber zu beantworten (vgl. [Nievergelt 83, S.44]). Unterstützung bei der Beantwortung kann das Bereitstellen entsprechender Informationen, den sog. „Kontextinformationen“ (vgl. [Kaul 98, S. 12f]), auf jedem Knoten sein.
Ein Beispiel einer solchen Kontextinformation innerhalb einer hierarchischen Struktur stellt ein sog. „Pfad“ dar: Dort sind ausgehend von der Wurzel alle Knoten des Pfades zum aktuellen Knoten aufgelistet. Auf diese Weise kann der Benutzer erkennen, an welcher Stelle er sich augenblicklich befindet.
Eine wichtige Kontextinformation wird durch die „Labels“ (s.o.) vermittelt: Das Label eines Hyperlinks dient dazu, den Inhalt des Zielknotens abstrakt vorwegzunehmen. Dies geschieht, indem ein Text und/oder ein Bild („Icon“) dargestellt werden, über die der Inhalt beschrieben wird. Ein Label sollte dem Benutzer also vermitteln, was sich hinter dem entsprechenden Hyperlink für ein Inhalt verbirgt. Dadurch ist es ihm mög-

lich sich zu orientieren und abzuschätzen, ob ihn der Hyperlink seinem Ziel näher bringt.

In [Jul & Furnas 97, S. 47ff] werden drei Dimensionen einer Struktur betrachtet, nämlich „physical“, „imposed“ und „cognitive“, wobei letztere das mentale Modell eines Benutzers von der Struktur darstellt, die nicht unbedingt im Einklang mit der „echten“ Struktur (imposed bzw. physical) stehen muss. Da ein Benutzer die Struktur als Ganzes nicht unbedingt zu Gesicht bekommen wird, sondern immer nur den aktuellen Knoten inklusive der Hyperlinks, wird sich sein mentales Modell aufgrund anderer Umstände bilden. So zum Beispiel durch sein Vorwissen, oder auch durch Hinweise in Form von Kontextinformationen.
Durch derartige Entwurfsmittel kann der Aufbau eines adäquaten mentalen Modells unterstützt oder auch behindert werden. Durch entsprechende Gestaltung der Hyperlinks können selbst assoziative Hyperlinks (vgl. Kapitel 2.6) verwendet werden, ohne dass das mentale Modell Schaden nimmt. In diesem Zusammenhang hat es sich zum De-facto-Standard entwickelt, dass strukturelle Hyperlinks optisch zu einer oder mehreren Gruppen zusammengefasst werden und sog. „Navigationspanels“ bilden (vgl. [Kaul 98, S. 38f]), die über alle Knoten hinweg in ihrer Erscheinung und ihrer Funktion konsistent sind.

2.8. Kommunikationsmittel

Wie schon festgestellt wurde, fungiert eine Website als Medium für eine Kommunikation, die sich an bestimmte Benutzer richtet. Die dabei übermittelten Inhalte stammen von Menschen, wie z.B. Einzelpersonen, Organisationen oder anderen Gruppen, die in diesem Zusammenhang als Sender auftreten und dies zu einem bestimmten Zweck verfolgen.

Für die Kommunikation über eine Website stehen verschiedene Mittel zur Verfügung, die sich mithilfe von HTML realisieren lassen. Bei diesen Mitteln handelt es sich im Wesentlichen um visuelle, d.h. solche die durch das Auge des Menschen wahrgenommen werden. Auch wenn es durchaus Bemühungen gibt, die Inhalte von Websites über spezielle Browser auch akustisch zu vermitteln, so ist der überwiegende Teil der Browser doch auf das Visuelle ausgerichtet.

Ziel dieses Kapitels ist es, die wichtigsten über HTML zur Verfügung stehenden Kommunikationsmittel zu beschreiben und in Bezug auf ihre Wirkung bzw. Bedeutung zu reflektieren. Es soll ein Verständnis dafür gefördert werden, was bei der Gestaltung einer Website diesbezüglich möglich ist.

Vorweg soll jedoch noch ein Aspekt behandelt werden, der sich auf die ursprüngliche Intention von HTML bezieht: Grundsätzlich wurde HTML entworfen, um ein Mittel zur Verfügung zu haben, Inhalte strukturiert zu beschreiben und über das Internet zu verbreiten. Grundsätzlich sollten alle Aspekte der Darstellung unberücksichtigt bleiben, da man davon ausgegangen ist, dass dies grundsätzliche Aufgabe des Browsers ist. Das bedeutet, dass die Darstellung der Inhalte von Browser zu Browser unterschiedlich geleistet werden kann. In Bezug auf einen Text könnte dies bedeuten, dass dieser mit unterschiedlichen Farben und Schriftsätzen visualisiert wird.
Dieses Prinzip der Trennung von Inhalt und Darstellung (vgl. Kapitel 4.8) wurde über die verschiedenen Versionen von HTML immer weiter „aufgeweicht“ und erst wieder in jüngeren Versionen über den speziellen Mechanismus der „Cascading Style

Sheets“ (CSS)[3] verfolgt. In den folgenden Ausführungen wird dieser Aspekt nicht explizit betrachtet.

2.8.1. Text

Text ist zweifelsohne das auf Websites am häufigsten verwendete Kommunikationsmittel. Es gehört zu den auf menschlicher Sprache basierenden verbalen Kommunikationsmitteln, die allgemein bei Tätigkeiten wie Schreiben, Lesen oder Sprechen verwendetet werden.
Das Verhältnis von Sprache und Kommunikation ist allerdings nicht unbelastet, wie in [Macy, Anderson & Krygier, S. 293] festgestellt wird: „Language is fundamental error-prone, because our immensely complex human thought processes make the symbolic representation of any single idea a daunting task“. Mit anderen Worten: Das Kommunizieren eines Gedanken mittels Text sowohl in Bezug auf das Formulieren als auch auf das Lesen problematisch, weil Sprache prinzipiell missverständlich ist.
Trotz dieser Probleme hat Text als solcher seine ganz besonderen Qualitäten im Verhältnis zu anderen Kommunikationsmitteln, wie beispielsweise den Bildern. Nicht umsonst wurde diese Diplomarbeit im Wesentlichen als Text verfasst, anstatt über Bilder oder andere Mittel.

Text besteht aus Elementen wie Buchstaben, Worten, Sätzen, Absätzen usw., die ihm eine Struktur und damit auch eine Bedeutung geben. Beispielsweise signalisiert ein Absatz, dass es sich dabei um einen abgeschlossenen Gedanken handelt.
Bei der Darstellung von Text kommen Aspekte der Typographie zum Tragen, z.B. durch Auswahl eines Fonts, der Festlegung von Zeilenabstand usw. Neben Auswirkungen in Richtung Lesbarkeit hat Typographie auch die Möglichkeit gezielt die Aussage von Text mit zu unterstützen. Beispielsweise wird ein einzelnes fett dargestelltes Wort dem Text einen besonderen Akzent verleihen.

Obwohl das Schreiben von Texten zumindest in unserem Kulturkreis als natürlich erscheint und oft ohne Probleme durchgeführt werden kann, so ist das Schreiben von Texten für spezifische Zwecke problematischer. Das Schreiben einer Diplomarbeit verlangt ein anderes Vorgehen, als beispielsweise das Verfassen eines Briefes. Dies bezieht sich nicht ausschließlich auf fachliches Wissen, sondern auch auf den Schreibstil. Dieser muss der Aufgabe angemessen sein, was sich in dieser Diplomarbeit durch einen sachlichen Stil äußert. Nicht anders verhält es sich bei Websites: Hier muss z.B. genauso auf die Sprache der Zielgruppe Rücksicht genommen werden, wie auch auf die Besonderheiten des WWW. In einer Studie zum Leseverhalten im WWW haben [Morkes & Nielsen 97] u.a. herausgefunden, dass Texte weniger gelesen, als vielmehr „gescannt“ werden. Das bedeutet, dass nur ein oberflächliches Überfliegen von Texten stattfindet, bei dem einzelne Worte oder Sätze herausgepickt werden. Neben soziologischen Aspekten macht Nielsen auch den Umstand der schlechten Lesbarkeit von Text am Bildschirm dafür verantwortlich [Nielsen 97a]. Daraus resultiert, dass beim Schreiben von Texten für Websites auf diese Umstände eingegangen werden muss.

Text wird in HTML nahezu unmittelbar verwendet, d.h. er kann direkt in das HTML-Dokument eingeben werden und muss nur über spezielle Auszeichnungen strukturiert werden.

[3] http://www.w3.org/Style/CSS/ (10.10.2000)

2.8.2. Bilder

Neben dem Text stellen Bilder das zweite wichtige Kommunikationsmittel im Kontext von Websites dar. Während Text sequentiell, d.h. Wort für Wort, gelesen werden muss, ist der Zugang zu Bildern unmittelbarer. Untersuchungen der Wahrnehmungspsychologie haben herausgefunden, dass Bilder im Verhältnis zu Text die Aufmerksamkeit schneller auf sich ziehen. Dies wird damit erklärt, dass Bilder und Farben für den Betrachter „attraktiver“ erscheinen, als das Schwarz-Weiß von Text auf seinem Untergrund [Schneider & Raue 98, S. 153].
Aufgrund ihrer langen Tradition in der Menschheitsgeschichte (Stichwort: Höhlenmalerei) wird ihnen eine bestimmte Faszination und Bedeutung zugeschrieben (vgl. [Strothotte & Strothotte 97, S. 5f]), die sich z.B. über Redensarten wie „Bilder lügen nicht“ oder „Bilder sagen mehr als tausend Worte“ äußert.
Eine Unterteilung von Bildern in „Presentational Pictures“, „Abstract-Graphical Pictures“ und „Pictograms“ wird in [Strothotte & Strothotte 97, S. 43ff] vorgenommen, wobei jede dieser Arten in Bezug auf ihren Abstraktionsgrad und ihren Zweck unterschiedliche Aussagekraft und damit Bedeutung haben.

Im Bereich des WWW gelten Bilder als nicht ganz unproblematisch: Im Verhältnis zu Texten sind Bilddateien deutlich größer und sorgen somit für verhältnismäßig viel längere Übertragungszeiten. Auch wenn sich Probleme dieser Art durch die Aufstockung der Bandbreite des WWW relativieren lassen, so stellen sie dennoch einen qualitativen Faktor dar. Zudem sind Bilder durch ihre oftmals unreflektierte und exzessive Verwendung im WWW teilweise in Verruf geraten [Fleming 97].

2.8.3. Layout

Ein Layout beschreibt die Größe und Position einzelner Elemente, wie z.B. Textblöcke oder Bilder, auf einer Fläche. Dadurch werden diese Elemente in Beziehung gesetzt und bilden eine visuelle Hierarchie, mit der etwas über den Stellenwert der jeweiligen Elemente ausgesagt wird [Fleming 98, S. 64ff]. Weiter oben stehende oder auch größere Elemente werden beispielsweise vom Menschen als wichtiger wahrgenommen, als entsprechend tiefer platzierte und kleinere.

Im Zusammenhang mit Layout spielen auch Faktoren wie das Format sowie das Raster eine Rolle. Das Format beschreibt die Länge und das Verhältnis der Kanten einer Seite zueinander. Die Wahl des Formats hat laut [Duschek 89, S. 113] Auswirkungen auf die Ästhetik einer Seite, für die individuell ein Ideal gefunden werden kann. Ein Raster hingegen gibt auf grobe Art und Weise zulässige Positionen als Basis für das Layout vor und sorgt für Vereinheitlichung und Wiederverwendbarkeit [Duschek 89, S. 126ff].

Ebenso wie Typographie zählt auch Layout eher zu den Mitteln der Darstellung und wurde bei der Konzeption von HTML nicht berücksichtigt. Seitdem aber Tabellen in HTML eingeführt wurden, werden diese zu Layoutzwecken „missbraucht“. In jüngeren HTML-Versionen wird Layout aber inzwischen auch direkt unterstützt.

2.8.4. Farben

Auch wenn es auf den ersten Blick nicht offensichtlich erscheint, so haben selbst Farben verschiedene Bedeutungen und stellen somit ein Kommunikationsmittel dar. Als Beispiel sollen bestimmte Verkehrszeichen gelten, deren Wichtigkeit über die Farbe „Rot“ signalisiert wird.

Man spricht explizit von einer „Dynamik der Farben“, wenn sich zwischen einer Farbe und einem menschlichen Verhalten ein Zusammenhang herstellen lässt [Seitz 89, S. 158]. So beispielweise, wenn es bei anhaltendem rotem Licht zu emotionalen Auswirkungen wie Erregungszuständen kommt.

Neben der kulturbedingten Belegung von Farben mit bestimmten Bedeutungen (vgl. [Strothotte & Strothotte 97, S. 212]) können sie untereinander auch visuelle Hierarchien bilden [Fleming 98, S. 64ff], in denen bestimmte Farben deutlicher wahrgenommen werden als andere. So werden beispielsweise visuelle Elemente, deren Farben eine starke Sättigung aufweisen, gegenüber den eher blassen Elementen hervorgehoben. Ebenso verhält es sich mit Helligkeit, über deren Variation eine Beziehung zwischen den Elementen ausgedrückt werden kann.

In HTML lassen sich Farben zwar fast beliebig genau definieren, wobei aber insbesondere dort eine starke Abhängigkeit zum Endgerät des Benutzers besteht. Beispielsweise führt die Verwendung unterschiedlicher Grafikkarten mitunter zu unterschiedlichen Darstellungen der Farben auf dem Monitor.

2.8.5. Zusammenspiel der Kommunikationsmittel

In den vorherigen Kapiteln wurden verschiedene Kommunikationsmittel vorgestellt, die in ihrer Wirkung und Bedeutung unterschiedlich relevant sind. So lassen sich beispielsweise mit Texten und Bildern konkrete Inhalte gut transportieren, während mit Layout und Farbe eher Akzente gesetzt werden.
Neben der direkten Inhaltsvermittlung können mit Mitteln der Kommunikation auch Emotionen ausgelöst werden. In diesem Zusammenhang wird in [Franke 93, S. 108] der Zusammenhang zwischen Ästhetik und emotionalen Reaktionen erwähnt.

Keines dieser Mittel kann das andere adäquat ersetzen: Die Möglichkeiten, die beispielsweise durch den Einsatz von Bildern entstehen, unterscheiden sich grundlegend von den Möglichkeiten von Texten. Insofern ergänzen sich die Komponenten und führen im Zusammenspiel zu einer Gesamtwahrnehmung beim Empfänger der Kommunikation. Andersherum kann die Erstellung der einzelnen Komponenten nicht unabhängig von den anderen verlaufen.

3. Benutzerorientierung

Wie oben schon angedeutet wurde, kann man in Websites ein zweckgebundenes Kommunikationsmittel sehen, dessen Zweck in der Übermittlung ganz bestimmter Informationen liegt. Die Empfänger dieser Informationen stellen die sog. „Benutzer" dar, die aus bestimmten Gründen die Website besuchen. Auch wenn es sich bei diesen Gründen um eher unspezifische Interessen handelt, so soll in diesem Zusammenhang davon ausgegangen werden, dass die Motivation in einem bestimmten „Bedürfnis" zu suchen ist. In Bezug auf ein Informationssystem wird ein derartiges Bedürfnis in dem Auffinden einer bestimmten Information zu suchen sein. So gesehen kann eine Website als ein Medium gesehen werden, mit dem Benutzer ihre Bedürfnisse bzw. Aufgaben erfüllen.

Sofern ein Benutzer eine derartige Information auf einer Website findet, wird der Besuch aus seiner Perspektive erfolgreich gewesen sein, während es andersherum einen Misserfolg darstellt. Ein derartiger Misserfolg tritt dann ein, wenn die gesuchte Information auf der Website entweder nicht vorhanden ist, oder das System so unzureichend gestaltet ist, dass die Navigation fehl schlägt. In so einem Fall ist nicht unbedingt davon auszugehen, dass ein Benutzer die Website ein weiteres Mal besuchen wird.
In Bezug auf die Zweckgebundenheit einer Website muss also als allgemeiner Misserfolg gewertet werden, wenn die Bedürfnisse der Benutzer nicht erfüllt werden und sie somit die Website verlassen. Da die Entwicklung und der Betrieb einer Website direkt oder indirekt mit Kosten verbunden ist, ergeben sich aus einer „unbenutzbaren" Website finanzielle Verluste (vgl. [Nielsen 98c] und [Kalin 99]), von Verlusten in Bezug auf das Ansehen ganz zu schweigen. Um derartiges zu verhindern, ist es notwendig, die Entwicklung einer Website an den Bedürfnissen der Benutzer zu orientieren und so zu gestalten, dass die Benutzer ihre Bedürfnisse erfüllen können.

Menschen unterscheiden sich in hohem Maße untereinander in Bezug auf ihre jeweils individuellen Ziele, Motivationen und Aufgaben (vgl. [Shneiderman 98, S. 18ff] und [Sisson 99]), wodurch sich automatisch verschiedenste Bedürfnisse ergeben. Insofern wird das Medium „Website" nicht universell einsetzbar sein, sondern wird sich auf einen vergleichsweise kleinen Teilaspekt aller möglichen Bedürfnisse und Aufgaben beziehen müssen. Daher muss für den Entwurf eine genaue Betrachtung der jeweiligen Umstände bzw. der Bedürfnisse stattfinden, wobei es gefährlich wäre, ausschließlich von Mutmaßungen auszugehen. Stattdessen sollte, entsprechend dem Prinzip „Know the User", ein intensives Fokussieren der Benutzer stattfinden [Shneiderman 98, S. 67]. Ein derartiges Vorgehen wird in der englischsprachigen Literatur als „User-Centered Design" (UCD) bezeichnet [Tenny 96], wobei in dieser Diplomarbeit der deutsche Begriff „Benutzerorientierung" verwendet werden soll.

Als wesentliche Merkmale eines benutzerorientierten Entwurfsprozesses werden von [Gould, Boies & Lewis 91, S. 75] folgende Merkmale genannt:

- *Frühzeitiger Augenmerk auf den Benutzer*: Mittels direktem Kontakt mit Benutzern soll ein Verständnis bezüglich deren Kognition, Verhalten, Einstellungen und Anthropologie erlangt werden.
- *Integrierter Entwurf*: Alle Aspekte der Benutzbarkeit sollten unter gemeinsamer Leitung parallel entwickelt werden

- *Frühzeitige und andauernde Benutzertests*: Auf empirischer Basis sollen Tests mit Benutzern durchgeführt werden, deren Ergebnisse als Grundlage für Entwurfsänderungen dienen.
- *Iterativer Entwurf*: Über einen Zyklus von Implementation, Tests, Feedback, Evaluation und Änderungen wird das entwickelte System schrittweise verbessert.

Aus den Disziplinen der Software-Ergonomie bzw. des Software Engineering sind eine Reihe von Methoden bekannt, die konform mit dem Gedanken der Benutzerorientierung gehen und einen Entwicklungsprozess mit obigen Merkmalen unterstützen. Obwohl an dieser Stelle das Verhältnis zwischen Software und Websites noch nicht geklärt ist (siehe dazu Kapitel 4.2) sollen derartige Methoden dennoch schon an dieser Stelle vorgestellt werden.

3.1. Partizipation

Eine Ausprägung der Benutzerorientierung ist durch die sog. „Partizipation“ gegeben. Im Kontext der Softwareentwicklung wird mit diesem Begriff „[...] die Beteiligung der Arbeitnehmer und ihrer Interessenvertreter an der Planung, Gestaltung und Realisierung der Innovationsmaßnahmen.“ Beschrieben können [Reisin 94, S. 299f]. Das bedeutet, dass Benutzer in den Prozess der Softwareentwicklung direkt mit einbezogen werden, damit sie das Ergebnis im Sinne einer Qualitätssicherung positiv beeinflussen.
In einem Softwareentwicklungsprojekt, das ohne Partizipation durchgeführt wird, müssen Entwurfsentscheidungen ausschließlich von Entwicklern bzw. anderen Entscheidungsträgern getroffen werden. Solchen Menschen fällt es häufig schwer, sich in die Lage von gewöhnlichen Benutzern hineinzuversetzen und somit deren Bedürfnisse zu erfassen. Der Grund liegt darin, dass sie entweder nicht im entsprechenden Anwendungskontext beheimatet sind, oder aber - im Gegenteil - sich schon zu tief mit der Materie befasst haben. Im letzteren Fall spricht man von einer „disease of familiarity“ [Fleming 98, S. 35].
Insofern werden Entwurfsentscheidungen dann auf Basis von vermeintlichen Erfahrungen oder eigenen Bedürfnissen gefällt, die wahrscheinlich nicht der Realität der Benutzer entsprechen. Insofern stellt Partizipation die einzige Möglichkeit dar, überhaupt repräsentative Erkenntnisse bezüglich eines Anwendungskontextes zu erlangen.

Nicht jede Art von Software eignet sich dafür, in einem partizipativen Prozess erstellt zu werden, sondern es handelt sich natürlicherweise um solche, die von Benutzern unmittelbar im Rahmen von Arbeitstätigkeiten benutzt werden [Reisin 94, S. 303], wie z.B. interaktive Anwendungssoftware und insbesondere auch Websites.

Ohne die Methoden im Einzelnen schon vorzustellen und damit Kapitel 3.4 vorzugreifen, sollen hier schon Beispiele für die Möglichkeiten der Partizipation genannt werden. Zum einen können Benutzer allgemein um ihre Meinung bezüglich bestimmter Sachverhalte befragt werden, wie z.B. nach erwünschten Funktionen innerhalb einer Software. Zum anderen ist es auch gängig, Benutzer mit frühen Versionen der Software arbeiten zu lassen, um herauszufinden an welchen Stellen Schwierigkeiten auftreten.
In allen Fällen läuft es darauf hinaus, dass über die Partizipation Erkenntnisse bezüglich der Problemstellung gewonnen werden. Allerdings reicht es nicht aus, diese Erkenntnisse nur zu gewinnen, sondern sie müssen konsequenterweise auch in die

Entwicklung integriert werden. Dabei ist es unerlässlich, dass dies vom jeweiligen Prozessmodell unterstützt wird: „Andernfalls bleibt die Beteiligung der Benutzer am Projekt nominell und erschöpft sich in unverbindlichen Absichtserklärungen“ [Reisin 94, S. 312]. Prinzipiell muss ein entsprechendes Prozessmodell über die Möglichkeit von Rückkoppelungen verfügen, bei denen Änderungen in Bezug auf partizipationsbedingte Erkenntnisse umgesetzt werden können. Die Behandlung eines entsprechenden Prozessmodells erfolgt in Kapitel 4.4.1.

Shneiderman sieht die Anwendung von Partizipation nicht gänzlich unkritisch: Zwar bestehen Vorteile, wie z.B. die Möglichkeit, genauere Informationen über den Benutzer zu erhalten, oder auch die gesteigerte Akzeptanz bei der Einführung des Ergebnisses. Als Nachteile ergeben sich neben höheren Kosten und längeren Entwicklungszeiten auch soziale Probleme, wie z.B. der Notwendigkeit u.U. Vorschläge von Benutzerseite unberücksichtigt zu lassen [Shneiderman 98, S. 109f].

3.2. Evaluation

Evaluation („Bewertung“) bedeutet im Rahmen von Softwareentwicklung „[...] inwieweit die gestellten Anforderungen auch tatsächlich im Entwicklungsprozess umgesetzt wurden“ [Oppermann & Reiterer 94, S. 336]. Damit ist gemeint, dass Ergebnisse oder Teilergebnisse aus dem Entwicklungsprozess in Hinblick auf die Zielsetzung begutachtet und reflektiert werden.

Eine Evaluation kann von unterschiedlichen Personengruppen vorgenommen werden, die jeweils eine gänzlich eigene Sicht auf das System haben [Weinreich 97, S. 81ff]: Zum einen können die Entwickler ihr System selber begutachten, wobei dies aufgrund mangelnder Objektivität nicht unproblematisch ist. Eine weitere Gruppe stellen die sogenannten „Experten“ dar, die über ihre Erfahrung und häufig auch mittels Leitfäden viele Probleme im System finden können. Neben ihrem speziellen Wissen ist es auch deren Objektivität, die für gute Ergebnisse sorgt.
Zuguterletzt stellen insbesondere die Benutzer eine wichtige Gruppe von Personen dar, die ein System evaluieren können. Dadurch, dass es sich um einen partizipativen Vorgang (s.o.) handelt, hat die Evaluation mit Benutzern einen besonderen Stellenwert und soll in dieser Diplomarbeit besonders ausführlich behandelt werden. In der gängigen Literatur wird diese Art der Evaluation auch als „User Tests“ (vgl. z.B. [Nielsen 94]) bezeichnet, während in dieser Diplomarbeit von „Benutzertests“ gesprochen werden soll.

In [Oppermann & Reiterer 94, S. 342ff] wird zwischen einer „subjektiven“ und einer „objektiven“ Evaluation unterschieden: Bei ersterer handelt es sich um Benutzertests, in denen die Benutzer ihre Gedanken direkt mitteilen, z.B. durch Fragen und Aussagen. Auf diese Weise lassen sich vergleichsweise unstrukturiert Probleme aufdecken und Hinweise zur Akzeptanz des System bekommen. Allerdings besteht die Gefahr, dass diese Evaluation durch ihren subjektiven Charakter unrepräsentativ wird. Insbesondere lassen sich Diskrepanzen zwischen den Gedanken und den Aussagen der Benutzer nicht nachweisen. Im Gegensatz dazu werden bei der „Objektiven Evaluation“ nach Möglichkeit alle subjektiven Komponenten ausgeklammert und Daten nur durch Methoden wie z.B. das „Beobachten“ gewonnen.

3.3. Prototyping

In diesem Abschnitt soll das sog. „Prototyping“ als ein entscheidendes Vorgehen in Bezug auf die Benutzerorientierung vorgestellt werden. Gegenstand dieses Vorge-

hens ist ein sog. „Prototyp“, der von Nielsen als eine Implementation der sich in der Entwicklung befindlichen Software gesehen wird, deren Komplexität reduziert ist [Nielsen 94]. In ähnlicher Weise wird dies von [Züllighoven et al. 98, S. 637f] definiert: „Ein Prototyp ist ein ablauffähiges Modell ausgewählter Aspekte des zukünftigen Anwendungssystems“. Mit einem Prototypen liegt ein Medium vor, mit dem experimentell Erfahrungen in Hinblick auf bestimmte Probleme des Entwurfsgegenstandes erlangt werden können. Auf diese Weise kann er z.B. als Diskussions- und Entscheidungsgrundlage dienen.

Gemäß der jeweils verfolgten Zielsetzung unterscheidet man zwischen explorativem, experimentellem und evolutionärem Prototyping [Floyd 84, S. 6ff]: Ein exploratives Prototyping wird bei der Systemspezifikation eingesetzt, indem Vorstellungen und Ideen implementiert und auf Brauchbarkeit überprüft werden. Auf diese Weise kann das System frühzeitig kennen gelernt und dessen Funktionalität ermittelt werden.
Beim experimentellen Prototyping hingegen liegt der Schwerpunkt eher auf der Überprüfung von Machbarkeit und Zweckmäßigkeit einzelner Teilkomponenten, die mittels eines Prototypen implementiert und überprüft werden.
Beim evolutionären Prototyping verschwimmen die Grenzen zwischen dem Endprodukt und den Prototypen, indem ein Zyklus von Prototypentwicklung und deren Bewertung durchgeführt wird, der eine inkrementelle Systementwicklung zum Ziel hat. In diesem Zusammenhang ist das Konzept der sog. wiederverwendbaren Prototypen interessant, bei denen einige Komponenten in den nächsten Prototypen bzw. in das Endprodukt übernommen werden. Demgegenüber werden die „Wegwerfprototypen“ unterschieden, die nach der Erfüllung ihres Zwecks in keiner Weise weiterverwertet werden.

Zwischen Prototypen und der Evaluation besteht ein enger Zusammenhang, weil Prototypen ein Medium darstellen, mit dem eine Evaluation durchgeführt werden kann. Durch ihren Charakter, nur einen Teilaspekt des Gesamtsystems zu repräsentieren, besteht grundsätzlich die Möglichkeit, derartige Medien schnell und billig realisieren zu können. Dies hat die Konsequenz, dass Evaluationen schon frühzeitig, nämlich vor der Fertigstellung der Software, durchgeführt werden können. Wie oben schon ausgeführt wurde, ist dies in Hinblick auf ein iteratives Vorgehen im Rahmen der Benutzerorientierung von entscheidender Bedeutung. So können rechtzeitig Probleme bzw. Entwurfsfehler erkannt werden, so dass diese noch im Entwicklungsprozess berücksichtigt werden können. Zwar können Entwurfsentscheidungen grundsätzlich auch auf anderem Wege vermittelt und bewertet werden, wozu aber [Reisin 94, S. 320] feststellt: „Der Vorteil der Prototypingmethoden besteht darin, dass der Umgang mit Prototypen konkrete sinnliche Erfahrungen vermittelt, die beim bloßen Lesen von textuellen oder grafischen Beschreibungen nicht erreicht werden können“.

Uneinigkeit herrscht laut [Pomberger & Blaschek 96, S. 3] darüber, ob die Ausführbarkeit von Prototypen definitionsgemäß direkt gegeben sein sollte oder ob auch eine Simulation der Ausführbarkeit ausreicht.
Während die direkte Ausführbarkeit nur über eine Software zu erreichen ist, würde die indirekte auch Mittel wie „Mockups“ mit einschließen. Es handelt sich dabei um einfache gegenständliche Modelle, z.B. in Form von auf Karton geklebten „Papier-Screens“, die einen Bildschirm bzw. einen Computer repräsentieren. Aktionen von Benutzerseite werden verbal veranlasst und von einer Person manuell z.B. durch Austauschen der „Screens“ durchgeführt. Der Vorteil gegenüber der Erstellung von

Prototypen als Software liegt in der extrem kurzen Zeit, mit der sie erzeugt werden können. Gerade in einer Umgebung, in der nur wenig Zeit zur Verfügung steht, stellen sie u.U. das einzige adäquate Mittel zum Prototyping dar (vgl. [Snyder 96]). In dieser Diplomarbeit soll der Einfachheit halber das Mittel der „Mockups“ ebenfalls unter dem Oberbegriff der „Prototypen“ geführt werden.

Entsprechend ihres Einsatzes wird die Qualität und der Umfang des jeweiligen Prototypen variieren. In diesem Zusammenhang ist die von Nielsen verwendete Klassifikation hilfreich, bei der Prototypen in vertikale und in horizontale unterteilt werden. Während bei den horizontalen Prototypen der Schwerpunkt auf der Darstellung möglichst vieler Funktionen liegt und somit eher die grafische Benutzungsoberfläche repräsentiert wird, so ist es bei vertikalen Prototypen eher die Ausarbeitung der Funktionalität einzelner Features. So gesehen stellen Mockups Prototypen dar, die sowohl horizontal als auch vertikal extrem reduziert sind [Nielsen 94].

Für das Erstellen von Prototypen wird häufig bestehende Software zweckentfremdet eingesetzt, wobei je nach Kontext unterschiedliche Werkzeuge verwendet werden können. Dabei sollte es einleuchtend sein, dass z.B. bei horizontalen Prototypen eher solche verwendet werden können, in denen grafische Kapazitäten beinhaltet sind, als bei vertikalen, in denen interaktive Elemente mittels einer Programmiersprache implementiert werden müssen.

3.4. Benutzerorientierung im WWW

Viele Methoden aus dem Umfeld der Benutzerorientierung sind in Bezug auf Ressourcen wie Zeit und Geld recht aufwendig, was mitunter zu einem Verzicht eines solchen Vorgehens führt. Demgegenüber gibt es aber Ansätze, bei denen nur extrem preisgünstige bzw. wenig zeitintensive Methoden verwendet werden („Discount Usability Engineering“), die aber ihrerseits den Großteil aller Benutzbarkeitsprobleme aufdecken [Nielsen 94]. Das Aufzeigen derartiger Methoden in Bezug auf das WWW soll das Ziel dieses Abschnittes sein.

Aus der einschlägigen Literatur sind eine Reihe von Methoden bekannt, die als eine konkrete Anwendung der oben allgemein geschilderten Methoden in Bezug auf das WWW gesehen werden können. Sie erscheinen in Bezug auf knappe finanzielle und zeitliche Ressourcen für die Anwendung in dem hier behandelten Entwurfsprozess als geeignet. Derartige Methoden sollen in diesem Kapitel grundsätzlich vorgestellt und diskutiert werden, so dass sie innerhalb eines Entwicklungsprozesses gemäß der Kapitel 6 und 7 angewandt werden können.

Es kann davon ausgegangen werden, dass die bei der Entwicklung von Websites beteiligten Personen Zugriff auf das WWW und andere Dienste des Internet haben. Durch den Charakter dieser Dienste, weltweit zugänglich zu sein, entstehen gegenüber „herkömmlichen“ Methoden gänzlich neue Möglichkeiten, was beispielsweise die Kommunikation dieser Personen untereinander entspricht. Es ist z.B. nicht unbedingt mehr notwendig, dass bestimmte Personen physikalisch oder zu bestimmten Zeiten anwesend sind. Durch Mittel der Automatisierung ist es zudem möglich, Zeit bei der Auswertung einzusparen. Insgesamt kann der Einsatz entsprechender Mittel zu schnelleren und kostengünstigeren Ergebnissen führen.

3.4.1. Vorbereitung von Benutzertests

Damit Benutzertests durchgeführt werden können, ist es offensichtlich nötig, dass entsprechende Teilnehmer gefunden und rekrutiert werden. Hier sollen in Anlehnung an [Fleming 98b] Hinweise gegeben werden, wie und wo auf geeignete Personen zugegriffen werden kann.

Zunächst einmal besteht die Möglichkeit, Personen aus dem direkten eigenen Umfeld zu rekrutieren. Dazu gehören Freunde, Verwandte, Nachbarn, Kollegen, usw. Obwohl es sich dabei um die einfachste und möglicherweise die schnellste Möglichkeit handelt, ist dies nicht unproblematisch. Gerade, wenn es um die Bewertung der eigenen Arbeit geht, kann man von nahestehenden Personen nicht bedingungslose Ehrlichkeit erwarten, weil diese u.U. keine Gefühle verletzen möchten. Aufgrund dieses vermeintlich rücksichtsvollen Verhalten wird mit Sicherheit nicht der volle Grad der Effektivität erreicht. Insofern sollte eher nach Personen gesucht werden, zu denen kein direkter Bezug besteht und von denen man grundsätzlich auch unangenehme Wahrheiten zu hören bekommt.

Grundsätzlich ist es sinnvoll Personen zu rekrutieren, die mit der jeweiligen Materie vertraut sind. Zum einen sollten Erfahrungen mit dem WWW vorhanden sein und zum anderen mit der jeweiligen Thematik der konkreten Website. Insbesondere sollte dabei möglichst auf „potentielle Benutzer“ zurückgegriffen werden, was für das Bestimmen von konkreten Problemen mit einem System unerlässlich ist [Nielsen 96b]. Beispiel: Wenn die Website für Kinder gedacht ist, dann sollten auch Kinder die Testpersonen stellen (vgl. [Instone 97]).

Je nachdem wer diese potentiellen Benutzer sind, bestehen unterschiedliche Möglichkeiten, mit diesen in Kontakt zu treten. Dabei kann insbesondere auf bestehende Kontakte, z.B. in Form einer Kundenkartei, zugegriffen werden. Bei einer schon bestehenden Internetinfrastruktur, in Form von Diskussionsforen, registrierten Benutzern oder Kontakt über e-Mail, bietet sich die Verwendung dieser Ressourcen an. Unter Umständen besteht auch die Möglichkeit, über externe Dienste Zugang zu solchen Personen zu erlangen, z.B. in Form eines öffentlichen Diskussionsforums auf einer fremden Website oder aber einer Newsgroup. In allen Fällen sollte darauf geachtet werden, dass es sich um „Orte“ handelt, in denen sich potentielle Benutzer aufhalten könnten. Dies gilt nicht nur für virtuelle Orte, sondern auch für physikalische [Fuccella & Pizzolato 98], wie z.B. (in Anlehnung an obiges Beispiel) ein Kindergarten.

Neben dieser aktiven Rekrutierung besteht auch noch die Möglichkeit einer passiven, wobei insbesondere die sog. „Banner“ im WWW interessant sind: Es handelt sich dabei um animierte Grafiken, die auf verschiedenen Websites platziert werden können. Sie dienen als „Werbung“ und können mit einem Hyperlink ausgestattet werden, so dass die Betrachter auf bestimmte Dokumente im WWW gelenkt werden können. Dieser Mechanismus kann genutzt werden, indem Benutzer auf bestimmte „Rekrutierungsseiten“ geführt werden, wobei in [Fuccella & Pizzolato 98] allerdings vor der Verwendung dieser Möglichkeit gewarnt wird: Es besteht die Gefahr, dass zu viele „themenfremde“ Benutzer teilnehmen und somit für unrepräsentative Ergebnisse sorgen.

Moderator
Sofern ein direkter Umgang mit Benutzern stattfindet, muss mindestens eine Person zugegen sein, die eine leitende Rolle einnimmt und für den Kontakt zu den Teilnehmern zuständig ist. Zudem müssen die Ergebnisse des Kontaktes aufgezeichnet werden. Eine solche Person soll im Folgenden der Einfachheit halber allgemein als „Moderator“ bezeichnet werden, auch wenn dieser Begriff in manchen Situationen nicht besonders passend erscheinen mag.

Zur Auswahl und Einsatz eines solchen Moderators werden im Folgenden zunächst einige allgemeine Umstände diskutiert: Es sollte einleuchten, dass ein Moderator nach Möglichkeit objektiv und zurückhaltend sein muss, um die Teilnehmer nicht zu beeinflussen und damit die Ergebnisse zu verfremden. Aus diesem Grund ist es problematisch, wenn die Entwickler selber diese Rolle übernehmen. Aufgrund der fehlenden Distanz zu ihrer Entwicklung werden sie in so einem Fall häufig beeinflussend tätig und versuchen ihre Entscheidungen zu rechtfertigen [Fleming 98b]. In Situationen, in denen die unmittelbare Anwesenheit des Moderators nicht unbedingt notwendig ist, sollte er sich deshalb auch aus dem sichtbaren Umfeld der Benutzer entfernen, um somit die Wahrscheinlichkeit einer Beeinflussung zu verringern [Wildman 95]. Dies wird jedoch in vielen Situationen nicht möglich sein, insbesondere dann nicht, wenn es ihre Aufgabe ist, z.B. eine Diskussion zu leiten oder konkrete Fragen zu stellen.
Terry Sullivan weist darauf hin, dass Benutzer dazu tendieren, bei Problemen die Fehler bei sich selber zu suchen und deshalb aus Scham bestimmte Aspekte nicht ansprechen. Da dies natürlich dem Zweck der jeweiligen Methode widerspricht, sollten die Teilnehmer vom Moderator explizit ermutigt werden, wirklich alles auszusprechen, was in dem jeweiligen Zusammenhang relevant sein könnte [Sullivan 96a].

3.4.2. Umfragen, Interviews und Diskussionen

Bei Umfragen, Interviews und Diskussionen werden Benutzer direkt dazu aufgefordert, ihre Meinungen und Vorstellungen bezüglich bestimmter Fragestellungen kundzutun. Die Ergebnisse werden entsprechend subjektiv ausfallen und sind daher entsprechend zu bewerten (vgl. Kapitel 3.2).

Umfragen
Umfragen sind ein häufig verwendetes Mittel, wenn es darum geht, kostengünstig eine große Anzahl von Benutzern zu befragen. Üblicherweise wird dazu ein Fragebogen mit einer Reihe von Fragen ausgearbeitet, der dann den Benutzern zugänglich gemacht wird.
Während dies klassischerweise über das Medium „Papier“ stattfindet, bietet das WWW die Möglichkeit zur elektronischen Durchführung. Während einerseits auf das simple Mittel der e-Mail zurückgegriffen werden kann, besteht andererseits die komfortablere Variante in der Bereitstellung eines Formulars in HTML: Auf einer HTML-Seite werden dazu Formularelemente (z.B. Textfelder) mit einem „Abschicken“-Button kombiniert. Bei Betätigung dieses Buttons werden die eingegebenen Daten vom Webbrowser an eine Applikation übermittelt, die die Daten speichern oder sogar auswerteten kann.
Das technische Erstellen von Umfragen dieser Art findet durch kommerzielle Komplettangebote[4] oder durch frei verfügbare Standardlösungen[5] eine breite Unterstüt-

[4] z.B. als webbasierte Lösung unter http://www.surveybuilder.com/ (24.10.2000)
[5] z.B. „FormMail“, http://www.worldwidemart.com/scripts/formmail.shtml (10.10.2000)

zung. Es lassen sich damit automatisiert eine große Menge an Einzelbefragungen durchführen.

Die Auswahl der zu stellenden Fragen ist sorgfältig vorzunehmen: [Fuccella & Pizzolato 98] weisen darauf hin, dass dabei eine Balance zwischen einerseits der Menge der erwünschten Antworten und andererseits der Motivation der Testpersonen zur Beantwortung der Fragen gefunden werden muss. Zum einen ist zu bedenken, dass sich u.U. keine zweite Möglichkeit zur Initiierung einer Umfrage ergibt, aber zum anderen Benutzer nicht gewillt sind, ihre Zeit und Geduld bei der Ausfüllung des Fragebogens übermäßig strapazieren zu lassen. Die Konsequenzen könnten unvollständig ausgefüllte oder ignorierte Fragebögen sein. Als ein guter Wert werden von [Sullivan 96a] 6-10 Fragen genannt, die einem Benutzer zuzumuten sind.

Die Auswahl konkreter Fragen ist natürlich vom jeweiligen Einsatzkontext abhängig und wird an gegebener Stelle in dieser Arbeit konkreter behandelt. Grundsätzlich ist das Formulieren von Fragen gerade in Hinblick auf die oftmals nicht gegebene Möglichkeit der Rückfrage ein kritischer Faktor für den Erfolg einer Umfrage. Dies hängt beispielsweise von der Eigenschaft der menschlichen Sprache ab, mehrdeutig und missverständlich zu sein (vgl. Kapitel 2.8.1). Zur Bekämpfung dieser Umstände haben sich in Bereichen wie z.B. der Psychologie eine Reihe von Methoden bewährt, mit denen Fragebögen optimiert werden können. Eine Behandlung solcher Methoden soll an dieser Stelle nicht geleistet werden, sondern z.B. auf [Weinreich 97, S. 95] verwiesen werden.

Von [Fuccella, Pizzolato & Franks 99] werden Umfragen in „explorativ“ und „iterativ“ unterschieden. Während die explorativen Umfragen in der Regel nur einen Durchlauf haben, sind es bei der iterativen Variante mehrere, wobei zunehmend detailliertere Fragen gestellt werden. In jedem Durchlauf bauen die Fragen auf den Antworten des vorhergehenden auf, so dass sich schrittweise einer spezielleren Fragestellung angenähert werden kann. Während beispielsweise in einem ersten Schritt Ideen zu einem bestimmten Thema gesammelt werden, können deren Gesamtergebnisse in einem zweiten Schritt auf einer Skala nach Wichtigkeit bewertet werden. Durch die mehrfache Erstellung eines Fragebogens und Auswertung der Antworten wird natürlich verhältnismäßig viel mehr Zeit zur Durchführung und Geduld der Teilnehmer beansprucht.

Interviews

Bei Interviews handelt es sich typischerweise um eine Sitzung, in denen ein Befragter und ein Interviewer („Moderator“) anwesend sind. Vom Interviewer werden vorbereitete Fragen gestellt, die dann vom Teilnehmer beantwortet werden. Auf diese Weise kann ein Gespräch entstehen, in dem beiderseitig Rückfragen stattfinden können. Der Teilnehmer kann Verständnisprobleme in Bezug auf die Fragen äußern, während der Moderator bei interessanten Aspekten genauer nachfragen kann. So ergibt sich eine hohe Produktivität, die gegenüber anderen Methoden eine besondere Qualität hat [Shneiderman 98, S. 145]. Allerdings besteht durch die naturgemäß vergleichsweise geringere Anzahl von teilnehmenden Personen potentiell die Gefahr, dass die Ergebnisse unrepräsentativ sind.
Dadurch, dass es schwierig für den Moderator ist, gleichzeitig Fragen zu stellen, zuzuhören und die Antworten festzuhalten, bieten sich Hilfsmittel wie beispielsweise Videokamera oder Kassettenrekorder an, mit dem sich der Gesprächsverlauf aufzeichnen und zeitversetzt in einem späteren Schritt auswerten lässt.

Da pro Interviewer zeitgleich immer nur ein Teilnehmer befragt werden kann und Gespräche auch länger andauern können, wird ein Interview insgesamt zu einer zeitintensiven und damit teueren Angelegenheit.

Diskussionsgruppen
In einer Diskussionsgruppe („focus group“) finden sich mehrere Teilnehmer inklusive einem Moderator ein und diskutieren eine bestimmte Themenstellung. Die Aufgabe des Moderators liegt darin, die Diskussion zu initiieren und ggf. zu lenken.

Von [Fuccella, Pizzolato & Franks 99] werden zwei Arten von Diskussionsgruppen unterschieden, nämlich “Traditional Focus Group” und “Electronic Focus Group“. Während erstere eine Diskussionsgruppe im allgemeinverständlichen Sinn ist, also ein direktes Beieinandersein der Teilnehmer darstellt, kommunizieren sie in der „Electronic Focus Group“ mittels bestimmter Software und sind räumlich getrennt. Vorteilhaft wirkt sich dies in Bezug auf Aufzeichnung und Auswertung der Ergebnisse aus. Zudem birgt die Anonymität der Teilnehmer untereinander einen ebenfalls nicht zu unterschätzenden Faktor in Hinblick auf Ehrlichkeit und innovativen Gedanken.

Im WWW hat sich eine Tätigkeit etabliert, die als „Chatten“ („Plaudern“) bekannt ist. Dort nehmen zu beliebigen Zeitpunkten Personen an sog. „Chat-Runden“ teil und diskutieren beliebige Themen. In einigen Fällen wird dies auch moderiert, d.h. eine bestimmte Person nimmt eine administrative Sonderstellung ein. Die dabei verwendete Software[6] kann aufgrund ihrer strukturellen Ähnlichkeit auch als explizites Medium für eine „Electronic Focus Group“ verwendet werden. Andere Möglichkeiten bestehen in der Nutzung von Newsgroups, e-Mail, usw. [Nielsen 97b].

In Kombination mit Interviews eignen sich Diskussionsgruppen zur Überprüfung der Allgemeingültigkeit von in Interviews gewonnenen Aussagen [Shneiderman 98, S. 145]. Nach [Nielsen 97b] eignen sich Diskussionsgruppen aber ansonsten eher zum Einschätzen von Bedürfnissen und Gefühlen und nicht so sehr zur Klärung von Sachfragen.
Während die Diskussionssitzung aus Sicht der Teilnehmer relativ unstrukturiert und „frei fließend“ stattfindet, muss der Moderator dafür sorgen, dass einerseits bestimmte Themen zur Sprache kommen und andererseits der Verlauf der Diskussion nicht abbricht. Insofern ist ein diesbezüglich trainierter Moderator und zudem eine Vorbereitung der Fragen von Vorteil wenn nicht sogar unumgänglich.

Eine günstige Anzahl der Teilnehmer wird von [Fuccella & Pizzolato 98] mit unter 10 Personen bestimmt, während Nielsen von einer Zahl zwischen 6 und 9 spricht [Nielsen 97b]. Wichtig ist dabei, dass der Moderator noch den Überblick und die Kontrolle über die Gruppe behält.
Die Verwendung von technischen Geräten (z.B. Kassettenrekorder und Videokamera) zur Aufzeichnung des Gesprächverlaufes sind an dieser Stelle von besonderer Wichtigkeit, weil mitunter eine Menge Informationen in kurzer Zeit zusammenkommen. Dies ist zwar grundsätzlich eine positive Eigenschaft einer Diskussionsgruppe, wobei die Auswertung der dabei gewonnenen Informationsflut zu den eher schwierigen Dingen gehört.

[6] z.B. Cassiopeia, http://www.cassiopeia.com/ (10.10.2000)

3.4.3. Logfiles, Gästebuch & e-Mails

Während im vorangegangenen Kapitel Methoden beschrieben wurden, in denen letztendlich Benutzer auf explizit gestellte Fragen antworten, so sollen in diesem Kapitel Methoden behandelt werden, in denen Benutzer indirekt etwas preisgeben.

Logfiles

Im WWW ist es üblich, dass Webserver jeden einzelnen Zugriff auf ihre Dokumente in einem sog. „Logfile“ (genauer: „Webserver Logfile“) verzeichnen. Dabei handelt es sich um eine Datei, in der neben allgemeinen Daten wie „Datum des Zugriffs“ und „angefordertes Dokument“ auch solche verzeichnet sind, die Rückschlüsse auf individuelle Benutzer zulassen: Über das Verzeichnen des anfragenden Rechners steht ein Schlüssel zur Verfügung, mit dem sich mehrere Zugriffe einem Individuum zuordnen lassen. Auch wenn dies aufgrund technischer Umstände nicht immer eindeutig funktioniert, so lassen sich damit dennoch grundsätzlich Benutzungsmuster erkennen.

Als Format hat sich dabei das „Common Logfile Format“[7] (CLF) etabliert, welches durch seine standardisierten Felder eine Auswertung der Daten mithilfe einer Vielzahl von Werkzeugen[8] unterstützt. Mit solchen Werkzeugen lassen sich z.B. die riesigen Datenmengen der Logfiles nach bestimmten Kriterien analysieren und statistisch aufbereiten. Fortgeschrittene Werkzeuge beherrschen u.a. auch das sog. „Logtracking“, bei dem oben erwähnte Benutzungsmuster gefunden und analysiert werden können.

In der Regel werden diese Logfiles dazu verwendet, den Erfolg einer Website zu messen, wobei die Anzahl der abgerufenen Dokumente innerhalb eines Zeitraums („Hits“) als grober Bewertungsmaßstab gelten. Andere Maßeinheiten, wie sog. „Pageviews“ oder „Pageimpressions“ bieten ein detaillierteres Bild in Bezug auf die Akzeptanz einer Website. Gerade in Hinblick auf Vermarktung und der Schaltung von Werbebannern werden derartigen Werten große Bedeutung beigemessen. In [Fuller 96] hingegen wird deutlich gemacht, dass diese Werte mitnichten ein zuverlässiges Maß für den Erfolg einer Website darstellen, sondern sogar im Gegenteil aufgrund eines schlechten Designs zustande kommen können. Obwohl sich die Wirtschaft dieser Tatsache durchaus bewusst ist, wird mangels Alternativen an diesen Maßeinheiten festgehalten.

Eine andere Art von Logfiles stellen die sog. „Search Log Files“ [Yu, Prabhu & Neale 98] dar. Diese spielen eine Rolle, wenn eine Website über einen Suchmechanismus verfügt. Nach Eingabe eines Suchbegriffs seitens des Benutzers werden von so einem Mechanismus alle Dokumente des Webservers nach diesem Begriff durchsucht und ausgegeben. Zusätzlich zur Funktionalität der Suche können diese Begriffe zwecks späterer Analyse auch in einem speziellen Logfile gespeichert werden.

Im Kontext der Benutzerorientierung können die aus einem Logfile hervorgehenden Größen qualitativ betrachtet werden um beispielsweise Rückschlüsse über die Akzeptanz einzelner Dokumente oder aber allgemeine Probleme bei der Bedienung zu erhalten.

---

[7] http://www.apache.org/docs/mod/mod_log_common.html (19.10.2000)
[8] z.B. „Websuxxess“, http://www.exody.net/ger/products/websuxess/websuxess.html (24.10.2000)

Gästebuch und e-Mail
Sofern schon eine Präsenz im Internet beispielsweise in Form eines e-Mail-Systems (vgl. [Norton 99]) oder auch einer Website mit einem Gästebuch (vgl. [Yu, Prabhu & Neale 98]) besteht, können über diese Kanäle die Aussagen von Benutzern zu bestimmten Themen gewonnen werden. Sei es nun, dass sie Kritik, Hinweise oder Wünsche äußern.
Dieses Feedback kann analysiert werden und lässt unter Umständen Rückschlüsse auf verschiedene Sachverhalte zu. Das Ausgangsmaterial wird in der Regel unspezifisch, unstrukturiert und in großer Anzahl auftreten, was die Analyse entsprechend aufwendig macht. Vorteil ist in diesem Zusammenhang aber, dass es sich über einen längeren Zeitraum ansammelt und somit stückweise in kleineren zeitlichen Abständen (z.B. tageweise) analysiert werden kann.

3.4.4. Szenarios

Mittels der Methode der „Szenarios“ steht eine Möglichkeit zur Verfügung, mit der auch ohne den Zugriff auf Benutzer benutzerorientiert vorgegangen werden kann. Dabei versetzt sich der Entwickler in die Rolle eines typischen Benutzers und schildert eine Situation aus dessen Perspektive, die er in Form einer Geschichte festhält. In dieser Geschichte sollte der fiktive Benutzer mit typischen Situationen bei der Ausführung von Aufgaben konfrontiert werden und der Umgang mit dieser Situation geschildert werden [Shneiderman 98, S. 111]. Merholz charakterisiert Szenarios so: “They force you to think the way users act - not rationally, but impulsively and emotionally” [Merholz 98].

Eine Schwierigkeit besteht darin, die Geschichte ohne konkreten Bezugspunkt auf rein hypothetischer Basis durchführen zu müssen. Somit muss sowohl das Verhalten des Benutzers, als auch das des benutzten Systems quasi simuliert werden. Hilfreich kann es dabei sein, wenn im Vorfeld schon der Charakter eines solchen hypothetischen Benutzers in einem sog. „Profil“ definiert ist (vgl. [Fleming 98, S. 8]).
In Bezug auf eine Website kann bei einem Szenario mit der Fragestellung begonnen werden, welche Information (z.B. „die Telefonnummer eines Notfalltechnikers“) unter welchen Umständen (z.B. „ein System ist defekt und muss schnellstens repariert werden“) gefunden werden soll.

Dieses Mittel soll nicht die Partizipation von realen Benutzern ersetzen, sondern hat seine Stärken innerhalb der Startphase eines neuen Projekts, in der viele Rahmenbedingungen noch ungeklärt sind. Der Einsatz von Szenarios kann dabei als erster Anhaltspunkt für Entwürfe bzw. als Diskussionsgrundlage dienen. Indem man sich in die Rolle eines solchen Benutzers hineinversetzt, bekommt man u.U. ein Gefühl dafür, wo allgemeine Probleme liegen könnten.

In [Fuccella, Pizzolato & Franks 99] werden Szenarios unter geringfügig anderen Rahmenbedingungen definiert. In diesem Fall werden sie nicht vom Entwickler erstellt, sondern vom Benutzer selbst. Die Grundvoraussetzung, dass es sich um ein hypothetisches Vorgehen an einem wiederum hypothetischen System handelt, bleibt allerdings dieselbe.

3.4.5. Sitereview

In diesem Abschnitt soll unter dem Oberbegriff „Sitereview“ eine Evaluationsmethode anhand eines Prototypen vorgestellt werden. Es handelt sich dabei um den Prototypen einer Website, d.h. beispielsweise einer Struktur aus miteinander verknüpften

HTML-Seiten oder aber einem Mockup. Im Laufe dieser Diplomarbeit wird konkreter vorgestellt werden, wie Website-Prototypen erstellt werden können, während an dieser Stelle nur die Verwendung derartiger Prototypen geschildert werden soll.

Grundsätzlich sollten diese Prototypen nach [Nielsen 96b] so benutzt werden, dass Benutzer daran „echte Aufgaben“ erfüllen. Damit ist gemeint, dass nicht nur einfach am Prototyp „herumgespielt“ wird, sondern Aufgaben zu erfüllen sind, wie sie in der Realität vorkommen würden. Da es sich in dieser Diplomarbeit um Informationssysteme dreht, werden Aufgaben weitestgehend nach dem Schema „Finde Information xyz“ gestaltet sein. Ziel ist es dabei, unter möglichst realistischen Bedingungen herauszufinden, an welcher Stelle Probleme auftreten.

Als Methode, solche Probleme zu erkennen, hat das sog. „Laute Denken“ („Thinking Aloud“) eine große Bedeutung gewonnen (vgl. [Weinreich 97, S. 89]). Während der Benutzer eine Aufgabe erfüllt, soll er nach Möglichkeit permanent verbalisieren, was ihm derzeit durch den Kopf geht. Damit kann zumindest in Ansätzen das mentale Modell des Benutzers bezüglich des evaluierten Systems verstanden und eventuelle Missverständnisse erkannt werden. Da Benutzer aber häufig dazu neigen, den Strom der Gedanken auszusetzen, sollte ein anwesender Moderator immer wieder zur Fortsetzung ermuntern („was geht Dir gerade durch den Kopf?“).
Eben dies stellt einen der Kritikpunkte am „Thinking aloud“ dar, der von [Wildman 95] angeführt wird. Grundsätzlich sieht er wesentliche strukturelle Mängel in dieser Methode, da das Reden während des Problemlösens ungewohnt ist und somit eine Menge an kognitiver Energie auf sich zieht. Diese Doppelbelastung muss somit zu einer Beeinflussung des eigentlichen Problemlösungsvorganges führen. Als Alternative wird die sog. „Paired-User“-Methode genannt, bei der jeweils zwei Benutzer gleichzeitig eine Aufgabe bearbeiten und wechselseitig von ihren Fähigkeiten profitieren. Die dabei entstehenden Gespräche und Diskussionen lassen ihrerseits Rückschlüsse auf Probleme mit dem Prototypen zu.

Unabhängig von der konkreten Methode kann der Ablauf eines Sitereviews so aussehen, dass den Benutzern konkrete Aufgabenstellungen in schriftlicher Form vorliegen, die sie innerhalb einer Sitzung Aufgabe für Aufgabe abarbeiten. Dabei kann beispielsweise die Zeit gemessen werden, die zur Bearbeitung benötigt wird, oder auch die Richtigkeit des Ergebnisses. In Hinblick auf eine Aufgabenstellung wie „finde Information xyz“ ist zudem die Anzahl der verwendeten Hyperlinks interessant.
Zwar kann das Messen dieser Größen vom Moderator übernommen werden, wobei aber auch die Möglichkeit der Logfiles (s.o.) zur Verfügung steht.

Neben der Möglichkeit anhand konkreter Aufgaben grundsätzliche Probleme bei der Benutzung des Prototypen zu ermitteln, kann er auch als Basis für subjektive Kritik, beispielsweise bezüglich der Ästhetik der Website, dienen [Sullivan 96b]. In diesem Fall kann auf oben beschriebene Methoden, wie beispielsweise eine Diskussionsrunde, zurückgegriffen werden, die im Anschluss an die Inspektion des Prototypen stattfinden kann.

3.4.6. Evaluation nach Heuristiken & Guidelines

Im Laufe der Zeit haben sich bei der Entwicklung von Softwaresystemen und insbesondere auch bei Websites Prinzipien, sog. Heuristiken [Nielsen o.D.], herauskristallisiert, was bei der Gestaltung solcher Systeme z.B. in Bezug auf Benutzbarkeit grundsätzlich falsch oder richtig gemacht werden kann. Andererseits sind auch in

sog. „Guidelines“ Regeln definiert, nach denen in Bezug auf Konsistenz ein System gestaltet werden sollte.

Mit diesen Regeln als Grundlage kann die Evaluation vom Entwickler bzw. von besonderen Experten durchgeführt werden, wobei jeweils aufgelistet werden kann, welche dieser Regeln verletzt wird.
Als Evaluationsgegenstand eignen sich sowohl Prototypen, als auch Entwürfe die wesentlich abstrakter (z.B. in Form von Notizen) vorliegen, weshalb eine solche Evaluation schon zum frühestmöglichen Zeitpunkt stattfinden kann.
Zwar ist die Wahrscheinlichkeit, dass ein Experte einige Probleme übersieht, recht groß, aber über eine Untersuchung hat Nielsen dafür eine Lösung gefunden: Der Einsatz mehrerer Evaluatoren lässt die Wahrscheinlichkeit steigen, dass alle Probleme gefunden werden, wobei sich mit der Anzahl von drei bis fünf solcher Durchläufe ein gutes Verhältnis ergibt [Nielsen o.D.].

Obwohl quantitativ gesehen eine derartige Evaluation oftmals als erfolgreich angesehen werden kann, so darf nicht unberücksichtigt bleiben, dass sich konkrete Probleme von Benutzern nicht bzw. nur sehr schwer auf diese Weise herauskristallisieren lassen (vgl. [Weinreich 97, S. 85]). Dies mag daran liegen, dass die zugrundeliegenden Regeln nur auf einer sehr verallgemeinernden Basis entstanden sind.

Heuristiken und Guidelines gibt es für die verschiedensten Kontexte[9], weshalb ein allgemeingültiger Prinzipienkatalog unpraktikabel ist. Insofern sei exemplarisch auf die vergleichsweise universellen „Ten Usability Heuristics“ in [Nielsen o.D.b] verwiesen.

3.4.7. Auswertung der Ergebnisse

Da die in diesem Kapitel vorgestellten Methoden liefern je nach Anwendung und nach Kontext unterschiedliche Ergebnisse, deren Auswertung hier aufgrund der Komplexität nicht allgemein beschrieben werden kann. Grundsätzlich sollte die Auswertung aber immer in Bezug auf die übergeordnete Fragestellung erfolgen. Mittel wie die hier vorgestellten werden nicht aus reinem Selbstzweck eingesetzt, sondern verfolgen ein bestimmtes Ziel, wie z.B. das Herausfinden bestimmter Probleme der Benutzbarkeit.
Zwar werden einige Ergebnisse quantitativer Natur sein, was die Anwendung statistischer Mittel ermöglicht, aber oftmals sind die Ergebnisse unstrukturiert und können nur qualitativ betrachtet werden. Insbesondere bei subjektiven Methoden muss mitunter „zwischen den Zeilen gelesen“ werden und Muster müssen in Beziehung zu anderen Ergebnissen erkannt werden (vgl. [Fleming 98b]).

Sofern mehrere Testpersonen an ein und demselben Benutzertest teilnehmen, werden die dabei erlangten Ergebnisse untereinander nicht immer eindeutig, sondern im Extremfall gänzlich unterschiedlich sein. Das ist prinzipiell nicht überraschend, wenn man davon ausgeht, dass die Bedürfnisse verschiedener Menschen auch extrem unterschiedlich sein können. Um solche Ergebnisse zu verwerten, kann man eine Grenze setzen, nach der man ein Ergebnis als eindeutig ansehen will. Eine solche Grenze, stellt die „80/20“-Regel dar, nach der 80% der Teilnehmer zum selben Er-

[9] beispielsweise für Körperbehinderte in der „Web Accessibility Initiative“ (WAI), http://www.w3.org/WAI/ (10.10.2000)

gebnis kommen müssen, damit dieses Ergebnis als eindeutig angesehen wird [Fleming 98, S. 42].

4. Entwicklung von Websites

Nachdem im vorletzten Kapitel das grundsätzliche Wesen von Websites als Sonderfall eines Hypertextes analysiert wurde, soll in diesem Kapitel ein grundsätzlicher Blick auf den Entstehungsprozess von Websites und dessen Umstände geworfen werden.

4.1. Kennzeichen und Probleme

In der Disziplin des Software Engineering hat sich über die Zeit ein Verständnis entwickelt, wie Softwareprojekte erfolgreich bearbeitet und beendet werden können. Bis zur sog. „Software-Krise“ in den 60er Jahren wurden Softwareprojekte allgemein eher unstrukturiert bearbeitet, was sich in der mangelnden Qualität der Software und dem Ressourcenverbrauch bei deren Entstehung niedergeschlagen hat [Pflüger 95, S. 201]. Aus dieser Krise heraus sind eine Menge von Ansätzen hervorgegangen, z.B. in Form von systematischen Vorgehensweisen, durch die die Softwareentwicklung zu einer ingenieurshaften Disziplin werden konnte. Auf diese Weise lassen sich Softwareprojekte deutlich erfolgreicher bearbeiten.

Demgegenüber mangelt es bei der Erstellung von Websites deutlich an einem derartigen Vorgehen [Murugesan, Deshpande, Hansen & Ginige 99]: Die Entwicklung von Websites ist durch ein „ad hoc“ Vorgehen gekennzeichnet, in dem oftmals weder Systematik, noch eine Qualitätskontrolle zu erkennen ist. Die Wartung eines solchen Systems ist zudem als ein kontinuierliches Flickwerk an der Oberfläche der Probleme gekennzeichnet, was insgesamt zu eher minderwertigen Systemen und der allgemeinen Gefahr einer „Web-Krise“ führt.

Die Gründe für ein derartig undiszipliniertes Vorgehen können in vielerlei Richtungen gefunden werden. Zum einen handelt es sich beim WWW um ein vergleichsweise junges System, das erst in den letzten wenigen Jahren an großer Bedeutung gewonnen hat. Insofern war vielleicht einfach die Zeitspanne zu kurz, in der sich im Vergleich zum Software Engineering entsprechende Standards hätten entwickeln können. Zum anderen spielt sicherlich auch die leichte Erlernbarkeit von HTML und den damit verbundenen Werkzeugen eine große Rolle, wodurch die Möglichkeit für nahezu Jedermann geboten wird, eine Website zu erstellen. Somit sind wahrscheinlich eine Menge Personen bei der Erstellung von Websites beteiligt, die keine Kenntnis essentieller Prinzipien sowie kein notwendiges Verständnis der Materie haben.

Das Ergebnis eines solchen Vorgehens führt neben dem Scheitern des jeweiligen Projekts durch beispielsweise zu hohe Kosten, zu langer Entwicklungszeit oder Qualitätsmangel auch insbesondere zu mangelnder Akzeptanz des Ergebnisses (vgl. [Jones & Lynch 99]). Zwar entstehen oft Websites, die optisch ansprechend sind und zudem oberflächlich gesehen auch „funktionieren“, die aber nicht als Erfolg im Hinblick auf die Ziele oder den eigentlichen Zweck verbucht werden können.
Oftmals fehlt aber das Verständnis dafür, dass es einen unmittelbaren Zusammenhang zwischen dem Ergebnis der Entwicklung und mangelnden Besucherzahlen gibt. Sofern eine Website eine schlechte Benutzbarkeit aufweist und sich nicht an den Bedürfnissen der Benutzer orientiert, werden die Benutzer fortbleiben, ohne dass die Gründe unbedingt erkannt werden [Fleming 98, S. 7].

Langsam entstehen allerdings Ansätze in Form des „Web-Engineering“, in der solche Probleme erkannt und behoben werden. Dabei soll der Begriff des „Engineering“

wiederum für die Anwendung wissenschaftlicher Erkenntnisse bei der Erstellung entsprechender Lösungen stehen (vgl. [Deshpande, Hansen & Murugesan 99] und [Murugesan, Deshpande, Hansen & Ginige 99]). An genau diesem Prinzip sollen sich die weiteren Ausführungen in dieser Diplomarbeit orientieren.

4.2. Website als Software

Wie im vorherigen Kapitel dargestellt wurde, soll ein ingenieurshafter Ansatz bei der Entwicklung von Websites behilflich sein. Durch die zunächst offensichtliche thematische Nähe von Software Engineering und Web-Engineering liegt es nahe, Methoden und Erkenntnisse des ersteren in das Web-Engineering zu übernehmen. Als Legitimation dieses Vorgehens soll zunächst das Verhältnis von Software zu Websites geklärt werden.

Software wird allgemein als „Gesamtheit aller Programme, die auf einer Rechenanlage eingesetzt werden können“ definiert, wobei unter einem Programm die „Formulierung eines Algorithmus“ verstanden wird [Engesser 93, S. 655 und S. 541]. Der Aspekt der Algorithmen, der somit für Software gilt, ist in Bezug auf Websites allerdings nicht relevant[10], da es dort letztendlich um das Erstellen von algorithmuslosem HTML geht. Das Erstellen von Websites besteht weiterhin aus dokumentorientierten Tätigkeiten wie Inhaltserstellung oder Grafikdesign [Murugesan, Deshpande, Hansen & Ginige 99, S. 2], weshalb der Vergleich zur Erstellung eines Magazins nahe liegt [Norton 99, S. 2].
Allerdings bauen Websites auf einem System von Softwarekomponenten, wie z.B. Webservern und Webbrowsern, auf, die es gebührend zu berücksichtigen gilt. Andererseits hat eine Website interaktive Elemente in Form von Hyperlinks, deren Funktionalität zwar nicht direkt erstellt wird, die aber dennoch durch das explizite Setzen der Hyperlinks definiert werden muss. Außerdem setzt das erfolgreiche Erstellen von Websites ein gewisses Maß an Verständnis der zugrundeliegenden Infrastruktur voraus, beispielsweise in Form der verwendeten Protokolle. So gesehen kann auch das Erstellen von Websites als eine Art der Programmierung angesehen werden.

Bei Websites handelt es sich um ein interaktives Medium, das eine Menge von Informationen bereitstellt, auf die von Benutzern zugegriffen wird. Es kann davon ausgegangen werden, dass der Zugriff auf eine entsprechende Information nicht aus reinem Selbstzweck geschieht, sondern dass der Benutzer von einem konkreten Problem bzw. Bedürfnis angetrieben wird. Insofern besteht ein Zusammenhang zu Anwendungssoftware: „Anwendungssoftware ist ein Mittel zum Zweck, um fachliche Aufgaben in einem oder mehreren Anwendungsbereichen zu erledigen.“ [Züllighoven et al. 98, S. 124].

Alles in allem können Websites in dieser Diplomarbeit als Grenzfall von Software betrachtet werden. Daraus ergibt sich die Konsequenz, dass die Methoden des Software Engineering sich zumindest teilweise auf das Web-Engineering übertragen lassen können. Dies soll in dieser Arbeit im Wesentlichen im Rahmen der Benutzerorientierung (vgl. Kapitel 3) geschehen und zusätzlich auch in Bezug auf die Prozessmodelle (vgl. Kapitel 4.4).

[10] Techniken wie z.B. „Javascript“ bleiben in diesem Zusammenhang unberücksichtigt.

4.3. Beteiligte Personen

Bei der Erstellung von Websites unter dem Gesichtspunkt der Benutzerorientierung sind mehrere Gruppen von Personen beteiligt, die jeweils auf eine bestimmte Weise Einfluss auf den Entwicklungsprozess nehmen. In diesem Kapitel sollen diese Gruppen definiert und ihre Beziehungen untereinander dargestellt werden.

Als beteiligte Gruppen sollen die „Entwickler“, die „Auftraggeber“ und die „Benutzer“ unterschieden werden. Grob gesehen besteht eine Beziehung zwischen diesen drei Gruppen in Form von folgendem typischen Szenario: Die Auftraggeber möchten aus bestimmten Gründen eine Website erstellt haben, in der ein bestimmtes Zielpublikum in Form der „Benutzer“ angesprochen werden soll. Mit der Erstellung werden die „Entwickler“ beauftragt, die dies entsprechend der Vorgaben der Auftraggeber und unter Einbeziehung der Bedürfnisse der Benutzer (vgl. Kapitel 3.1) leisten. Die Gruppe der Benutzer wurde schon im o.g. Kapitel beschrieben, weshalb hier insbesondere auf die „Auftraggeber“ und „Entwickler“ eingegangen werden soll.

In [Rosenfeld & Morville 98, S. 20] wird die Entwicklung von Websites als interdisziplinäres Feld beschrieben, in dem im einzelnen folgende Richtungen vertreten sind:

- *Marketing*: Das Marketing legt das Augenmerk auf die angestrebten Ziele in Bezug auf das Zielpublikum. Dabei ist es für Strategien zuständig, wie das Publikum dazu gebracht wird, die Website zu besuchen und auch wiederzukehren.
- *Informationsarchitektur*: Die Informationsarchitektur entwirft die Organisation der Daten und den Zugriff auf diese in Form der Navigation.
- *Grafikdesign*: Das Grafikdesign ist für die visuellen Mittel der Website, wie z.B. Layout, zuständig.
- *Redaktion*: Die Redaktion ist für die Erstellung bzw. Bearbeitung der Inhalte zuständig, d.h. es werden z.B. Texte geschrieben oder auch Bilder ausgewählt.
- *Technik*: In der Technik wird die technologische Infrastruktur z.B. in Form von Servern bereitgestellt.
- *Projektmanagement:* Das Projektmanagement sorgt für den reibungslosen Projektablauf in Hinblick auf das Budget und Zeitrahmen.

Zusätzlich zu den hier genannten Disziplinen erscheinen noch die Nennungen der „Software-Ergonomie“ sowie der „Programmierung“ als sinnvoll. Während die Programmierung für die konkrete Umsetzung des Websiteentwurfs in HTML und entsprechend anderer Technologien zuständig ist und somit auch zur „Technik“ gezählt werden kann, ist die Software-Ergonomie für die ergonomische und benutzungsfreundliche Gestaltung der Website zuständig. Allerdings wird in [Morville 99] deutlich, dass zwischen der Informationsarchitektur und der Software-Ergonomie in diesem Kontext ein Zusammenhang besteht: In beiden Disziplinen spielt die Entwicklung von benutzergerechten Systemen eine Rolle.

Beim Auftraggeber wird es sich wahrscheinlich ebenfalls nicht um eine Einzelperson, sondern vielmehr um eine Organisation aus mehreren Menschen handeln, deren einzelne Rollen hier aber nicht betrachtet werden sollen. Stattdessen soll davon ausgegangen werden, dass die Auftraggeber aus Sicht der Entwickler als eine homogene Gruppe aufgefasst werden können.
Wie im obigem Szenario deutlich gemacht wurde, findet eine Website-Entwicklung unter Einflussnahme sowohl der Benutzer als auch der Auftraggeber statt. Seitens

der Benutzer findet eine Orientierung an deren Bedürfnissen statt, während die Auftraggeber z.B. ihre Zielsetzung oder eine Firmenphilosophie einbringen. In [Rahardja 99] wird in diesem Zusammenhang auch allgemein von „Stakeholdern“ („Anteilseigner“) gesprochen, also Menschen, die „Anteil“ an der Entwicklung nehmen.

Die Motivationen der einzelnen Stakeholder können sowohl von höchst unterschiedlicher als auch von sich widersprechender Natur sein. Während die Auftraggeber beispielsweise mittels plakativer Anpreisung den Verkauf eines ihrer Produkte ankurbeln wollen, möchte ein einzelner Benutzer vielleicht lieber kritische Informationen über dieses Produkt erhalten (vgl. [Fleming 98, S. 7]). Die Entwickler, die vielleicht eine persönliche Affinität zu bestimmten Technologien haben, wollen diese ihrerseits in die Website integrieren.
Die Schwierigkeit besteht nun darin, eine Balance zwischen den einzelnen Sichtweisen zu finden, ohne dass eine der beteiligten Gruppen unberücksichtigt bleibt und somit das Projekt als solches gefährdet ist (vgl. [Rosenfeld 99b]).
Zwar kann man einen Standpunkt vertreten, nach dem die Auftraggeber als Finanziers des Projektes auch die Entscheidungshoheit besitzen, wobei aber nicht vergessen werden darf, dass z.B. ohne die Akzeptanz der Benutzer das gesamte Projekt hinfällig ist. Insofern kann davon ausgegangen werden, dass auch der Auftraggeber Interesse an einer benutzergerechten Gestaltung des Systems hat und damit von einer „Benutzerorientierung“ als zentrales Leitbild ausgegangen werden kann.

In Bezug auf den Gedanken der „Stakeholder“ sollen die in den folgenden Kapiteln beschriebenen partizipativen Vorgehensweisen sich sowohl auf die Einbeziehung der Benutzer, als auch auf die der Auftraggeber beziehen. Wenn also im Folgenden beispielsweise von „Benutzertests“ die Rede ist, dann ist dabei implizit auch die Einbeziehung der Auftraggeber gemeint. Auf diese Weise lassen sich prinzipiell mit denselben Methoden die Bedürfnisse und Ansichten der allgemeinen Gruppe der Stakeholder bestimmen.

4.4. Prozess- und Datenmodelle

Da diese Diplomarbeit den Entwurf von Websites zum Thema hat, ist es notwendig, dass entsprechende Vorgehensmodelle aufgezeigt werden. Dazu soll untersucht werden, ob es bereits Modelle gibt, die den in den vorherigen Kapiteln ausgeführten Prinzipien folgen, bzw. inwieweit sich diese übernehmen lassen. Obwohl es eine deutlich größere Anzahl als die hier vorgestellten gibt, so sollen aufgrund des hier sehr begrenzten Rahmens nur die bekanntesten vorgestellt werden.

4.4.1. Prozessmodelle des Software Engineering

Wie schon angedeutet wurde, sind aus dem Software Engineering diverse Prozessmodelle bekannt, die sich zumindest auch theoretisch auf die Website-Entwicklung anwenden lassen. In Anlehnung an [Züllighoven et al. 98, S. 542ff] sollen die bekanntesten Vertreter hier vorgesellt werden:

Wasserfallmodell
Beim Wasserfallmodell handelt es sich wahrscheinlich um das prominenteste und damit klassischste aller Prozessmodelle. Die Entwicklung wird dabei als lineare Folge von Schritten gesehen, die sequentiell durchlaufen werden und deren Ende die Fertigstellung des Produktes kennzeichnet. Typischerweise bestehen diese Schritte aus „Analyse“, „Entwurf“, „Implementation“, „Test“ und „Einsatz“. Das Ende eines jeden Schrittes wird mittels eines sog. „Meilensteindokumentes“ dokumentiert, auf das

die folgenden Schritte zugreifen können. Es besteht dabei die grundsätzliche Annahme, dass jeder Schritt vollständig abgeschlossen werden kann und somit eine Überarbeitung vorheriger Meilensteindokumente nicht notwendig ist.

In der Praxis hat sich allerdings gezeigt, dass sich ein derartiges Vorgehen aus verschiedenen Gründen nicht vernünftig praktizieren lässt (vgl. [Züllighoven et al. 98, S. 543f]). Insbesondere spielt die Tatsache eine Rolle, dass ein Entwurf in hohem Maße dynamisch ist und ein Lernprozeß der Beteiligten stattfindet. Erst im Laufe der Zeit entwickelt sich ein Verständnis für die Materie, was sich im Prinzip in der Überarbeitung von Ergebnissen niederschlagen müsste. Dies wird vom Wasserfallmodell durch sein striktes sequentielles Vorgehen nicht berücksichtigt.

Spiralmodell
Beim Spiralmodell werden die einzelnen Entwicklungsschritte zyklisch mehrfach durchlaufen und münden nach einigen Durchläufen letztendlich in einem fertigen Produkt. Damit wird die Überarbeitung einzelner Entwurfsergebnisse ermöglicht und behebt somit ein großes Manko des Wasserfallmodells. Kritisiert wird dieses Modell in [Züllighoven et al. 98, S. 544] aber insbesondere deshalb, weil diesem die Vorstellung zugrunde liegt, dass der Entwurfsgegenstand irgendwann einmal als „fertig“ definiert werden kann („Produktsicht“) wodurch eine eher problematische Trennung zwischen „Herstellung“ und „Wartung“ definiert wird.

Evolutionäres Vorgehensmodell
Ein evolutionäres Vorgehen sieht ein iteratives Entwickeln vor, d.h. Entwurfsergebnisse befinden sich in einem permanenten Prozess der Bewertung und der Überarbeitung. Auf diese Weise wird der Entwicklungsgegenstand Stück für Stück unter Einbeziehung aktueller Einflüsse weiterentwickelt. Dies erfolgt auf Basis von sog. Entwicklungsdokumenten (s. Kapitel 4.5), die den jeweiligen Stand unter verschiedenen Aspekten festhalten. Wichtig ist beispielsweise, dass im Gegensatz zum Wasserfallmodell keine vorgegebene Reihenfolge existiert, in der Dokumente bearbeitet werden müssen. Stattdessen soll es zu jedem Zeitpunkt möglich sein, auf bestehende Dokumente zurückzugreifen und diese ggf. zu modifizieren ([Züllighoven et al. 98, S. 545]).

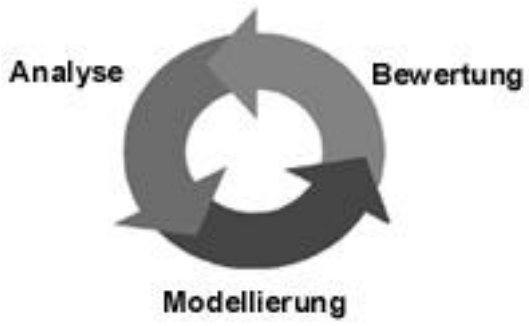

Abbildung 4.1: Der Autor-Kritiker-Zyklus, bei dem die Phasen „Analyse“, „Bewertung“ und „Modellierung“ zyklisch durchlaufen werden.

Von [Züllighoven et al. 98, S. 129] wird als wesentlicher Bestandteil eines evolutionärem Vorgehens der sog. „Autor-Kritiker-Zyklus“ beschrieben, der den zyklischen Ablauf der drei Phasen „Analyse“, „Modellierung“ und „Bewertung“ vorsieht: In der Analyse werden die jeweiligen Umstände von den „Autoren“ betrachtet bzw. analysiert und als Konsequenz Entwurfsentscheidungen getroffen, die in einem „Modell“ fest-

gehalten werden. Dieses Modell wird von den „Kritikern“ bewertet, was wiederum zu einer erneuten Situationsanalyse durch die Autoren führt.

Hinter diesem Vorgehen steht die grundsätzliche Erkenntnis, dass es schwierig ist, die Konsequenzen aktueller Entwurfsentscheidungen abzusehen. Das Herausfinden solcher Konsequenzen wird über die „Kritiker“ ermöglicht, die aus einer externen Warte heraus Einfluss nehmen. Kritiker können z.B. Benutzer sein, die einen aktuellen Entwurf in Form eines Prototypen testen und ihre Schwierigkeiten damit schildern. Dabei entsteht zumindest bei den Entwicklern ein Lernprozess, in dem langsam die Natur der Materie begriffen wird. Als wichtige Möglichkeit der Modellierung gelten die sog. „Prototypen“ (vgl. Kapitel 3.3), bei denen es sich im Prinzip um die Umsetzung einzelner Entwurfsaspekte in Form einer ablauffähigen Implementierung handelt.

Potentiell findet eine evolutionäre Entwicklung kein Ende (s.o.), sondern führt zu einer permanenten Weiterentwicklung des Entwurfsgegenstandes. Dies mag zunächst ungewohnt klingen, macht aber in Bezug auf das herkömmliche Konzept der „Wartung“ Sinn: Nachdem ein Softwareprodukt ausgeliefert wurde, werden oft weitere Arbeiten daran durchgeführt, die z.B. das Beheben von Programmierfehlern betreffen oder das Anpassen an neue Gegebenheiten. So gesehen ist die Auslieferung aus Sicht der evolutionären Entwicklung nur ein willkürlich festgelegter Zeitpunkt innerhalb der Entwicklung.

Gerade in Hinblick auf ein benutzerorientiertes Vorgehen und insbesondere in Bezug auf Methoden des Prototyping, stellt ein evolutionäres Prozessmodell ein adäquates Mittel dar [Oppermann & Reiterer 94, S. 350].

4.4.2. Datenmodelle für Hypertexte

In den vorherigen Kapiteln wurden Prozessmodelle vorgestellt, die zwar aus der Welt des Software Engineering stammen, dennoch aber offensichtlich auf die Entwicklung von Websites anwendbar sind. Als Modelle, die ein möglichst breites Spektrum an unterschiedlichen Softwareprojekten abdecken sollen, können sie natürlich nur auf einer sehr abstrakten Ebene beschrieben werden und müssen im Einzelfall an die gegebenen Umstände adaptiert werden. Demgegenüber sind explizit für Hypertexte Prozessmodelle entwickelt worden, die bezogen auf das Vorgehen konkreterer Natur sind. Bevor auf diese Modelle im folgenden Kapitel eingegangen wird, soll in diesem Kapitel der Modellierungsgegenstand solcher Prozessmodelle genauer untersucht werden.
Dazu soll wiederholt werden, dass ein Hypertext bzw. eine Website in ihrem Wesen als Informationsraum prinzipiell als Ansammlung von miteinander verknüpften Daten gesehen werden kann. Insofern wird die Entwicklung eines Hypertextes zum Großteil die Modellierung dieser Daten zum Gegenstand haben. Eine entsprechend abstrakte Sicht auf diese Daten in Form von Datenmodellen stellt somit eine Notwendigkeit beim Hypertextentwurf dar [Isakowitz, Stohr & Balasubramanian 95, S. 35].

In Kapitel 2.6 wurde eine hierarchische Datenmodellierung als Orientierung für die Website-Entwicklung vorgestellt. Parallel zu einer derartigen Struktur gibt es technikorientierte Datenmodelle, wie z.B. das sog. „Entity-Relationship“-Modell (ER): Entities stellen dabei Repräsentanten von Dingen der realen Welt (z.B. „Person“) dar, die mittels Attributen (z.B. „Name“, „Alter“, ...) charakterisiert werden. Zwischen einzelnen Entities können mittels „Relationships“ Beziehungen modelliert werden [Enges-

ser 93, S. 231]. Die Entities bzw. Relationsships stellen Schablonen dar, nach denen die eigentlichen Daten (die in diesem Zusammenhang „Datensätze“ genannt werden sollen) abgelegt werden.

Kennzeichnend für diese aus der Welt der Datenbanken entstammenden Datenmodelle ist ein hoher Grad von Struktur, der den einzelnen Datensätzen zugrunde liegt. Solch eine Struktur äußert sich beispielsweise darin, dass alle Datensätze dieselben Attribute aufweisen und somit eine homogene Datenbasis bilden. Vorteile eines solchen festen Schemas liegen z.B. in der Möglichkeit, große Mengen an Daten zu verwalten und auf diese zugreifen zu können. Auf Basis der vorgegebenen Struktur lassen sich Abfragen z.B. in Form von SQL formulieren und somit auf diese Daten zugreifen.

In Bezug auf den Entwurf von Websites liegt in der homogenen Strukturierung grundsätzliches ein Problem, wie in [Rosenfeld & Morville 98, S. 41] festgestellt wird: Der Inhalt von Websites ist oftmals in hohem Maße heterogen, was eine Modellierung gemäß obiger Modelle verhindert bzw. erschwert. Die Heterogenität ergibt sich aus der Orientierung an den Bedürfnissen der Benutzer nach bestimmten Informationen. Konsequenterweise führt dies zu einer Ansammlung von entsprechenden Daten aus unterschiedlichen Bereichen zu einer gemeinsamen Informationsbasis.
Wenn man zudem davon ausgeht, dass der Entwicklungsprozess gleichzeitig ein Lernprozess ist (s. Kapitel 4.4.1), werden die Entwurfsergebnisse innerhalb des Prozesses grundsätzlich einer kontinuierlichen Veränderung unterliegen. Ein starres Datenschema wird sich mit einem derartigen Prozess nur unzureichend vereinbaren lassen.
In Anlehnung an [Tenny 96] kann in diesem Zusammenhang von einer „datenorientierten“ gegenüber einer „benutzerorientierten“ Sicht gesprochen werden. Zwar hat die datenorientierte Sicht in Bezug auf Wiederverwendbarkeit von Daten durchaus einen Sinn [Rosenfeld 99a], wobei sich dies aber in Bezug auf die in dieser Arbeit angestrebte Benutzerorientierung nur schlecht vereinbaren lässt.

Insofern sei festgestellt, dass Datenmodelle, die von einem homogenen und starren Datenschema ausgehen, für einen benutzungsorientierten Entwurfsprozess nicht, bzw. nur unzureichend geeignet sind. Daher lassen sich auch Prozessmodelle, die sich auf derartige Datenmodelle beziehen, nur schlecht auf den hier angestrebten benutzungsorientierten Entwurfsprozess anwenden.
Als Konsequenz wurde in dieser Arbeit ein spezielles Modell einer Website entwickelt, das seinerseits als Datenmodell bzw. Modellierungsgegenstand dienen soll.

4.4.3. Prozessmodelle für Hypertexte

Im Zusammenhang mit dem Entwurf von Hypertexten werden in der Literatur (z.B. [Scharl 99] oder [Balasubramanian & Bashian 98]) im Wesentlichen zwei Vorgehensweisen genannt: Es handelt sich dabei zum einen um die „Relationship Management Methodology“ (RMM) und zum anderen um das „Object-Oriented Hypermedia Design Modell“ (OOHDM). In beiden werden jeweils Prozessschritte definiert und deren Zusammenspiel beschrieben.

RMM
RMM liegt das eigens entwickelte Datenmodell RMDM (Relationship Management Data Model) zugrunde, welches als Modellierungsgegenstand bei der Entwicklung dient. Dieses Datenmodell besteht aus mehreren Entwurfsprimitiven, die sich grund-

sätzlich an den Hypertextelementen „Knoten“ und „Hyperlink“ anlehnen, aber durch spezielle Primitive (z.B. in Bezug auf die Navigation) erweitert sind. Kennzeichnend ist, dass sich RMM an einer stark strukturierten Datenbasis orientiert, was (wie oben erwähnt) problematisch ist.
Der Entwurfsprozess selbst beschreibt den Entwurf eines Hypertextes auf Basis dieser Primitive durch Definition mehrerer Schritte („ER-Design“, „Entity Design“, „Navigation Design“, „Conversion Protocol Design“, „User-Interface Screen Design“, „Runtime Behavior Design“, „Construction“ und „Testing and Evaluation“), die zwar teilweise durch Rückkoppelung miteinander verbunden sind, aber grundsätzlich in einem linearen Ablauf angeordnet sind. RMM behandelt in keiner Weise Themen der Benutzerorientierung, sondern geht davon aus, dass bekannte Methoden dieser Art zusätzlich integriert werden können [Isakowitz, Stohr & Balasubramanian 95].

OOHDM
OOHDM ist zwar grundsätzlich zunächst allgemein für Hypertexte entwickelt worden [Schwabe & Rossi 95], wobei aber zusätzlich eine spezielle Ausrichtung für die Belange des WWW in Form des OOHDM-WEB (vgl. [Schwabe & De Almeida Pontes 98]) entwickelt wurde.
Es werden grundsätzlich vier Schritte definiert, nämlich „Conceptual Design“, „Navigational Design“, „Abstract Interface Design“ und „Implementation“, die über einen inkrementellen Prozess miteinander verbunden sind, wodurch dieses Modell grundsätzlich ein evolutionäres Vorgehen unterstützt. Die Daten werden als Klassen modelliert (vgl. [Balasubramanian & Bashian 98, S. 109]), wodurch das Vorgehen einen objektorientierten Charakter bekommt.

4.4.4. Diskussion der Modelle

In Bezug auf die Brauchbarkeit der oben vorgestellten Modelle soll einleitend festgestellt werden, dass keines vollständig auf die in dieser Arbeit verfolgten Prinzipien zutreffend ist.
Zwar erscheint das oben beschriebene evolutionäre Vorgehensmodell als grundsätzlich geeignet, wobei es aber derart allgemeingültig beschrieben ist, dass sich daraus keine konkrete Schritte ablesen lassen. In dieser Hinsicht bieten die Hypertextmodelle RMM und OOHDM eine kleine Verbesserung, da dort auf hypertextspezifische Umstände, wie z.B. den Navigationsentwurf, eingegangen wird. Allerdings wird jeweils von einem deutlich datenorientierten Vorgehen ausgegangen [Scharl 99] und die Belange der Benutzer werden nur unzureichend berücksichtigt. So ist beispielsweise von partizipativen Methoden oder evaluativen Elementen nicht die Rede.
Zudem lassen die jeweils zugrundeliegenden Datenmodelle keine Ansätze z.B. in Bezug auf eine Hierarchie oder den in Kapitel 2.8 vorgestellten Aspekt der websitespezifischen Kommunikationsmittel erkennen. Außerdem kommt die stark strukturierte Modellierung der Daten nicht unbedingt einem evolutionären Vorgehen entgegen (s.o.), was in Bezug auf Benutzerorientierung von Notwendigkeit wäre.

Zusammengefasst verfolgen die hier behandelten Modelle die behandelte Thematik entweder zu abstrakt oder aber verfolgen eine grundsätzlich andere Richtung. Aus diesem Grund wurde für diese Diplomarbeit ein eigenes Datenmodell bzw. Websitemodell und ein spezielles Entwurfsprozessmodell entwickelt. Beide nehmen durchaus Bezug auf die hier vorgestellten, wobei entweder Teilaspekte übernommen oder aber genauer ausgearbeitet werden. Diese Modelle werden in Kapitel 4.8 bzw. Kapitel 5 vorgestellt.

4.5. Entwicklungsdokumente

Ein Entwicklungsprozess sieht grundsätzlich vor, dass bei dessen Beendigung der Entwurfsgegenstand fertiggestellt in Form eines bestimmten Mediums vorliegt. Eine Website wird in so einem Fall beispielsweise als HTML vorliegen und über das WWW zu erreichen sein.
Während des Entwicklungsvorganges werden Teil- oder Zwischenergebnisse anfallen, die ebenfalls in irgendeiner Form repräsentiert werden müssen. Dabei ist es nicht zwangsläufig nötig, dass diese Repräsentation der des entgültigen Ergebnisses entspricht. Mit anderen Worten: Zwischenergebnisse liegen nicht unbedingt in HTML vor, sondern können vielmehr in abstrakteren Darstellungen, wie zum Beispiel einer schriftlichen Ausführung oder einer Illustration, auftreten. Solch eine Darstellung soll allgemein als Dokument bezeichnet werden: „Ein Dokument ist ein Beleg, d.h. eine Repräsentation von einem Sachverhalt unter Verwendung von Zeichen. Im Kontext Softwareentwicklung dienen Dokumente zur Darstellung des Softwaresystems in seiner Dynamik und Statik und des Softwareprojekts. Dazu gibt es auch Dokumente auf der Meta-Ebene (z.B. Dokumente über Dokumentation).“ [Züllighoven et al. 98, S. 131].

Im Kontext eines Entwicklungsprozesses sind insbesondere die Entwicklungsdokumente interessant, die das System bei seiner Erst- und Weiterentwicklung repräsentieren [Züllighoven et al. 98, S. 139]. Entwicklungsdokumente sind somit zentraler Gegenstand im Entwicklungsprozess, weil sie das jeweilige „Produkt“ zu einem bestimmten Zeitpunkt repräsentieren und damit die Ergebnisse der jeweiligen Entwurfsschritte festhalten und verfügbar machen. In diesem Zusammenhang wird von einem „dokumentgetriebenen Entwicklungsprozess“ gesprochen, wenn über eine geeignete Auswahl von Dokumenten gewährleistet ist, dass alle relevanten Aspekte systematisch behandelt werden [Züllighoven et al. 98, S. 128]. Gemäß des in Kapitel 4.4.1 beschriebenen Autor-Kritiker-Zyklus stellen Entwicklungsdokumente zudem den Gegenstand der Kritik bzw. Diskussion dar, d.h. anhand dieser wird eine Bewertung des jeweiligen Entwicklungsstandes vorgenommen. Daraus ergeben sich ggf. Änderungen am Entwurf, die ihrerseits zu einer Modifikation der entsprechenden Entwicklungsdokumente führt.

Grundsätzlich könnte man annehmen, dass der Entwurfsgegenstand der Website sich mittels eines einzigen Entwicklungsdokumentes repräsentieren lassen könnte. Dass dies ein Irrtum ist, wird in [Rosenfeld 97] in Bezug auf die Informationsarchitektur erläutert: Eine Website stellt einen komplexen Informationsraum dar, dessen einzelne Aspekte sich aufgrund der begrenzten menschlichen Wahrnehmung nicht konzentriert darstellen lassen. Insofern müssen einzelne Aspekte herausgegriffen werden, um sie einzeln zu dokumentieren. Beispielsweise macht es Sinn, die Struktur einzelner Seiten einer Website unabhängig von deren konkretem Inhalt zu dokumentieren, da eine Kombination beider nicht mehr überschaubar wäre.

Beispiele für Entwicklungsdokumente im Kontext von Websites können Strukturdiagramme, Layoutskizzen oder HTML-Prototypen sein, mit denen Entwurfsentscheidungen, wie z.B. eine hierarchische Struktur oder darstellungsrelevante Aspekte, repräsentiert werden. Neben derartigen produktbezogenen Entwicklungsdokumenten gibt es grundsätzlich auch prozessbezogene, in denen z.B. der Projektablauf in Form eines Zeitplanes behandelt wird.

4.6. Entwicklungswerkzeuge

Es existieren eine Reihe von Softwareprodukten, die man im engeren Sinne als Werkzeuge zur Erstellung von Websites ansehen kann. Es stellt sich die Frage, inwieweit solche Werkzeuge dazu geeignet sind, auch den Entwurfsprozess eines webbasierten Informationssystems zu unterstützen. Um dieser Frage nachzugehen, sollen derartige Programme schematisch vorgestellt und in Bezug auf ihre Verwendbarkeit reflektiert werden. Dazu soll sich der in [Schwabe & De Almeida Pontes 98] vorgestellten Klassifikation in „Page Editors“, „Web Site Editors“ und „Web Site Building Environment“ bedient werden, die hier durch den Punkt der „Programmiersprachen“ erweitert wurde. Zwar können Werkzeuge nicht unbedingt als Werkzeug betrachtet werden, aber unter dem Gesichtspunkt eines Hilfsmittels macht diese Erweiterung durchaus Sinn.

Page Editors

Bei den „Page Editors“ handelt es sich um Werkzeuge, die zur Erstellung einzelner HTML-Seiten geeignet sind. Dazu lassen sich einzelne Elemente wie Text und Grafiken zusammenfügen und mit visuellen Mitteln wie Layout und Typographie versehen. Dabei stehen oftmals zwei grundsätzliche Sichten zur Verfügung, nämlich einmal die WYSIWYG-Umgebung, in der dies mittels komfortabler direkter Manipulation geleistet werden kann, und zum anderen in einer HTML-Sicht, die das direkte Erstellen von HTML ermöglicht.
Zwar wird das Setzen von Hyperlinks auf andere Seiten ermöglicht, aber im Prinzip wird keine bzw. wenig Unterstützung in Bezug auf eine Strukturierung dieser Seiten gegeben. Insofern kann von einer Modellierungeinheit „Website“ nur sehr bedingt die Rede sein. Andererseits ermöglichen sie aber den oftmals technisch unerfahrenen Benutzern die Erstellung von HTML-Seiten. Beispiele solcher Editoren stellen u.a. „Homesite“ von Allaire und „Hotmetal“ von Softsquad dar.

Web Site Editors

Editoren dieser Art beinhalten jeweils neben der Komponente eines „Page Editor“ einen übergeordneten Zugriff auf die Modellierungseinheit der Website. Das bedeutet im Wesentlichen, dass einzelne Pages zu einer Struktur angeordnet und diese websiteübergreifend manipuliert werden können. Vertreter dieser Kategorie von Editoren sind neben „Frontpage“ von Microsoft insbesondere „Fusion“ von NetObject, „Dreamweaver“ von Macromedia und „GoLive“ von Adobe.

Grundsätzlich verfolgen diese Programme ein gewisses Maß an Abstraktion von den technischen Umständen und Möglichkeiten von HTML. Dadurch bekommen diesbezüglich weniger versierte Benutzer die Möglichkeit zur Erstellung von Websites, ohne sich ausgiebig mit der Materie auseinandersetzen zu müssen. Die Schwierigkeit bei einer zu starken Abstraktion besteht aber darin, dass bestimmte Möglichkeiten von HTML keine Berücksichtigung mehr finden, bzw. dass eine Feinjustierung nicht möglich ist. Eine zu schwache Abstraktion hingegen setzt ein entsprechend größeres Maß an Wissen über die Materie voraus, was nicht zwangsläufig verlangt werden kann. Insofern befinden sich alle diese Programme in einem Spagat zwischen erfahrenen und unerfahrenen Entwicklern.

Neben der Editierungsmöglichkeit einzelner Seiten lassen diese Programme auch die Erstellung und Änderung der Struktur zu, indem die Seiten mittels direkter Manipulation zu einer Hierarchie zusammengefügt werden können. Dies schlägt sich auf

den einzelnen Seiten in Form von Hyperlinks nieder, die oftmals sogar als Navigationspanels automatisch eingefügt werden.
Unabhängig von einer einzelnen Seite können zudem visuelle Elemente definiert werden (z.B. Layout und Typographie), die dann auf alle Seiten der Website angewandt werden können und somit für ein konsistentes „Look & Feel“ sorgen.
Zusätzlich werden oftmals auch Funktionen angeboten, die eine Unterstützung beim Management des Entwicklungsprozesses darstellen. Zum Beispiel wird von „Frontpage“ rudimentär die Verwaltung einzelner Aufgaben für mehrere an einem Projekt arbeitende Personen unterstützt. Gängig ist auch eine Unterstützung beim Abgleich einer lokalen Version mit einer bereits im WWW publizierten.

Web Site Building Environment
„Web Site Building Environments“, die auch als „Document Management Systems“ (vgl. [Balasubramanian & Bashian 98]) oder „Content Management Systems“ (CMS) bekannt sind, sind weniger zur Entwicklung bzw. zum Entwurf von Websites als vielmehr für den Betrieb großer diesbezüglicher Projekte geeignet. Dazu unterstützen sie z.B. das kooperatives Arbeiten beteiligter Personen, indem sie beispielsweise ein Versionsmanagement und verteiltes Arbeiten ermöglichen. Dabei wird oftmals auf eine Abstraktion vom Zielmedium HTML gesetzt: Inhalte können von technisch unerfahrenen Redakteuren erstellt und verwaltet werden, die dann vom System nach bestimmten Regeln automatisiert zu HTML-Seiten konvertiert werden. Derartige Software, z.B. in Form des „StoryServer“ von Vignette oder „CoreMedia“ der CoreMedia-AG, ist mitunter recht aufwendig gestaltet und verlangt somit nach einer umfassenden Konfiguration und Administration.

Programmiersprachen
Neben den drei oben genannten Werkzeugtypen soll hier, wie erwähnt, noch die zusätzliche Kategorie „Programmiersprachen“ angeführt werden. Wie der Name schon vermuten lässt, handelt es sich dabei nicht um Applikationen im eigentlichen Sinne, sondern sie stellen eher Frameworks zur Erstellung dar. Dabei werden HTML-Seiten mittels eines, in der jeweiligen Programmiersprache formulierten, Algorithmus erzeugt, wobei z.B. mittels einer Datenbankabfrage bestimmte Daten mit in diese Seite einfließen können. Gängig ist beispielsweise auch, dass eine Seite in verschiedene Fragmente unterteilt wird, für die jeweils ein eigener Algorithmus zuständig ist. Auf diese Weise lassen sich wiederkehrende Elemente modularisieren, um z.B. eine konsistente Website zu erhalten.

Obwohl dies mit nahezu jeder Programmiersprache geleistet werden kann, so haben sich einige Programmiersprachen bzw. Umgebungen als besonders geeignet herausgestellt. In diesem Zusammenhang seien die Sprachen „Perl“ und „PHP“ sowie das JAVA-basierte Framework der „Servlets“ (vgl. Kapitel 8.1.3) genannt. Diese sind für den Einsatz dynamischer HTML-Seiten geeignet, d.h. sie stellen Lösungen für WWW-spezifische Technologien (z.B. Cookies) bereit. Entwicklungsumgebungen dieser Art verlangen allerdings Wissen in Bezug auf den allgemeinen Umgang mit Programmiersprachen sowie den Protokollen des WWW.

Fazit
Die hier vorgestellten Werkzeuge erscheinen als nicht übermäßig geeignet für den Entwurf eines Informationssystems im WWW. Zwar lassen sich mit allen den hier geschilderten Möglichkeiten letztendlich Websites erstellen, denen man das verwendete Werkzeug nicht unbedingt ansieht, aber gerade für den Entwurf liefern sie unzureichende Unterstützung. Grund dafür ist die mangelnde Orientierung an einem hie-

rarchischen Informationsraum, in dem Daten verschiedener Granularität baumartig konstruiert werden. Stattdessen findet zum Großteil eine Orientierung an der festen Granularität „HTML-Seite“ statt, in der die Sicht auf die eigentlichen Daten sehr verwässert wird. Dies liegt u.a. an der Eigenschaft von HTML, Inhalt und Darstellung zu kombinieren.
Am ehesten geeignet sind offensichtlich die „Web Site Editors“, in denen die geschilderte Problematik in Ansätzen gelöst wird. So z.B. in Form der oben erläuterten Möglichkeit, eine hierarchische Struktur zu erstellen oder aber ein einheitliches „Look & Feel“ zu etablieren. Gerade aber bei Elementen der Darstellung oder der Navigation werden keine ausreichenden Mittel geboten, diese einerseits websiteübergreifend zu definieren und andererseits genügend konfigurieren zu können. So bietet z.B. „Frontpage“ zwar die Möglichkeit, eine Navigation konsistent auf alle HTML-Seiten anzuwenden, wobei aber nur auf einen Standardsatz von Navigationselementen zugegriffen werden kann. Spezielle und bedarfsgerechte Navigationselemente lassen sich damit nicht definieren und verwenden.
Aus diesen Gründen bleibt der Nutzwert dieser Werkzeuge in Bezug auf die hier behandelte Problemstellung nur sehr eingeschränkt. Dies soll aber nicht bedeuten, dass sich nicht Teilaspekte dieser Werkzeuge sinnvoll verwenden lassen können.

4.7. Entwurfsmuster

Innerhalb einer Website werden sich bestimmte Elemente wiederholen, oder aber eine strukturelle Ähnlichkeit untereinander aufweisen. Ein Beispiel dafür stellt ein Navigationssystem dar, welches, bezogen auf alle Seiten der Website, nach einem konsistenten Schema funktioniert.
Aus Sicht des Entwicklers ist es sinnvoll, derartige „Muster“ einmalig abstrakt zu definieren und sie jeweils auf konkrete Fälle anzuwenden. Im obigen Beispiel würde dies z.B. für die Navigation bedeuten, dass nur das dahinterstehende Schema definiert wird, um es dann auf die einzelnen Seiten anzuwenden. Zum einen bedeutet ein derartiges Vorgehen eine Arbeitsersparnis für den Entwickler, weil Entwurfsentscheidungen wiederverwendet werden, zum anderen wird auf diese Art und Weise für Konsistenz gesorgt. In Bezug auf Benutzungsschnittstellen hat Konsistenz einen hohen Stellenwert (vgl. [Vogt 99, S. 9f]), weil der Benutzer dadurch verschont bleibt, die Verwendung bestimmter Bedienungselemente immer und immer wieder zu erlernen. Untereinander konsistente Elemente müssen vom Benutzer nur einmalig verstanden werden, so dass er bei wiederholter Konfrontation auf sein erlerntes Wissen zurückgreifen kann. Auf diese Weise werden seine kognitiven Ressourcen geschont, was prinzipiell eine Form der Benutzbarkeit ausmacht.

Aus dem Umfeld des Software Engineering sind solche Muster prinzipiell unter dem Begriff „Design Patterns“ („Entwurfsmuster“) bekannt und stellen eine anerkannte Methode für die Softwareentwicklung dar (vgl. z.B. [Gamma, Helm, Johnson & Vlissides 95]). Auch aus dem Umfeld der Hypertexterstellung ist dieses Prinzip bekannt und wird z.B. in [Nanard, Nanard & Kahn 98] oder [Rossi, Schwabe & Garrido 97] behandelt. In diesem Kapitel sollen einige Muster aufgezeigt werden, die in Bezug auf den Entwurf von Websites als relevant erscheinen. Ziel ist es, dass ein entsprechender Entwicklungsprozess von derartigen Mustern getrieben werden kann und somit einerseits Arbeitsersparnis durch Wiederverwendbarkeit und andererseits Konsistenz bestimmter Elemente entsteht.

Navigationssystem
Wie oben schon angedeutet, eignet sich die Navigation für die Betrachtung und den Entwurf als Muster. Eine Navigation wird bekanntermaßen über Hyperlinks und Kontextinformationen definiert, die sich an der hierarchischen Struktur orientieren. Aus Gründen der Benutzbarkeit sollte die Navigation konsistent sein [Fleming 98, S. 14f], d.h. auf jedem einzelnen Knoten in ähnlicher Art und Weise bzw. nach ein- und demselben Schema funktionieren. Dazu soll ein Navigationssystem als eine Art von Algorithmus gesehen werden, der sich auf jeden einzelnen Knoten anwenden lässt und in Abhängigkeit von der jeweiligen Position innerhalb der Struktur für konkrete Ergebnisse sorgt. Beispielsweise beschreibt die abstrakte Vorschrift "Setzte einen Hyperlink auf jeden untergeordneten Knoten und auf den übergeordneten Knoten" einen solchen Algorithmus, der sich auf alle Knoten anwenden lässt und somit für ein konsistentes Navigationssystem sorgt.

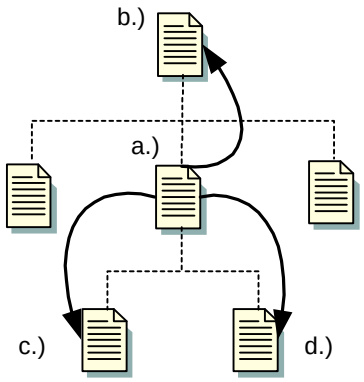

Abbildung 4.2: Obige Vorschrift wird auf Knoten a.) angewandt und bewirkt in Abhängigkeit zur Struktur (gestrichelte Linien) das Erzeugen von Hyperlinks (Pfeile) auf die untergeordneten Knoten c.) und d.), sowie den übergeordneten Knoten b.).

Neben der Wichtigkeit in Bezug auf Konsistenz ist hier auch der hohe Grad an Wiederverwendbarkeit entscheidend, da dies u.U. jeden einzelnen Knoten betrifft.

Dokumenttypen
Es kann durchaus vorkommen, dass auf einer Website einzelne Seiten existieren, die eine ähnliche inhaltliche Struktur aufweisen. Als Beispiel soll angenommen werden, dass auf einer Website eine Menge von Mitarbeitern dargestellt werden sollen. Jeder dieser Personen soll auf einem separaten Knoten präsentiert werden, die natürlich jeweils dieselbe Elemente, wie z.B. den Namen, ein Bild, eine Telefonnummer usw., beinhalten.
In diesem Zusammenhang soll der Begriff des „Dokumenttyps“ eingeführt werden: Unter einem Dokumenttyp soll eine Struktur verstanden sein, die als Grundlage für eine oder mehren Seiten dient. In einem solchen Dokumenttyp wird definiert, welche Art von Chunks für den jeweiligen Inhalt zulässig sind, bzw. aus welchen er sich zusammensetzt. In Bezug auf obiges Beispiel kann z.B. ein Dokumenttyp „Mitarbeiter“ definiert werden, der aus den Chunks „Name“, „Bild“, usw. besteht.
Das grundsätzliche Konzept der Typisierung ist u.a. bekannt aus vielen Programmiersprachen und stellt ein wichtiges Instrument des Software Engineering dar. Im Zusammenhang mit Variablen definieren „Typen“ eine Menge von Werten, die für die Variable dieses Typs zulässig ist. Analog verhält es sich mit Dokumenttypen: Ein Dokumenttyp definiert eine Menge von Inhalten, die für eine Seite dieses Typs zulässig

sind. Ein wesentlicher Unterschied besteht allerdings darin, dass sich eine Typisierung von Dokumenten auf semantischer Ebene abspielt.

Das Konzept der „Dokumenttypen“ bezieht sich grundsätzlich immer nur auf ein komplettes Dokument, wie z.B. eine komplette HTML-Seite, und soll an dieser Stelle etwas weiter gefasst werden. Dahinter steht der Gedanke, dass sich mehrere Dokumente zwar nicht unbedingt komplett über denselben Dokumenttyp beschreiben lassen, sich aber deren einzelne Chunks strukturell ähneln. So zum Beispiel „Copyrightinformationen“, wie sie sich auf einigen Websites am unteren Ende jeder einzelnen Seite befinden.
Auf diese Weise kann die Struktur innerhalb einzelner Chunks als ein Muster betrachtet und wiederverwendet werden. Obwohl in so einem Fall anstatt von „Dokumenttypen“ vielleicht besser von „Chunktypen“ gesprochen werden kann, so soll im Folgenden ersterer Begriff verwendet werden.

Templates
Das Konzept der Templates stellt eine Methode dar, mit der abstrakt Darstellungselemente definiert werden können, die dann praktisch als Schablonen für die Inhalte dienen (vgl. [Clark 99] und [Nanard, Nanard & Kahn 98, S. 15]). Beispielsweise kann über ein Template das Layout einer Seite beschrieben werden, in das die einzelnen Inhalte einfließen können. Bei der Anwendung einer solchen Schablone auf einen konkreten Inhalt werden bestimmte Platzhalter in der Schablone durch bestimmte Chunks ersetzt, wodurch eine konkrete visuelle Darstellung des Inhalts erzeugt wird.
Eine derartige Schablone kann auf alle Inhalte angewandt werden, die einer bestimmten Struktur (z.B. demselben Dokumenttyp) folgen, wodurch diese dann eine sich entsprechend ähnelnde visuelle Darstellung bekommen. Somit stellen Templates Muster in Bezug auf die Darstellung dar.
Analog zu dem oben skizzierten Verhältnis von „Dokumenttyp“ und „Chunktyp“ kann hier eine Unterscheidung zwischen „Template“ und „Fragment“ getätigt werden. Ein Fragment stellt einen Ausschnitt aus einer Seite bzw. einem Template dar (vgl. [Challenger, Iyengar & Witting 00]) und sorgt somit analog für die Darstellung eines speziellen Chunks.

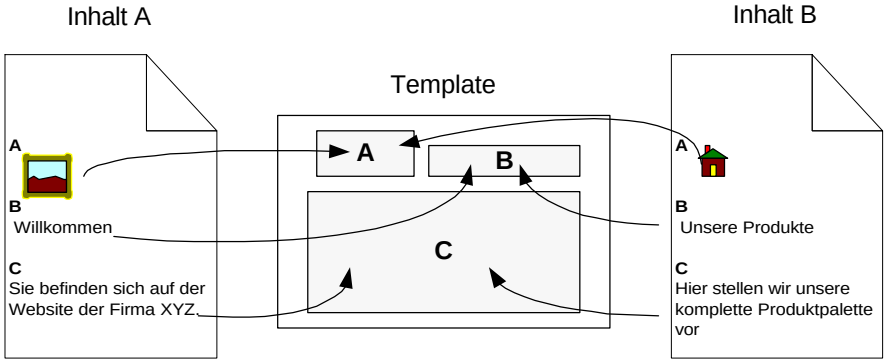

Abbildung 4.3: In einem Template (Mitte) wird ein Layout über die Position und Größe von drei Feldern (A,B,C) definiert. Zwei verschiedene Inhalte können ein- und dasselbe Template verwenden, indem ihre Chunks (A,B,C) jeweils in den entsprechenden Felder platziert werden.

Inhalte
Neben den Dokumenttypen, in denen die Struktur von Inhalten beschrieben wird, werden sich u.U. auf einer Website auch bestimmte konkrete Inhalte wiederholen. Ein Beispiel stellen die Labels innerhalb eines Navigationssystems (s.o.) dar: Aus Gründen der Konsistenz werden sich über mehrere Seiten hinweg bestimmte Hyperlinks wiederholen, die jeweils mit einem Label versehen sind. Damit der Benutzer nicht verwirrt wird, sollten sich Labels der Hyperlinks auf jeweils gleichen Seiten wiederum gleichen. Ein anderes Beispiel stellen Copyrightinformationen dar, wie sie sich auf bestimmten Websites auf jeder einzelnen Seite finden. Der Text dieser Copyrightinformation wiederholt sich dabei oftmals auf jeder Seite. Sich derartig wiederholende Inhalte können wiederum als Muster aufgefasst werden und nach einmaliger Definition wiederverwendet werden.

4.8. Ein benutzungsorientiertes Prozessmodell

Diese Diplomarbeit hat sich unter anderem zum Ziel gesetzt, einen Entwurfsprozess darzulegen, mit dem ein informationsgetriebenes Vorgehen (vgl. Kapitel 2) mit Bezügen zur Benutzerorientierung (vgl. Kapitel 3) beschrieben wird. Ein solcher Prozess soll modellhaft in diesem Kapitel skizziert werden, während eine ausführliche Schilderung der einzelnen Schritte in den Kapiteln 6 und 7 stattfinden wird. Ziel soll es hier sein, eine Menge von Vorgaben bzw. Vorüberlegungen zu diesem Prozess zu formulieren.

Evolutionäres Vorgehen
In Kapitel 4.4.1 wurden diverse Prozessmodelle aus dem Software Engineering vorgestellt, die sich prinzipiell auch auf das Web-Engineering übertragen lassen. Dabei ergibt sich in Hinblick auf die Aufgabenstellung (vgl. Kapitel 1.2) und den daraus entstehenden Implikationen quasi zwangsläufig die Verwendung eines evolutionären Prozessmodells. Zum einen stellt dies eine Notwendigkeit in Bezug auf eine Benutzerorientierung dar (vgl. Kapitel 3.1) und zum anderen ist gerade die Umgebung des WWW eine ideale Vorraussetzung dafür, wie [Norton 99] ausführt: Der Grund dafür liegt in der Natur des WWW, in der Änderungen an einer bereits online geschalteten Version ohne Zeitverzug global veröffentlicht werden können. Damit können auch „Vorabversionen“ der Kritik einer breiten Massen ausgesetzt werden und als Basis für eine Überarbeitung dienen.
Weiterhin wird eine Website in der Regel weiterentwickelt, d.h. es werden neue Informationen bereitgestellt oder ältere aktualisiert. Beispiele für diesen Sachverhalt finden sich insbesondere auf Websites, in denen permanent Neuigkeiten bereitgestellt werden[11]. In diesem Zusammenhang sieht Nielsen den jeweils aktuellen Stand einer Website als Prototyp des nächsten Durchlaufes [Nielsen 98b].

Entwurfsschritte
Im Prozessmodell werden verschiedene Entwurfsschritte definiert, in denen jeweils Teilaspekte des gesamten Entwurfsgegenstandes behandelt werden. Die Definition dieser Schritte orientiert sich an den Ergebnissen der letzen Kapitel, wobei die vielen verschiedenen Quellen dieser Diplomarbeit als Inspiration gegolten haben. Diese sollen Schematisch in folgenden Ausführungen bzw. der Abbildung vorgestellt und erläutert werden:

[11] Als Beispiel mag die Onlineausgabe des Nachrichtenmagazins „Der Spiegel“ unter http://www.spiegel.de gelten.

Bei der „Analyse“ werden die Ziele und die Zielgruppen der Website definiert und eine Bedarfsanalyse anhand der Benutzer durchgeführt. Ziel dieser ersten Phase ist es, die Ausgangssituation zu untersuchen und genügend Informationen darüber zu erhalten, wo aktuelle Probleme und Bedürfnisse liegen. Aufgrund dieser Vorgaben können dann die Entwurfsentscheidungen innerhalb der folgenden Phase „Entwurf“ getroffen werden. Es werden in dieser Phase also einzelne Aspekte einer Website entworfen, deren Kombination die komplette Website spezifiziert.

Beim „Entwurf“ wird über die „Konzeption“ eine sehr grobe Entwurfsrichtung vorgegeben, die dann in den einzelnen Schritten „Inhalte“, „Struktur“, „Navigation“, „Labels“ und „Darstellung“ verfeinert und mittels „Prototyping“ evaluiert wird.
Im Schritt „Inhalte“ werden prinzipiell die Inhalte einzelner Seiten festgelegt, die ihrerseits im Schritt „Struktur“ zu einer hierarchischen Struktur zusammengefügt werden. Die Schritte „Navigation“ und „Labels“ befassen sich mit dem Entwurf eines Navigationssystems, d.h. es werden konsistente Schemata entworfen, nach denen die Hyperlinks zwischen den Seiten gesetzt und gestaltet werden. Im Schritt „Darstellung“ werden zuguterletzt alle visuellen Aspekte der Website, wie z.B. das Layout, festgelegt.
Diese Unterteilung in genau diese genannten Schritte wurde durch [Rosenfeld & Morville 98], [Shiple 98] und auch [Fleming 98] inspiriert und erscheint in Bezug auf Arbeitsteilung sinnvoll und notwendig. Auf diese Weise kann in einem Team von Entwicklern, in dem jeder einzelne spezielle Fähigkeiten mitbringt, zum Teil unabhängig voneinander an der Website gearbeitet werden. Oftmals ist es sogar möglich, die Ergebnisse dieser Schritte jeweils für sich zu evaluieren und Anhaltspunkte für eine Überarbeitung zu bekommen.

Das Auftrennung in separate Bearbeitungseinheiten ist durchaus üblich, wie anhand der häufig beschriebenen Trennung von „Inhalt“ und „Darstellung“ beispielhaft gesehen werden kann (vgl. [Gould, Boies & Lewis 91, S. 80] oder [Nanard, Nanard & Kahn 98, S. 15]). Allerdings sind derartige Trennungen nicht immer offensichtlich durchzuführen, so logisch sie oftmals auch erscheinen. Dies wird alleine schon dadurch deutlich, dass jegliche Dokumentation eines Inhalts, sei es als Text, Bild o.ä., im Prinzip schon eine Darstellung des dahinterstehenden Gedanken ist. So kann beispielsweise ein einzelnes fett gedrucktes Wort als inhaltliche Betonung gesehen werden, andererseits aber auch als darstellendes typographisches Element. Zur Veranschaulichung dieser Problematik sei beispielhaft auch auf die Ausführungen in [Siegel 97] verwiesen.

Insofern muss beim Entwurf der Einzelaspekte immer berücksichtigt werden, dass es sich jeweils um Teile eines großen Ganzen handelt, die sich wechselseitig beeinflussen. Auch wenn bei der detaillierten Ausführung der Einzelschritte nicht immer explizit darauf hingewiesen wird, so soll davon ausgegangen werden, dass alle Einzelergebnisse zu einem Ganzen integriert werden müssen. Dieses Gesamtergebnis muss evaluiert werden, was im Schritt „Prototyping“ geschieht. Aus diesem Grund sollen die Einzelschritte als lose gekoppelt gesehen werden, die mittels evolutionärem Vorgehen als „Autor-Kritiker-Zyklus“ wiederholt durchlaufen werden. Zusätzlich werden sich im Laufe der Entwicklung u.U. auch weitere Umstände und Erkenntnisse ergeben, die auch ein wiederholtes Durchlaufen der Analysephase bedingen.

Am Ende jedes Entwurfszyklus kann der augenblickliche Stand der Entwicklung in eine „Implementation“ überführt werden, was allerdings nicht Gegenstand dieser Dip-

lomarbeit sein soll. In Zusammenhang mit einem evolutionärem Vorgehen fällt es allerdings auch schwer, von den grundsätzlichen Phasen „Entwurf“ und „Implementation“ zu sprechen, da diese eher den konventionellen Prozessmodellen entnommen sind. In diesem Kontext sei mit dem „Entwurf“ die Spezifikation einer Website gemeint, auf deren Basis eine zügige „Implementation“ geleistet werden kann. Mit „Implementation“ ist seinerseits die Transformation der Ergebnisse nach HTML bzw. die Anpassung an das Zielsystem des Webservers gemeint.

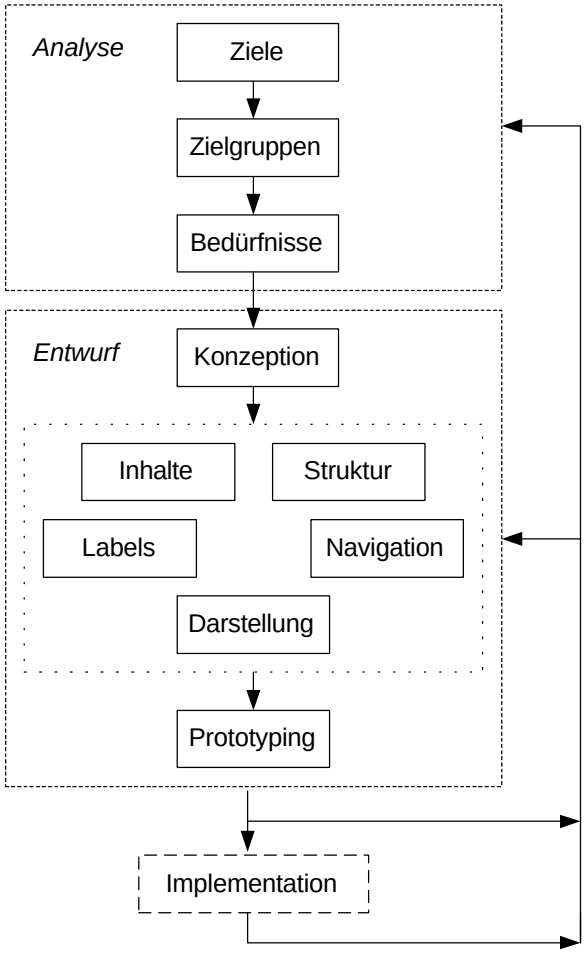

Abbildung 4.4: Das schematisierte Prozessmodell, in dem die Phasen „Analyse“, „Entwurf“ und „Implementation“ zyklisch miteinander verbunden sind. Die hier behandelten ersten beiden Phasen bestehen ihrerseits aus mehreren Schritten.

Grundsätzlich unterscheidet man in einem Entwicklungsprozess prozessorientierte und produktorientierte Tätigkeiten, wobei sich die produktorientierten auf den konkreten Entwicklungsgegenstand, also auf die Website, beziehen. Dies sind Aktivitäten wie z.B. die hier geschilderten Phasen „Analyse“ oder „Entwurf“. Die prozessbezogenen Tätigkeiten hingegen sind kooperativer und koordinativer Natur und beziehen sich daher eher auf den Ablauf der Entwicklung anstatt direkt auf das Produkt [Züllighoven et al. 98, S. 126]. Auch wenn die prozessorientierten Tätigkeiten von großer Wichtigkeit sind, so soll das Augenmerk dieser Diplomarbeit auf den produktorientierten liegen. Natürlich gibt es viele Überschneidungen, die eine klare Trennung erschweren, weshalb solche Tätigkeiten u.U. zwar implizit angesprochen aber nicht vertiefend behandelt werden.

Entwicklungsdokumente

Wie oben ausgeführt wurde, soll in dieser Arbeit ein Entwurfsprozess dargelegt werden, der der Spezifikation einer Website dient. Eine derartige Spezifikation wird über Entwicklungsdokumenten (s. Kapitel 4.5) festgehalten, weshalb das Aufzeigen entsprechender Dokumente innerhalb dieser Arbeit angestrebt wurde. Soweit wie möglich wurde dabei auf Entwicklungsdokumente zurückgegriffen, die in der Literatur beschrieben werden und sich somit offenbar bewährt haben. Allerdings war es in Bezug auf den hier verwendeten speziellen Entwurfsprozess nicht immer möglich, ein geeignetes Dokument zu finden. In so einem Fall wurden neue Ideen für entsprechende Entwicklungsdokumente vorgestellt bzw. skizziert.

Insgesamt wurde nach Möglichkeiten gesucht, die Ergebnisse systematisch und eindeutig festzuhalten, was oftmals einer abstrakten Form bedurfte. Diese Form mag nicht jedermann unmittelbar zugänglich sein, sondern bedarf mitunter einer gewissen Einarbeitung. Dies ist deshalb ein wenig problematisch, da das Feld der Website-Entwicklung interdisziplinär ist und somit Personen involviert sind, die grundsätzlich wenig Übung in Bezug auf abstrakte Spezifikationen haben.

5. Ein informationsorientiertes Website-Modell

In Kapitel 4.4.2 wurden festegestellt, dass das Vorhandensein eines Datenmodells innerhalb des Hypertextentwurfes von Notwendigkeit ist. Daher wurde in dieser Diplomarbeit ein Modell von einer Website entwickelt, das als Grundlage für den in den folgenden Kapiteln dargestellten Entwurfsprozess dient. Dabei wurde Bezug auf die Ergebnisse aus Kapitel 2 genommen, in denen eine Website unter dem Gesichtspunkt der Information beschrieben wird und außerdem auf die technischen Umstände von HTML eingegangen wird.
Daraus ist ein Modell entstanden, dass einerseits eine abstrakte Sicht auf einen heterogenen und hierarchischen Informationsraum liefert und andererseits auch technische Umstände, d.h. HTML-relevante Themen, berücksichtigt. Grund für letzteres ist die damit verbundene Möglichkeit, den Entwurfsgegenstand leichter nach HTML überführen zu können. Wenn nämlich von vornherein auf diese Umstände eingegangen wird, dann kann die HTML-Repräsentation praktisch direkt abgelesen werden, ohne vorher eine Konvertierung durchführen zu müssen.

5.1. Einleitung

Eine Website soll im Wesentlichen als ein hierarchisch gegliederter Informationsraum gesehen werden, in dem die einzelnen Inhaltselemente („Chunks“) baumartig angeordnet sind. Die Art und Weise, wie diese Anordnung definiert ist, bezieht sich auf die Bedeutung der einzelnen Chunks und ihrer inhaltlichen Beziehung zueinander. So definiert der Wurzelchunk ein Thema, das in seinen Einzelaspekten über die nachfolgenden Chunks zunehmend verfeinert wird. Das bedeutet also, dass jeder untergeordnete Chunk eine Verfeinerung des Themas darstellt und selbst wieder das übergeordnete Thema seines untergeordneten Teilbaumes definiert. Jeder Chunk kann somit als „Übersicht“ über seine Nachkommen aufgefasst werden. Die Reihenfolge der Nachkommen beschreibt dabei grundsätzlich die Bedeutung der damit verbundenen Inhalte untereinander. Diese Vorstellungen sollen beispielhaft anhand von folgender Abbildung illustriert werden:

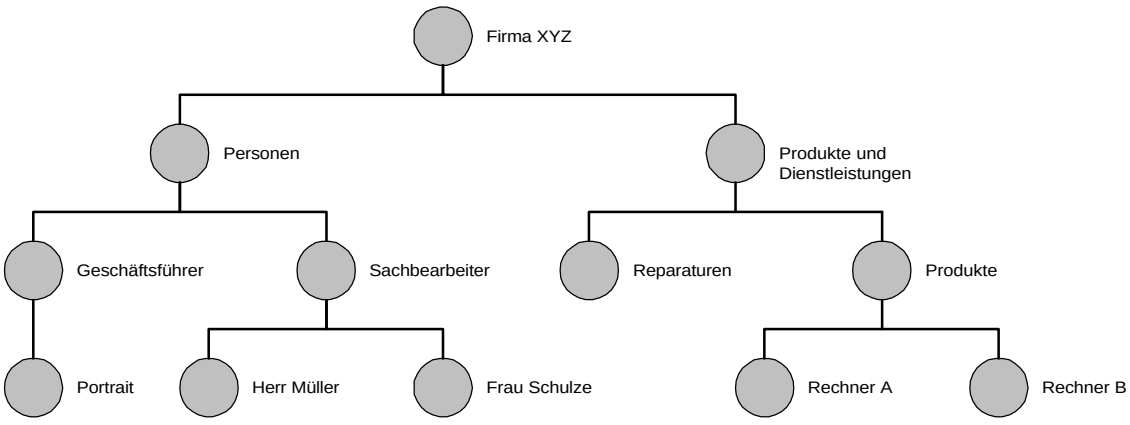

Abbildung 5.1: Ein fiktiver Informationsraum, bestehend aus einer hierarchischen Anordnung von Chunks. Der oberste Chunk „Firma XYZ“ definiert das Thema des Informationsraumes, nämlich die Präsentation einer (fiktiven) Firma. Ein Aspekt dieser Organisation stellen die dort beschäftigten „Personen“ dar, die in „Geschäftsführer“ und „Sachbearbeiter“ verfeinert werden. Dabei wird der Instanz des „Geschäftsführers“ über die Reihenfolge eine höhere Bedeutung beigemessen, als der der „Sachbearbeiter“. Der Chunk „Sachbearbeiter“ dient dabei als Übersicht über die Inhalte „Herr Müller“ und „Frau Schulze“.

Bei der Überführung eines derartig strukturierten Informationsraumes in eine Website wird ein Knoten bzw. eine Seite inhaltlich dadurch festgelegt, indem ein oder mehrere Chunks als zusammengehörig erklärt werden. Genauer gesagt, werden jeweils Teilbäume durch sog. „Chunking“ unter einer anderen Granularität betrachtet und repräsentieren dadurch den Inhalt einer HTML-Seite. Die Struktur innerhalb dieser Seite entspricht dabei der des ursprünglichen Teilbaumes.
Die Festlegung dieser Teilbäume soll zunächst als willkürlich angesehen werden, da damit lediglich die Menge der gleichzeitig auf einer Seite dargestellten Inhalte beschrieben wird und sich an der Struktur im Prinzip nichts ändert.

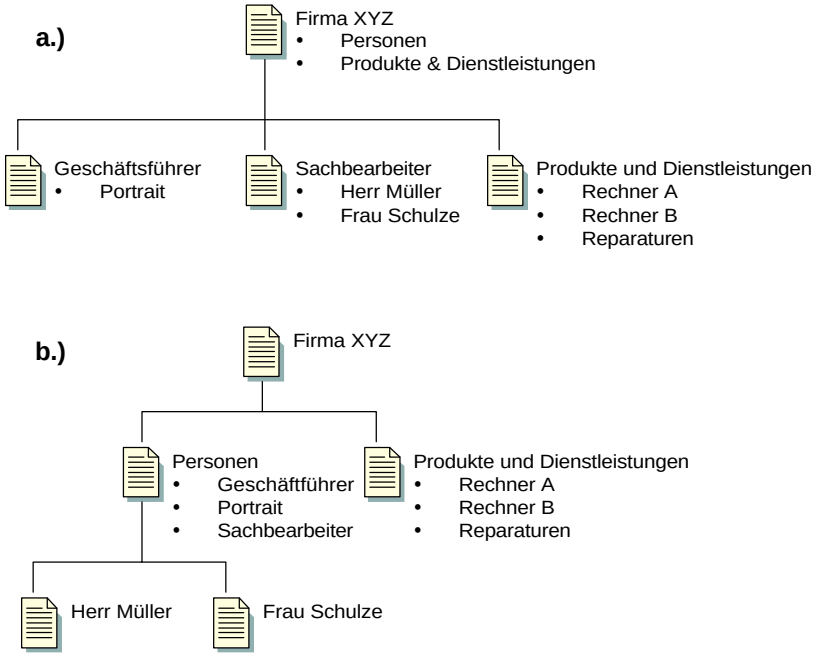

Abbildung 5.2: Zwei beispielhafte Möglichkeiten, wie die Chunks aus obigem Informationsraum zu Seiten zusammengefasst werden können.

Durch dieses „Chunking“ entsteht ein Gebilde, in dem „Inhaltseinheiten“ hierarchisch miteinander verbunden sind und somit wiederum in Beziehung zueinander stehen. Dieses gesamte Gebilde soll im Folgenden als „Struktur“ bezeichnet werden, während jede einzelne Einheit als ein „Inhalt“ gelten soll. Zusätzlich soll hier ein weiterer Aspekt einer Website, nämlich die „Darstellung“, eingeführt werden: Dort wird für jeden Inhalt beschrieben, wie dessen Struktur in Form einer HTML-Seite repräsentiert werden soll.
Diese drei Aspekte, nämlich „Struktur“, „Inhalt“ und „Darstellung“, sollen im Folgenden über ihre Grundbausteine genauer beschrieben werden:

5.2. Struktur

Eine implementierte Website kann bekanntermaßen als eine Sammlung von HTML-Seiten, Bildern u.ä. gesehen werden, die über Hyperlinks zu einer hierarchischen Struktur miteinander verbunden sind. Dieser Sachverhalt soll abstrahiert werden, indem die drei Bausteine „Page“, „Object“ und „Container“ eingeführt und dabei erläutert werden soll, wie aus diesen eine Struktur konstruiert werden kann.

Page
Eine Page stellt einen Knoten innerhalb eines Hypertextes dar und repräsentiert damit prinzipiell eine HTML-Seite. Im Unterschied dazu, soll die Page aber ausschließ-

lich die inhaltlichen Aspekte repräsentieren und somit keine Darstellungselemente, wie z.B. das Layout, beinhalten. Damit wird auf ein Konzept Bezug genommen, das sich beispielsweise in der Entwicklung von XML niedergeschlagen hat (vgl. Kapitel 8.1.1).
Zu einer vollständigen HTML-Seite kann eine Page erst dann werden, wenn sie mit einer Darstellung (s. Kapitel 5.4) kombiniert wird. Insofern stellt sie nur ein abstraktes inhaltliches Konstrukt dar, welches grob gesehen aus einer hierarchischen Struktur von Chunks besteht (s.o.). Eine genauere Beschreibung der verschiedenen inhaltlichen Aspekte findet im folgenden Abschnitt statt und soll somit hier nicht weiter vertieft werden. Allerdings soll hier schon erwähnt werden, dass sich innerhalb von Pages Hyperlinks definieren lassen, so dass damit der wahrscheinlich wichtigste Aspekt eines Hypertextes ausgedrückt werden kann. Somit soll die Page als das zentrale Element innerhalb dieses Modells gesehen werden.

Object
Neben den HTML-Seiten beinhaltet eine Website auch andere Medien, wie z.B. Bildern, Animationen usw. Diese Medien können entweder separat als Hypertextknoten dienen, oder aber sie werden in eine HTML-Seite eingebettet (vgl. Kapitel 2.2). Als übergeordnetes Konstrukt, welches alle diese Medien auf einen Nenner bringt, soll das „Object“ dienen. Damit wird prinzipiell nur ausgedrückt, dass es sich damit um keine HTML-Repräsentation, sondern um eines der übrigen Objekte des WWW handelt.
Die Tatsache, dass ein Object nicht in HTML beschrieben ist, impliziert, dass von ihnen auch keine Hyperlinks ausgehen können. Insofern lassen sich ausschließlich mit Objects auch keine Hypertexte beschreiben, sondern erst im Zusammenspiel mit den übrigen Elementen entfalten sie diese Bedeutung. Im Gegensatz zu Pages haben Objects ihre Darstellung prinzipiell integriert und lassen sich inhaltlich auch nicht weiter unterteilen, so dass sie als atomare Einheiten gesehen werden können.

Einerseits lassen sich Objects mittels einer Verknüpfung in Pages einbetten und erscheinen dadurch aus Sicht des Benutzers als integrierter Bestandteil dieser Page. Andererseits wird das Object durch das Setzen eines Hyperlinks von einer Page *auf* ein Object zu einem eigenständigen Hypertextknoten. Genau genommen handelt es sich bei diesem Knoten um ein Blatt, da ja keine Hyperlinks von einem Object ausgehen können.

Im Prinzip werden Objects durch einen Strom von Binärdaten definiert, die zusammen mit einer zusätzlichen Kennung, dem sog „Mime-Type“ (vgl. [Freed & Borenstein 96]), die Identifikation als ein bestimmter Medientyp zulassen und somit von einem Webbrowser dargestellt werden können. Beispiel: Die Kennung „image/gif“ identifiziert die Binärdaten als ein Bild im „GIF-Format“ und kann vom Webbrowser als solches dargestellt werden. Es soll davon ausgegangen werden, dass Objects ebenso wie Pages einen bedeutungsvollen Inhalt darstellen, wie es beispielsweise in Kapitel 2.8.2 in Bezug auf Bilder beschrieben ist.
Zusätzlich zu den Binärdaten können Objects auch weitere Inhalte haben, die hier als „Attribute“ bezeichnet werden sollen und als allgemeineres Konzept im folgenden Abschnitt vorgestellt werden.

Container
Wie schon in Kapitel 2.5 erläutert wurde, beschreibt eine Struktur die Art und Weise, wie die einzelnen Knoten eines Hypertextes in inhaltlicher Beziehung zueinander stehen. Diese Struktur wird zwar letztendlich durch Hyperlinks repräsentiert, was

aber in diesem Zusammenhang zunächst keine Rolle spielen soll. In Kapitel 2.6 wurde weiterhin festgelegt, dass die Hierarchie als die grundlegende Struktur einer Website angesehen werden soll und somit mit dem Begriff „Struktur“ immer „hierarchische Struktur“ gemeint sein soll.
Anstatt die Struktur über (strukturelle) Hyperlinks zu beschreiben, soll dies über die Verwendung des „Containers“ geschehen: Zunächst kann ein Container als Behälter gesehen werden, in dem Pages, Objects und wiederum Container gesammelt bzw. gruppiert werden. Dadurch, dass sich Container auch schachteln lassen, besteht die Möglichkeit, eine hierarchische Struktur zu konstruieren.

Dadurch entsteht eine Eltern-Kind-Beziehung zwischen den Elementen „Page“, „Object“ und „Container“: Jedes dieser Elemente hat einen Vorgänger (sofern einmal von der „Homepage“ abgesehen wird), während Container auch einen oder mehrere Nachfolger haben können.
Dabei sollen zwei Fälle unterschieden werden: Einerseits kann der Container zur ausschließlichen Gruppierung verwendet werden, zum anderen kann er zusätzlich auch zur Übersicht über die so gruppierten Elemente dienen. Im letzten Fall stellt der Container gleichzeitig eine Page dar, die ihrerseits dann auch Nachkommen hat. Das bedeutet, dass eine Page ebenfalls einen Behälter bereitstellt, in dem sich wiederum nachfolgende Elemente gruppieren lassen. Dies lässt sich so interpretieren, dass jedes dieser Nachkommen eine Verfeinerung der Page darstellt und somit als ein inhaltlicher Bestandteil dieser Page gesehen werden kann.

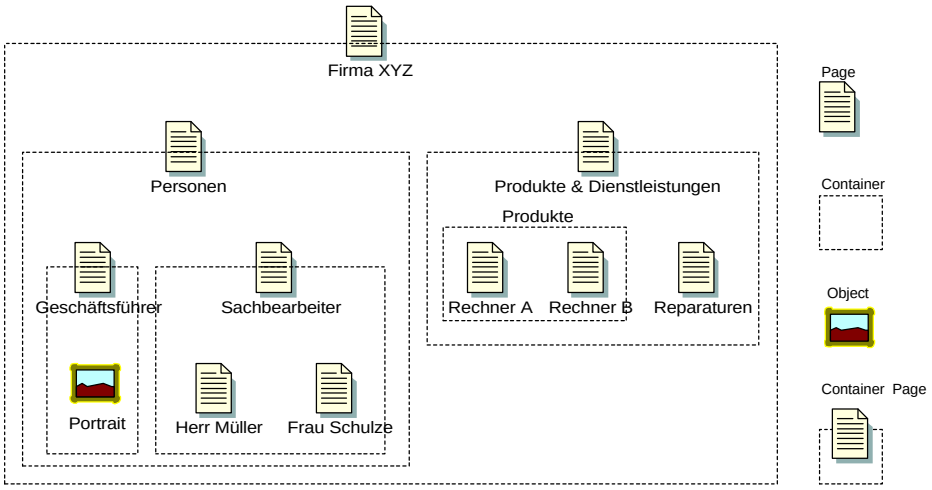

Abbildung 5.3: Eine mögliche Repräsentation des Informationsraumes aus Abbildung 5.1 mittels Pages, Objects und Container.

Unabhängig davon, ob es sich nun um einen „reinen“ Container oder aber um eine „Container Page“ handelt, sei festgestellt, dass der Container eine inhaltlich Bedeutung besitzt: Er kann immer als eine inhaltliche Abstraktion seiner enthaltenden Elemente gesehen werden. Wenn man sich verdeutlicht, dass ein Container Elemente einer Gruppe ähnlicher Inhalte repräsentiert (vgl. Kapitel 2.5), wird dieser Umstand umso deutlicher. Deshalb ist es nur logisch, dass auch Container durch zusätzliche Attribute beschrieben werden können, wie z.B. durch ein Label.

Zwar stellt ein Container, genau wie Objects und Pages auch, einen Baustein für die Erstellung von Websites dar, wobei er aber im Gegensatz dazu nichts „Gegenständliches“ repräsentiert, d.h. dass er aus Sicht des Benutzers nie direkt sichtbar wird.

5.3. Inhalt

Im vorherigen Kapitel wurden die zwei Elemente „Page“ und „Object“ definiert und diese als Träger von Inhalten identifiziert. Objects sollen vom inhaltlichen Standpunkt aus als atomare Einheiten betrachtet werden, d.h. dass sie sich nicht in kleinere Chunks unterteilen lassen. Die Pages hingegen setzen sich ihrerseits aus mehreren Chunks zusammen, die daher in diesem Kapitel genauer definiert werden sollen. Dazu sollen die möglichen Chunks, aus denen sich eine Page zusammensetzen kann, unter zwei verschiedenen Aspekten, nämlich „Datentyp“ und „Inhaltstyp“ betrachtet werden. Jeder einzelne Inhalt lässt sich gemäß dieser zwei Dimensionen charakterisieren und vermittelt damit eine bestimmte Sicht auf die Daten einer Website.

5.3.1. Datentypen

Die einzelnen Elemente einer Page, d.h. ihre Chunks, lassen sich grundsätzlich jeweils einem der drei Datentypen Text, Hyperlink oder Object zuordnen. Die Definition dieser Datentypen bezieht sich auf die Vorgaben aus HTML und spiegelt somit die technischen Möglichkeiten wieder.

Text
Mit „Text“ ist jenes schon in Kapitel 2.8.1 beschriebene Mittel gemeint, Sachverhalte über menschliche Sprache in Form von Buchstaben auszudrücken. Texte weisen für gewöhnlich eine hierarchische (Unter-)Struktur auf, die sich z.B. in Form von Strukturelementen wie „Wort“, „Satz“, „Absatz“, „Kapitel“ usw. oder auch „Überschrift“, „Einleitung“ usw. äußern. Derartige Elemente können Inhalte sowohl syntaktisch (z.B. besteht ein „Wort“ aus der begrenzten Aneinanderreihung von „Buchstaben“), als auch semantisch (z.B. eine „Einleitung“ dient als Einführung in ein Thema) beschreiben. Eine Unterteilung von Texten findet wiederum über die hier beschriebenen Bausteine „Hyperlink“, „Object“ und „Text“ statt, so dass ein Text einen rekursiven Träger von Inhalten darstellt.

Hyperlink
Obwohl ein Hyperlink im Grunde eine „Kante“ zwischen *zwei* Pages bzw. einer Page und einem Object darstellt, so soll er in diesem Modell als inhaltlicher Bestandteil der Ausgangspage gelten. Wie in Kapitel 2.6 beschrieben wurde, lassen sich Hyperlinks in strukturelle und in assoziative untergliedern. Während die assoziativen Hyperlinks beliebige Querverweise darstellen, folgenden die strukturellen Hyperlinks einer klaren hierarchischen Struktur und sollen somit explizit der „Navigation“ zugerechnet werden (s. Kapitel 2.7).

Die Granularität eines Hyperlinks ist über dessen Komponenten „Referenz“ und „Label“ festgelegt: Die Referenz stellt den eigentlichen Verweis auf eine andere Page (bzw. einem Object) dar und wird über eine URL (vgl. Kapitel 2.2) repräsentiert. Weiterhin besteht ein Label entweder aus einem „Text“, einem „Icon“ oder einer Kombination aus beiden.

Object
Mit „Objects“ sind die im obigen Abschnitt beschriebenen Medien gemeint, die an dieser Stelle in Bezug auf eine Einbettung betrachtet werden sollen. Das bedeutet, dass die Objects zwar als separate Elemente definiert werden, innerhalb einer Page aber mittels einer URL repräsentiert und eingebunden werden. Somit erscheint ein Object aus Sicht eines Betrachters als elementarer Bestandteil einer Page.

5.3.2. Inhaltstypen

Im Gegensatz zu den „Datentypen“, in denen im Prinzip eine syntaktische Unterscheidung der Inhalte vorgenommen wurde, soll mit den „Inhaltstypen“ eine Betrachtung in Hinsicht auf ihre Semantik geleistet werden. Dazu soll davon ausgegangen werden, dass sich die Inhalte einer Page grundsätzlich in „Informationselemente“, „Navigationselemente“ und „Attribute“ unterteilen lassen.

Informationselemente
Unter Informationselementen sollen solche Inhalte verstanden werden, die potentiell eine „echte“ Information darstellen, also etwas sind, wonach von Benutzern gezielt gesucht wird und was den eigentlichen Wert einer Website ausmacht. Dies kann beispielsweise eine „Produktbeschreibung“ sein, oder aber die „Telefonnummer einer bestimmten Person“. Mittels Querverweisen, also assoziativen Hyperlinks, kann dabei Bezug auf andere Informationselemente genommen werden.

Navigationselemente
Demgegenüber sind die „Navigationselemente“ solche, die den Benutzer dabei unterstützen, zu einer bestimmten Information zu gelangen. Dabei handelt es sich um strukturelle Hyperlinks, die im Zusammenspiel mit Kontextinformationen die Orientierung und die Bewegung im Hypertext ermöglichen (vgl. Kapitel 2.7). Zwar repräsentieren Navigationselemente potentiell ebenfalls Informationen, nämlich bezüglich der „Nievergeltschen Fragen“, wobei sie aber in Bezug auf eine Website eher ein Mittel zum Zweck darstellen.

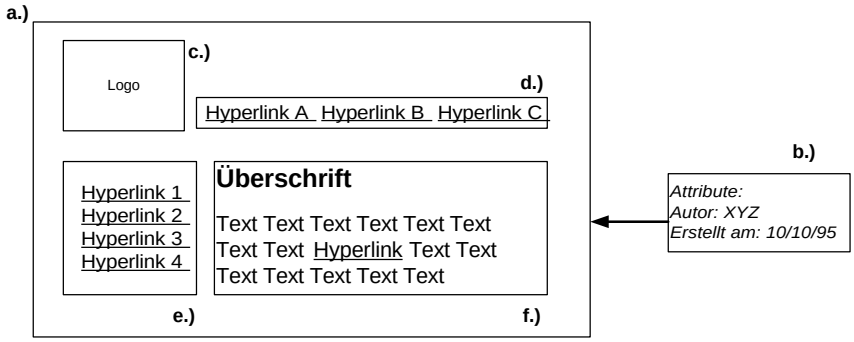

Abbildung 5.4: Ein Beispiel für die Inhaltstypen einer dargestellten HTML-Seite (a), wobei sowohl Informationselemente (c,f), als auch Navigationselemente (d,e) verwendet werden. Obwohl der weitere Inhaltstyp „Attribut“ (b) nicht mit dargestellt wird, ist er dieser Seite inhaltlich dennoch zuzurechnen.

Attribute
Attribute stellen ebenfalls Inhalte dar, die aber im Gegensatz zu den „Informationselementen“ eher als eine Art von „Meta-Informationen“ dienen. Es handelt sich dabei nicht um die eigentlichen Informationen, die den Wert einer Website ausmachen, sondern um übergeordnete Inhalte, die die Informationen genauer charakterisieren.
Dies kann beispielsweise die Angabe eines Autors oder der Zeitpunkt der Erstellung sein. Sie müssen außerdem nicht unbedingt einen direkten inhaltlichen Bezug haben, sondern können sich beispielsweise auch auf den Entwicklungsprozess beziehen, indem z.B. der Status einer Page („fertig“, „überarbeitungswürdig“, ...) vermerkt wird.

An diesen Beispielen soll zweierlei deutlich werden: Zum einen müssen Attribute nicht unbedingt für den Benutzer einer Website sinnvoll erscheinen und werden da-

her nicht zwangsläufig in den HTML-Seiten dargestellt. Zum anderen sind die Grenzen zwischen den „Informationselementen“ und den „Attributen“ fließend und damit eher subjektiv.

5.4. Darstellung

Wie oben beschrieben wurde, wird mittels der Pages jeweils nur der inhaltliche Aspekt einer HTML-Seite definiert und alle Elemente der Darstellung werden außen vor gelassen. Andererseits kann erst im Zusammenspiel mit darstellenden Elementen aus dem abstrakten Konstrukt einer Page eine vollwertige HTML-Seite konstruiert werden.

Elemente der Darstellung wurden schon implizit in Kapitel 2.8 identifiziert und sollen hier im Wesentlichen in „Layout“, „Farben“ und „Typographie“ unterschieden werden. Derartige Mittel können die Aussage der Inhalte unterstützen oder andererseits nur rein dekorativer Natur sein. Damit ist die Darstellung weder nur schmückendes Beiwerk noch ausschließlich zentrales Element einer Website (vgl. [Fleming 97]).

Das Element „Bild“ kann zwar prinzipiell ebenfalls zur „Darstellung“ gezählt werden, wird hier aber aufgrund seiner „Zwitterstellung“ zwischen Inhalt und Darstellung zur eigenständigen Einheit „Object“ (s.o.) gerechnet. Dabei wird vorrausgesetzt, dass ein derartiges Bild einen Inhalt repräsentiert und somit eine Bedeutung hat. Demgegenüber soll hier eine andere Sorte von Bildern, die „Dekorationen“ (vgl. [Fleming 97]), definiert werden, die der Darstellung zuzurechnen sind. Ein solches „dekoratives Element“ wird zwar in HTML ebenfalls durch das Konstrukt eines Bildes repräsentiert, hat aber demgegenüber keine inhaltliche Bedeutung. Als Beispiel hierfür soll folgende Skizze einer HTML-Seite dienen:

Abbildung 5.5: Skizze einer HTML-Seite. Neben den Textinhalten ist ein Bildinhalt (Portrait) und ein Dekorationsbild (Efeu) abgebildet.

Während in obiger Abbildung das Bild der „Person“ eine inhaltliche Bedeutung hat und als Information über dessen Aussehen dienen kann, so ist dies beim Bild des „Efeus“ nicht der Fall. Dieses Bild lockert die Seite zwar auf und verleiht ihm eine besondere Ästhetik, wobei aber unterstellt werden kann, dass es keine explizite inhaltliche Bedeutung hat. Es kann somit durch eine andere Dekoration ersetzt oder aber komplett beseitigt werden, ohne dass sich die Bedeutung der Seite ändert.
Natürlich sind die Grenzen zwischen Dekoration und inhaltsbeladenen Bildern fließend, so dass an dieser Stelle keine klare und allgemeingültige Trennung dieser beider Arten definiert werden kann. Als Anhaltspunkt kann aber vielleicht die oben er-

wähnte Möglichkeit der Ersetzung dienen: Sofern sich ein Bild durch ein anderes ohne inhaltliche Auswirkungen ersetzen lässt, handelt es sich um eine Dekoration.

Eine Darstellung bezieht sich immer auf die einzelnen inhaltlichen Elemente, d.h. es muss explizit für jeden Chunk definiert werden, in welcher Art und Weise die Darstellungselemente darauf angewandt werden sollen. Beispielsweise kann die Darstellung eines „Text“-Chunks über dessen Position, sowie den typographischen Elementen und Farben, festgelegt werden. Dabei muss insbesondere auch auf das Zusammenspiel aller Inhaltselemente einer Page eingegangen werden und dies über das Layout, d.h. einer gemeinsamen Anordnung auf einer Fläche, ausgedrückt werden. Die Gestaltung der Zwischenräume zwischen den angeordneten Elementen sind dabei ebenso zu berücksichtigen, wie die Elemente selber. Mithilfe dieser Definition kann der Inhalt in eine HTML-Seite transformiert werden, wie es anhand von folgendem Beispiel demonstriert werden soll:

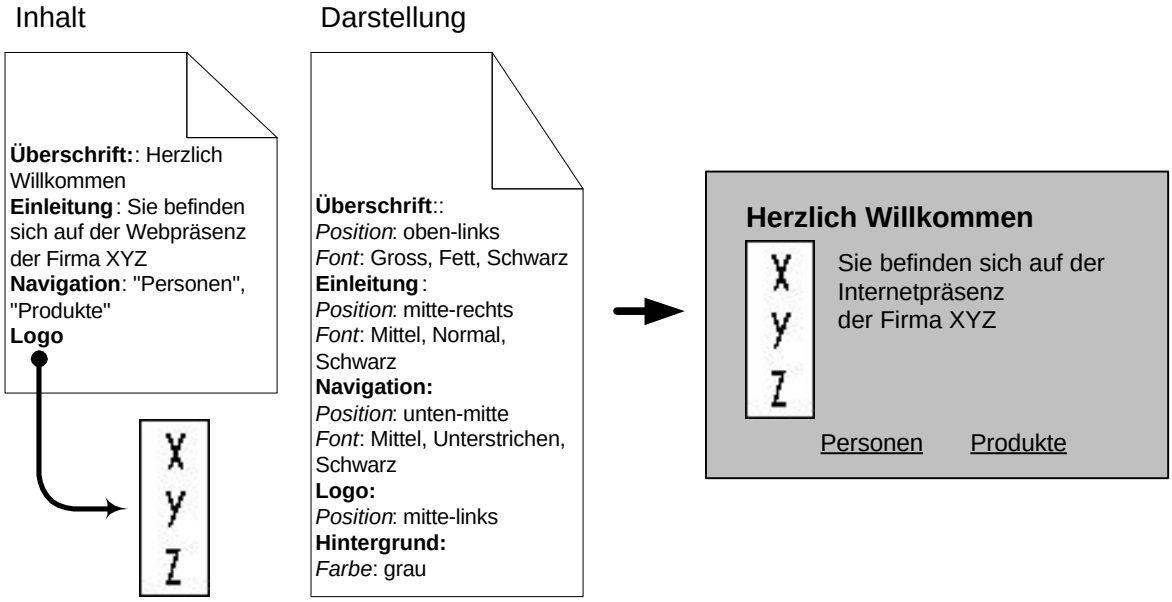

Abbildung 5.6: Beispiel für das Zusammenspiel von Inhalt und Darstellung. Für die vier Chunks des Inhalts (Drei Texte und ein eingebettetes Bild) wird eine Darstellung definiert. Das Ergebnis des Zusammenspiels ist eine HTML-Seite (rechts).

6. Analyse

In diesem Kapitel soll die „Analyse“ beschrieben werden, die als erster Schritt in Bezug auf den Entwurf einer Website zu verstehen ist. Sie dient dazu, genügend Informationen über den Entwurfsgegenstand zu sammeln, um so ein ausreichendes Verständnis über zu berücksichtigende Vorgaben zu gewinnen. Vorgaben werden seitens der Auftraggeber hervortreten, z.B. in Form einer Zielsetzung, aber auch über die Benutzer, deren Bedürfnisse untersucht werden müssen. Ohne derartige Informationen ist es unmöglich, mit dem eigentlichen Entwurf zu beginnen, da so nämlich jeglicher Ansatzpunkt für ein Vorgehen fehlt.

6.1. Ziele der Website

Zur Erinnerung sei folgender Sachverhalt noch einmal wiederholt: Bei einer Website handelt es sich um ein Kommunikationsmedium, über das Informationen an eine Benutzerschaft übermittelt werden sollen. Eine Kommunikation ist immer zweckgebunden, d.h. der Absender der Information möchte den Adressaten in irgendeiner Form beeinflussen (vgl. Kapitel 2.3).
Somit kann davon ausgegangen werden, dass der Auftraggeber in seiner Rolle als Absender der Information zumindest implizit Ziele darüber formuliert hat, was er erreichen möchte. Beispielsweise könnte es sein Wunsch sein, bestimmte Produkte auf der Website zu bewerben, um damit einen erhöhten Verkauf zu erreichen. Ein anderes Ziel könnte die Vermittlung eines bestimmten „Images“ sein: Auf der Website soll der Auftraggeber auf eine bestimmte Art und Weise dargestellt werden, um dessen Ansehen beim Benutzer positiv zu beeinflussen.

Dem Entwickler, der vom Auftraggeber für die Umsetzung dieser Ziele engagiert wurde, sind diese Hintergründe zu Beginn des Prozesses in der Regel noch verborgen. Insofern muss ein erster Schritt sein, diese Ziele in enger Zusammenarbeit mit dem Auftraggeber herauszuarbeiten. Fleming weist darauf hin, dass die vom Auftraggeber verfolgten Ziele oftmals nur sehr diffus dargelegt werden und somit einer genaueren Untersuchung bedürfen [Fleming 98, S. 85f]. Oftmals werden vom Auftraggeber Gründe für den Auftrag genannt, die zwar implizit ein Ziel beinhalten, dieses aber nicht offensichtlich erkennen lassen. Ein geradezu klassisches Beispiel stellt die Aussage „Wir benötigen eine Internetpräsenz“ dar. Damit ist in der Regel nicht etwa eine Website als Selbstzweck gemeint, sondern dahinter steckt möglicherweise die Furcht, in Bezug auf schon im WWW vertretende Konkurrenten, ins Hintertreffen zu geraten.

Gründe für solche unklaren Formulierungen bzw. verschleierte Ziele sieht Fleming beispielsweise darin, dass bestimmte Aussagen aus firmenpolitischen Gründen nicht gestattet sind, oder aber, dass die Unklarheit dieser Aussagen selber nicht erkannt wird [Fleming 98, S. 78f].

Um derartige Unklarheiten zu beseitigen, ist ein direkter und intensiver Kontakt zum Auftraggeber notwendig. Dieser Kontakt kann z.B. in Form von „Brainstorming“-Sitzungen (vgl. [Rosenfeld & Morville 98, S. 136ff]), direkter Konversation (vgl. [Fleming 98, S. 78ff]) oder Ähnlichem stattfinden. Dabei ist es wichtig, dass innerhalb dieser Treffen konkrete Fragen gestellt werden: Diese können zum einen allgemeiner Natur sein, z.B. bezüglich der Firmenphilosophie, aber sollten sich vor allem auf die konkrete Website beziehen. In Anlehnung an [Shiple 98] und [Rosenfeld & Morville 98, S. 138] soll hier exemplarisch folgender Fragenkatalog aufgeführt werden:

- Wie sieht das Selbstverständnis der Organisation aus?
- Welche kurz- bzw. langfristigen Ziele sollen mit der Website verfolgt werden?
- Welche Zielgruppen sollen angesprochen werden?
- Warum sollten Benutzer die Website besuchen kommen?
- Woran soll der Erfolg der Website gemessen werden?

Die Antworten müssen gefiltert und nach Mustern untersucht werden. Dabei sollte insbesondere darauf geachtet werden, dass auch indirekt genannte Ziele erkannt und berücksichtigt werden [Fleming 98, S. 86]. Auf diese Weise lassen sich die wahren Ziele seitens der Auftraggeber herausarbeiten und können so als Liste in einem Entwicklungsdokument zusammengefasst werden. Um die Relevanz einzelner Ziele einschätzen zu können, wird von [Rosenfeld & Morville 98, S. 138f] vorgeschlagen, in einem zweiten Durchlauf die einzelnen Punkte dieser Liste vom Auftraggeber jeweils nach Wichtigkeit bewerten zu lassen. Die Ergebnisse lassen sich als Mittelwert zusammenfassen und vermitteln einen Eindruck über die relevanten und weniger relevanten Ziele.

In der angeführten Literatur wird wiederholt auf die Wichtigkeit dieser Tätigkeiten hingewiesen, weil damit quasi das Fundament für die komplette Website gelegt wird. Die Ergebnisse dokumentieren die Idee der Website und stellen somit die Basis dar, die den kompletten Entwicklungsprozess antreibt.
Insofern sollte insbesondere auf einen Konsens zwischen allen Beteiligten beim Auftraggeber geachtet werden, damit nicht später unberücksichtigte Ziele zu einem Scheitern des Projekts führen (vgl. [Shiple 98]).

6.2. Zielgruppendefinition

Die im vorangegangenen Kapitel beschriebene Identifikation der Ziele schließt zumindest implizit eine bestimmte Zielgruppe mit ein. Unter einer Zielgruppe sollen alle potentiellen Benutzer der Website verstanden sein, für die die Website in Bezug auf die Ziele erstellt wird. Anders ausgedrückt stellt die Zielgruppe den Empfänger des oben geschilderten Kommunikationsprozesses dar. Bei der Zielgruppe wird es sich somit logischerweise nicht um alle Benutzer des WWW handeln, sondern nur um eine kleine Teilmenge aus dieser riesigen Gruppe.

Es wurde in Kapitel 3 dargelegt, dass der Erfolg einer Website von der Akzeptanz seitens des Zielpublikums abhängt: Sofern die Website von den Adressaten ignoriert wird, muss von einem Misserfolg gesprochen werden. Um dies zu verhindern ist es notwendig, dass auf die Bedürfnisse dieser Benutzer eingegangen wird und die Website in Hinblick auf deren Erfüllbarkeit erstellt wird. Um diese konkreten Bedürfnisse überhaupt bestimmen zu können, ist aber ein ausreichendes Verständnis darüber unablässig, wer diese Benutzer überhaupt sind.

Die über den Begriff der "Benutzer" bezeichnete Menge von Personen ist laut [Sisson 99] keine homogene Gruppe, sondern besteht aus Individuen, die jeweils ihre ganz eigenen Ziele, Motivationen und Aufgaben verfolgen. Da aber die Analyse einzelner Individuen gemäß der hier diskutierten Zielsetzung faktisch unmöglich ist, muss ein anderer Weg beschritten werden: Dabei können jeweils mehrere Individuen gemäß verschiedener Kriterien zu handhabbaren Gruppen zusammengefasst werden, die dann jeweils einzeln analysiert werden können. In [Sisson 99] wird als ein Gruppierungskriterium z.B. die "Rolle" der Benutzer genannt, also der Zweck, zu dem die

Benutzer die Website besuchen werden. Dies kann z.B. die Suche nach bestimmten Informationen sein, oder auch nur das ungezielte Durchstreifen der Website. Als anderes Kriterium wird auch "Wissen und Erfahrung" genannt, d.h. inwieweit die Benutzer überhaupt mit dem WWW vertraut sind.

Klassifikationen sind durchaus nicht unproblematisch, denn je nach Wahl des Klassifikationsschemas wird man eine bestimmte Sichtweise auf die abstrakte Benutzerschaft bekommen. Diese Sicht hat mit Sicherheit auch Auswirkungen auf das weitere Vorgehen, weshalb ein Klassifikationsschema sorgfältig und in Bezug auf die gegebenen Problemstellung zu wählen ist [Sisson 99].

6.3. Bedarfsanalyse

Auf Basis der im vorrangegangenen Abschnitt beschriebenen Gruppierung von Individuen zu handhabbaren Gruppen kann eine Bedarfsanalyse stattfinden. Ziel dabei ist es, Informationen über die Bedürfnisse der Benutzer jeder einzelnen Gruppe zu erlangen, um entsprechend dieser Angaben eine adäquate Gestaltung der Website vorzunehmen. Die Zielsetzung einer solche Analyse wird von [Lindgaard 94, S. 44] folgendermaßen charakterisiert: "The user-needs analysis is thus an analysis of what is currently done by users, a projection of what they will be doing, and a comparison of present and future tasks as well as a description of their equipment and the environment into which the new system will fit".

Diese Diplomarbeit behandelt den Entwurf eines Informationssystems, weshalb in diesem Zusammenhang im Wesentlichen die Frage nach den zu integrierenden Inhalten von Interesse ist. Man kann davon ausgehen, dass Benutzer, die auf ein Informationssystem zugreifen, auf der Suche nach einer bestimmten Information sind. Die Suche nach dieser Information findet zudem unter bestimmten Umständen statt, wie z.B. im Rahmen einer "langandauernden Recherche" oder aber "unter Zeitdruck über ein langsames Modem". Entsprechend solcher Umstände kann zum Beispiel die Strukturierung der Informationen unterschiedlich ausfallen, so dass wichtige Informationen über weniger Hyperlinks zu erreichen sind, als unwichtigere.
Insofern dreht sich diese Analyse in erster Linie zwar um die Frage "Welche Inhalte werden benötigt?", wobei aber die begleitenden Umstände nicht außer Acht gelassen werden dürfen.

Es ist eine falsche Annahme, man könne Benutzer direkt nach ihren Bedürfnissen in Bezug auf ein System befragen und würde dadurch befriedigende Antworten bekommen. Potentielle Benutzer haben oftmals keine Vorstellung davon, was ihnen ein bis dahin nur hypothetisches System bieten kann und was sie davon erwarten können. Insofern werden sie auf eine entsprechende konkrete Frage nicht mitteilen können, was sie z.B. für Wünsche haben. Überspitzt wird dies von [Rahardja 99] so formuliert: “to ask a question, one must know enough to know what is not known”. Als Konsequenz sollte versucht werden, allgemeinere Informationen über die Benutzer zu sammeln, mit deren Hilfe dann Rückschlüsse auf den Entwurfsgegenstand erlangt werden können.

6.3.1. Fragestellungen

In Anlehnung an [Fuccella & Pizzolato 98] sollen hier drei Kategorien von Fragen angeführt werden, die als Anhaltspunkte für eine Analyse gemäß obiger Ausführungen gelten können. Damit ist gemeint, dass auf Basis dieser Hinweise ein konkreter Fragekatalog zusammengestellt werden kann. Die Auswahl der Fragen hängt z.B.

von der Zielsetzung der Website ab und muss dementsprechend berücksichtigt werden.

Allgemeine Fragen
In Fragen, die unter diese Kategorie fallen, werden persönliche Angaben verlangt, die nichts mit der konkreten Materie des WWW zu tun haben. Darunter fallen beispielsweise "Beruf", "Einkommen" oder "Bildungsweg". Die Antworten auf diese Fragen liefern zwar keine direkten Informationen über bestimmte Bedürfnisse, lassen aber dennoch in diese Richtung Rückschlüsse zu. Beispielsweise könnte sich durch die Frage nach dem Bildungsweg herausstellen, dass ein Großteil der Benutzer Studenten sind, was vielleicht die Verwendung bestimmter sprachlicher Mittel in den Inhalten impliziert.

In einem in [Lindgaard 94, S. 46] vorgestellten Fragenkatalog werden selbst Fragen in Bezug auf physiologische Merkmale, wie z.B. eventueller körperlicher Behinderungen aufgeführt. Solche Fragen machen dann Sinn, wenn z.B. zu erwarten ist, dass ein Großteil der Benutzer eine Sehschwäche hat. Daraus könnte seinerseits resultieren, dass die visuelle Gestaltung der Website entsprechend an diesen Umstand adaptiert wird.
Anhand der Bandbreite solcher Fragen wird deutlich, dass deren Auswahl sorgfältig auf die konkreten Umstände angepasst werden muss, damit alle relevanten Aspekte behandelt werden.

Fragen zur Benutzung des WWW
Unter diese Kategorie von Fragen fallen alle solchen, die Hinweise darüber geben, wie und unter welchen Umständen das WWW benutzt wird. Dabei erscheint zunächst eine Untergruppe von Fragen interessant, die Hinweise auf die verwendete Hard- und Software liefert. In [Sisson 99] wird dargelegt, dass die Verwendung bestimmter Betriebssysteme oder Browser einen nicht zu unterschätzenden Einfluss auf die Darstellung der Website auf dem Zielsystem hat. Beispielsweise werden in Abhängigkeit vom verwendeten Browser bestimmte Techniken der Darstellung (z.B. "Cascading Style Sheets") unterschiedlich unterstützt. Andere interessante Einflussgrößen stellen die Zugangsgeschwindigkeit ins Internet oder auch die Farbkapazität der Grafikkarte und die Größe des Monitors dar.
Sofern sich herausstellen sollte, dass bei den Benutzern ein bestimmtes System überwiegt, könnte als Konsequenz eine Ausrichtung oder Optimierung der Website in dieser Hinsicht stattfinden.

Weiterhin könnte von Interesse sein, aus welchen Gründen das WWW besucht wird und ob es beispielsweise als "Unterhaltungsmedium" oder aber zur "Unterstützung der eigenen Arbeit" verwendet wird. In diesem Zusammenhang können die Zeiträume der Verwendung (z.B. "mehr als drei Stunden täglich") oder auch die Zeitpunkte (z.B. "Immer nur nachts") wichtige Ergebnisse der Analyse liefern.

Fragen zur Benutzung einer speziellen Website
Im Gegensatz zu den beiden vorherigen Kategorien von Fragen, in denen eher allgemeine Dinge geklärt werden, beziehen sich Fragen dieser Kategorie auf eine konkrete Website. Fragen in dieser Hinsicht haben das Ziel, konkrete Probleme und Bedürfnisse herauszufinden, woraus sich oftmals direkt Implikationen in Bezug auf den Entwurf ergeben. Insbesondere sind in diesem Zusammenhang konkrete Bedürfnisse in Form nachgefragter Inhalte von Bedeutung (s.o.). Außerdem lassen sich auch

subjektive Vorlieben oder Abneigungen gegenüber bestimmten Entwurfsentscheidungen herausfinden, wie z.B. die Verwendung bestimmter Farbschemata.

In [Sullivan 96a] wird ein Leitfaden für derartige Fragen aufgeführt, in dem die relevanten Aspekte einer Website behandelt werden:

- *Clarity of Communication*: z.B. "Ist die verwendete Sprache verständlich und natürlich?"
- *Accessibility*: z.B. "Ist die Ladezeit unter gegebenen technischen Umständen akzeptabel?"
- *Consistency*: z.B. "Hat die Website ein einheitliches Look&Feel?"
- *Navigation*: z.B. "Ist es offensichtlich, wohin einzelne Hyperlinks führen?"
- *Design & Maintenance*: z.B. "Gibt es tote Hyperlinks?"
- *Visual Presentation*: z.B. "Sind die verwendeten Farben vernünftig?"

In Bezug auf die zentrale Frage nach den Inhalten werden von [Fuccella, Pizzolato & Franks 99] u.a. folgende Fragen formuliert:

- "Aus welchen Gründen wird die Website besucht?"
- "Welche Inhalte sind oder wären nützlich?"
- "Warum wird ein bestimmter Inhalt benötigt? Welche Probleme werden damit gelöst? Welche Vorteile verschafft er?"

6.3.2. Durchführung und Probleme

Wie in Kapitel 3 schon ausführlich dargelegt wurde, kann eine Bedarfsanalyse im Wesentlichen nur durch direktes oder indirektes Einbeziehen von Benutzern durchgeführt werden, weil man ansonsten auf Mutmaßungen beschränkt wäre. Allerdings besteht zu diesem Zeitpunkt der Entwicklung die Schwierigkeit, dass es noch keine direkten Benutzer gibt, die in den Prozess einbezogen werden könnten. Stattdessen kann man sich zunächst mit Repräsentanten begnügen, d.h. mit Personen, die grundsätzlich einem ähnlichen Umfeld zugehörig sind. Hinweise zur Rekrutierung derartiger Repräsentanten finden sich in Kapitel 3.4.1.

Ähnlich problematisch ist zu diesem Zeitpunkt eine Bezugnahme auf eine konkrete Website, da diese in der Regel noch nicht existiert. Um dieses Manko zu umgehen, kann auch hier mit einem Stellvertreter gearbeitet werden, d.h. es wird Bezug auf eine oder mehrere fremde Websites genommen. Dabei kann es sich z.B. um solche handeln, die grundsätzlich eine ähnliche Thematik beinhalten. Zwar werden die Ergebnisse einer solchen Analyse nur indirekt verwertbar sein, aber unter Umständen lassen sie zumindest Rückschlüsse auf einen realen Entwicklungsgegenstand zu.
Anders sieht es für den Fall aus, dass eine schon existierende Website überarbeitet werden soll. In diesem Fall kann sowohl auf "echte" Benutzer zurückgegriffen werden als auch in Bezug auf eine reale Website nach konkreten Bedürfnissen und Problemen gefragt werden.

In Hinblick auf eine konkrete Problemstellung müssen die oben beispielhaft angeführten Fragen variiert und angepasst werden, weshalb diese Ausführungen nur als Anhaltspunkt zu verstehen sind. Dazu sollte vor der eigentlichen Analyse eine Vorbereitung stattfinden, in der Klarheit über bestimmte Fragestellungen und konkrete Fragen gewonnen wird. Auf diese Weise ist sichergestellt, dass die Ergebnisse in einer halbwegs strukturierten Form vorliegen, was u.a. der Auswertung zugute kommen wird. Zusätzlich bietet es sich u.U. an, die Fragen im Vorfeld zu

wird. Zusätzlich bietet es sich u.U. an, die Fragen im Vorfeld zu evaluieren, um zum Beispiel missverständliche Fragen zu vermeiden (vgl. Kapitel 3.4.2).

Bei der eigentlichen Durchführung der Analyse können die Fragen mittels der in Kapitel 3.4 aufgeführten Mittel, wie z.B. Interviews, Umfragen und Diskussion, beantwortet werden. Dabei sollte darauf geachtet werden, dass diese Mittel und die Fragen so geschickt kombiniert werden, dass nur wenige Durchläufe nötig sind. Das bedeutet z.B., dass ein Sitereview und eine Diskussion integriert stattfinden, so dass die beteiligten Benutzer nicht mehrfach bemüht werden müssen. Zum einen können die Benutzer nämlich nicht über die Maßen in Anspruch genommen werden, zum anderen stellt jeder Durchlauf inklusive Auswertung einen Zeitfaktor dar.

Eine Auswertung sollte in Bezug auf die jeweilige Frage stattfinden, wobei in den u.U. unstrukturierten Ergebnissen nach Mustern gesucht werden kann. In [Fuccella, Pizzolato & Franks 99] wird für wichtige Analyseergebnisse, insbesondere in Bezug auf erwünschte Inhalte, ein zweiter (Analyse-)Durchlauf, ähnlich wie in Kapitel 6.1, beschrieben: Dazu werden alle genannten Antworten aufgelistet und den einzelnen Beteiligten erneut vorgelegt. Durch Vergabe von "Noten" soll dann für jeden einzelnen Eintrag die Relevanz in Bezug auf die Website bewertet werden. Nach deren Auswertung lassen sich auf diese Weise genauere Aussagen darüber treffen, wo ein Konsens seitens der Benutzer vorherrscht, d.h. welche Aussagen eher wichtig und welche unwichtig sind.

6.3.3. Entwicklungsdokumente

Die Ergebnisse der Analyse müssen auf geeignete Art und Weise dokumentiert werden, so dass die jeweiligen Implikationen für das weitere Vorgehen erkannt werden können.
Wie oben schon erwähnt, sollen die erwünschten Inhalte als zentrales Ergebnis der Bedarfsanalyse betrachtet werden, weshalb ein entsprechendes Dokument an dieser Stelle hervorgehoben werden soll. In [Bailey 97] wird dazu ein sogenanntes "Content Inventory" vorgestellt, in welchem alle diese Inhalte tabellenartig auflistet werden. Als Tabellenfelder sind "Titel" und die "Beschreibung" sinnvoll: Mit ersterem wird der jeweilige Inhalt identifiziert, während der zweite als dessen Inhaltsangabe dient. Grundsätzlich kann diese Tabelle noch beliebig viele andere Felder beinhalten, in denen weitere Informationen zum entsprechenden Inhalt untergebracht werden. Dabei könnte z.B. wichtig sein, an welche Zielgruppe sich ein bestimmter Inhalt richtet oder auch welchen Zweck dieser Inhalt erfüllt.

Titel	**Beschreibung**
Produkte	Auflistung aller im Sortiment befindlichen Produkte inklusive detaillierter Beschreibung und Bestellmöglichkeit
Personen	Liste aller beschäftigten Mitarbeiter inklusive ihrer Position und Telefonnummer
…	*…*

Abbildung 6.1: Beispiel eines Content-Inventories. Jeder Eintrag wird zweispaltig mittels „Titel“ und „Beschreibung“ angelegt.

Für alle anderen Angaben, die hier nur unspezifisch betrachtet werden können, sollte versucht werden, jeweils thematisch Zusammengehörige zu jeweils einem Dokument zusammenzufassen. Als Orientierungspunkt für eine derartige Kategorisierung können grundsätzlich die Prozessschritte des Entwurfs entsprechend Kapitel 4.8 dienen, nämlich „Inhalte“, „Struktur“, „Navigation“, „Labels“ und „Darstellung“. Dies hätte den Vorteil, dass jedem Schritt die jeweiligen Vorgaben unmittelbar zur Verfügung stehen würden.

7. Entwurf

Mittels der Analysephase sollte ein erstes Verständnis über den zu entwickelnden Gegenstand gewonnen worden sein. Seitens der Auftraggeber und der Benutzer sind eine Reihe von Vorgaben gemacht worden, die es in dieser Phase in einen Entwurf zu überführen gilt. Das bedeutet z.B., dass für gegebene Probleme Lösungen gefunden werden müssen, die sich in Form von Entwurfsentscheidungen in der Website widerspiegeln.

7.1. Konzeption

Um den Entwurf einer Website mittels einzelner separater Schritte tätigen zu können, sollte eine übergeordnete Sicht auf das "große Ganze" vorhanden sein. Sollte dies nicht der Fall sein und jede der Komponenten würde komplett separat voneinander entwickelt werden, würden die Teilergebnisse nur schwerlich zu einem einheitlichen Ganzen zusammenzubringen sein. Insofern ist es notwendig, dass zwischen allen Beteiligten eine gemeinsame Sicht auf den Entwurfsgegenstand entsteht, die dann als Vorgabe für die Einzelentwürfe dient. Von besonderer Wichtigkeit ist in diesem Zusammenhang die Festlegung der Inhalte, die in den einzelnen Schritten als zentraler Bezugspunkt dienen.

Aus der Analysephase geht eine Reihe von Ergebnissen hervor, in denen mehr oder minder abstrakt Probleme bzw. Bedürfnisse beschrieben sind. Für diese Vorgaben gilt es, im Rahmen des Entwurfs Lösungen zu entwickeln, für die aber nicht unbedingt das "Rad neu erfunden" werden muss. Stattdessen kann und sollte versucht werden, Inspiration bei Lösungen ähnlicher Probleme zu suchen.
In diesem Zusammenhang wird von verschiedenen Autoren die Inspektion diverser Medien, wie z.B. CD-ROMs [Fleming 98, S. 86f] vorgeschlagen. Eine große Bedeutung hat dabei offenbar die Betrachtung von konkurrierenden Websites ("Competitive Analysis"), wie es beispielsweise in [Nielsen 98b], [Shiple 98] oder [Fuccella & Pizzolato 98] beschrieben ist.
In [Shiple 98] wird zudem vorgeschlagen, dass eine derartige Inspektion auf systematische und strukturierte Art und Weise durchgeführt werden sollte. Dabei werden alle in Frage kommenden Medien gemäß eines Leitfadens durchforstet und die bemerkenswerten Aspekte vermerkt. Ein derartiger Leitfaden ist durch die Struktur der Analyseergebnisse schon vorgegeben und wird sich z.B. an verschiedenen Oberthemen wie z.B. "Navigation" und "Darstellung" orientieren.
Über die Technik des Brainstorming können weiterhin gemeinsame zusätzliche Lösungsansätze gefunden und entwickelt werden [Rosenfeld & Morville 98, S. 148].

Die Ergebnisse der Inspektion sollten nicht mit den noch ausstehenden Entwurfsentscheidungen verwechselt werden, sondern beschreiben Lösungsideen, deren Einbeziehung noch nicht sichergestellt ist. Dementsprechend wird die Dokumentation derartiger Ergebnisse in [Rosenfeld & Morville 98, S. 142ff] als "Wish List" bezeichnet.

Auf Basis der Probleme und ihrer möglichen Lösungen muss eine Diskussion entstehen, in der diese unter verschiedenen Aspekten bewertet werden. Ziel ist es dabei, eine sog. "Vision" zu entwickeln, die in Anlehnung an [Kreitzberg 98] als eine "[...] klare, geteilte und kommunizierbare Sicht auf das Produkt" verstanden sein soll. Bei dieser Diskussion ist sicherzustellen, dass alle relevanten Personen mit einbezogen werden, damit alle von denselben Vorraussetzungen ausgehen können. Aspekte dieser Bewertung können beispielsweise die Machbarkeit in Bezug auf technische

und zeitliche Mittel sein, oder auch die Konformität mit den angestrebten Zielen. Neben der Einzelbetrachtung der Lösungen müssen diese auch untereinander in Bezug gesetzt werden, damit deren problemloses Zusammenspiel sichergestellt ist.

Die wesentlichen Ergebnisse der Konzeption mittels einen kurzen Prosatextes dokumentiert werden können (vgl. [Züllighoven et al. 98, S. 618]). Speziellere Ergebnisse, wie z.B. die Auswahl von Inhalten, sollte hingegen gesondert beschrieben werden, wie z.B. mit dem oben genannten "Content-Inventory".

7.2. Inhalte

In diesem Abschnitt „Inhalte“ soll ein Vorgehen beschrieben werden, bei dem jeder einzelne Inhalt der Website spezifiziert wird. Als Vorgabe für ein derartiges Vorgehen dient das "Content-Inventory", in dem alle zu verwendenden Inhalte zwar aufgelistet, aber nur sehr unzureichend spezifiziert sind: Bis auf einen Titel und einer groben Beschreibung ist über den jeweiligen Inhalt nichts weiter bekannt. Zunächst soll jeder Eintrag des Content-Inventory als inhaltliche Repräsentation genau einer Page gesehen werden, d.h. ein derartiger Inhalt beschreibt letztendlich das, was auf einer später implementierten HTML-Seite zu sehen ist.

Die Spezifikation der Inhalte soll in einer Weise stattfinden, die dem Modell aus Kapitel 5 entspricht. Das bedeutet beispielsweise, dass Inhalte als eine hierarchisch strukturierte Sammlung von Chunks verschiedener Datentypen konstruiert werden. In Bezug auf die Semantik der Inhalte sollen in diesem Schritt allerdings explizit die „Navigationselemente“ gemäß Kapitel 5.3.2 ausgenommen werden, da diesen ein eigener Entwurfsschritt in Kapitel 7.4 gewidmet ist.
Als Ergebnis der Inhaltsdefinition soll für jeden Eintrag des Content-Inventories einerseits dessen exakter Inhalt festgelegt und andererseits auch dessen Struktur definiert sein.

7.2.1. Vorgehen

Je nach Anwendungsfall wird ein Vorgehen beim Inhaltsentwurf eher "Top-Down" oder eher "Bottom-Up" sein. Dies ist abhängig davon, ob ein Inhalt gänzlich neu entworfen werden muss, oder aber vollständig oder zu einem Teil schon vorliegt. Letzteres ist z.B. dann der Fall, wenn Inhalte aus organisatorischen Gründen aus anderen Bereichen wiederverwendet werden sollen. Grundsätzlich ist ein derartiger Sachverhalt in [Rosenfeld 98] beschrieben, wobei darauf hingewiesen wird, dass die Richtung des "Top-Down" im Gegensatz zu einem "Bottom-Up" eher einem benutzerorientierten Vorgehen entspricht. Beim "Top-Down"-Entwurf wird ausgehend von einem sehr abstrakten Inhalt (der z.B. gemäß eines Bedürfnisses definiert wurde) die Struktur Schritt für Schritt verfeinert und mündet letztendlich in einem konkreten Inhalt. Beim "Bottom-Up"-Entwurf wird von diesem konkreten Inhalt ausgegangen, dessen Struktur Schritt für Schritt vergröbert wird. Grundsätzlich soll hier von einem Top-Down-Verfahren ausgegangen werden, wobei aber Bottom-Up-Einflüsse nicht grundsätzlich ausgeschlossen werden sollen.

Ausgehend von der abstrakten Beschreibung des Inhalts werden die damit verbundenen Chunks festgelegt und dann iterativ verfeinert. Zur Veranschaulichung soll folgendes Beispiel dienen: Angenommen der Autor dieser Diplomarbeit soll in einem "Steckbrief" vorgestellt werden. Ein erster Schritt könnte beispielsweise die Chunks "Name", "Bild" und "Selbstdarstellung" hervorbringen. In einem zweiten Schritt wird

die "Selbstdarstellung" in "bisherige Projekte" und "zukünftige Projekte" untergliedert usw.
In jedem Schritt wird also von einem abstrakten Konstrukt ausgegangen und dieses in ein konkreteres überführt. Dies muss im Prinzip solange durchgeführt werden, bis der Inhalt ausreichend detailliert vorliegt.

Die Untergliederung des Inhalts in mehrere Chunks ist natürlich in hohem Maße abhängig davon, was mit dem jeweiligen Inhalt für ein Zweck verfolgt werden soll bzw. welcher Natur der Inhalt ist. Entsprechend dieses Zwecks wird sich die Gestaltung möglicherweise von selber ergeben. Dazu kann auf die während der Bedarfsanalyse erlangten Zusatzinformationen zurückgegriffen werden, in denen u.U. vermerkt wurde, welchen Nutzen eine bestimmte Information den Benutzern bringt. Als partizipative Maßnahme können Benutzer auch direkt befragt werden, was ihrer Meinung nach die einzelnen Einträge für detailliertere Inhalte beinhalten sollen. Implizit ergeben sich derartige Aussagen aus dem in Kapitel 7.5.1 beschriebenen Evaluationsverfahren der "Category Description".

Jeder einzelne Chunk ist einem der Datentypen "Text", "Object" oder "Hyperlink" (vgl. Kapitel 5.3.1) zugehörig, woran sich bei deren Festlegung auch orientiert werden muss. Je nachdem, welcher Datentyp aktuell vorliegt, werden damit auch zulässige Untergliederungen definiert. So darf beispielsweise ein "Hyperlink" nur in die Einheiten "Label" und "Referenz" unterteilt werden. Demgegenüber muss ein Object, sofern einmal von den Attributen abgesehen wird, als atomar betrachtet werden und kann nicht weiter untergliedert werden.

7.2.2. Richtlinien zur Gestaltung von Texten

Für die Gestaltung von Texten haben sich im Laufe der Zeit bestimmte Leitfäden herauskristallisiert, deren Anwendung sich als vorteilhaft für das spätere Lesen bzw. Erfassen eines Textes herausgestellt hat. Ein Beispiel stellt das Konstrukt "Überschrift" dar: Bevor ein Leser einen Text überhaupt durch- oder angelesen hat, kann er mittels der Überschrift grob einschätzen, ob der Text im aktuellen Kontext für ihn relevant ist.
Es kann davon ausgegangen werden, dass ein Text so lesefreundlich wie möglich gestaltet werden sollte: So werden nämlich die Chancen verbessert, dass der Text zum einen auch wirklich durchgelesen und zum anderen in der intendierten Weise aufgefasst wird. Derartige Richtlinien lassen sich im Bereich des Journalismus finden, in dem das Schreiben von Texten (in Kombination mit Bildern) eine lange Tradition hat. Ohne dies im Detail zu behandeln, sei auf [Schneider & Raue 98] und [La Roche 95] hingewiesen, wo u.a. Hinweise zum strukturellen Aufbau von Texten gegeben werden.
In Bezug auf das Lesen von Texten aus dem WWW herrschen, wie schon in Kapitel 2.8.1 erwähnt, Besonderheiten vor, die es ebenfalls zu berücksichtigen gilt. Da Texte im WWW im Wesentlichen nur überflogen werden, sollten diese so knapp und klar wie möglich formuliert werden. Auch hierauf soll nicht detaillierter eingegangen werden, sondern stattdessen auf die zahlreiche Literatur wie z.B. [Morkes & Nielsen 97], [Outing 99] oder [Dougherty 97] hingewiesen werden.

Ein wichtiger Aspekt von Texten ist die dabei verwendete Sprache, die beispielsweise über bestimmte Worte und einen bestimmten Schreibstil definiert wird. Damit Texte von den Benutzern in der intendierten Weise verstanden werden, ist es natürlich wichtig, "deren" Sprache zu verwenden. Beispielsweise werden sich Texte für Pro-

fessoren gegenüber Texten von Grundschülern deutlich in ihrer Art voneinander unterscheiden. So gesehen ergibt sich eine Notwendigkeit, dass im Rahmen der Inhaltsdefinition auch eine bestimmte Sprache festgelegt wird (vgl. [Rosenfeld 99c]). Diese wird sich notwendigerweise an den Vorgaben durch die Benutzer orientieren müssen.

7.2.3. Entwurf von Objects

Wie schon erwähnt, werden alle Chunks vom Datentyp "Object" aus technischen Gründen als eigenständiger Inhalt separat von ihrer "Page" entworfen und nur mittels Referenz eingebunden. Dies bedeutet für den Entwurfsprozess, dass jedes neu festgelegte Object prinzipiell im Content-Inventory verzeichnet werden muss.
In einem ersten Entwurfsschritt kann es u.U. ausreichend sein, den vom Object dargestellten Inhalt in Textform zu beschreiben und z.B. als Attribut zu dokumentieren.

Da Objects alle Medientypen des WWW repräsentieren, ist es im Rahmen dieser Diplomarbeit ein unmögliches Unterfangen, derartige Entwurfsvorgänge für alle diese Möglichkeiten detailliert zu schildern: Der Entwurf einer Grafik verlangt beispielsweise andere Disziplinen als der Entwurf einer Animation. Dies bezieht sich zum einen auf das nötige Hintergrundwissen bzw. auf die damit verbundenen Erfahrungen und zum anderen auf die Form, wie solche Entwürfe dokumentiert werden: Während es bei Grafiken vielleicht auf Papier gebrachte Skizzen sind, sind es bei einer Animation möglicherweise jeweils eine Abfolge solcher Skizzen („Storyboard“). Auch die Auswahl entsprechender Werkzeuge zur Erstellung der Objects hängt von dessen Typ ab. Insofern soll diese Thematik hier nicht weiter verfolgt werden, sondern ganz allgemein auf die Erkenntnisse der jeweiligen Fachrichtung verwiesen werden, wie z.B. dem "Grafikdesign".

7.2.4. Evaluation

Für die Evaluation einzelner Inhalte sind aus der Disziplin der Informationsarchitektur keine expliziten Verfahren benannt. Insofern sollen hier kurz einige Ideen vorgestellt werden, in denen Richtungen für ein derartiges Vorgehen skizziert werden.
Als zu evaluierender Parameter kann grundsätzlich die "Verständlichkeit" eines einzelnen Inhaltes dienen, d.h. wie schnell er erfasst werden kann und ob es grundsätzliche Verständnisschwierigkeiten gibt. Dazu erscheint ein Szenario in grober Anlehnung an den in [Shneiderman & Kearsley 89, S. 71] beschrieben Versuch sinnvoll: Innerhalb eines bestimmten Zeitraumes muss ein Inhalt von einer Testperson "gelesen" werden, um ihn danach sinngemäß wiederzugeben. Inhaltliche Missdeutungen lassen dabei z.B. den Schluss auf unverständliche Formulierungen zu. Zusätzliche subjektive Bewertungen (z.B. "viel zu trocken geschrieben") stellen eine weitere Möglichkeit der Evaluation dar. Weiterhin kann auch auf Basis der in Kapitel 7.2.2 genannten Richtlinien eine entsprechende Evaluation stattfinden.

7.2.5. Entwicklungsdokumente

Für die Dokumentation der bei der Inhaltsdefinition erhaltenden Ergebnisse bietet es sich an, dies losgelöst vom Content-Inventory in einem gesonderten Dokument zu verzeichnen. So wird gewährleistet, dass einerseits die Übersichtlichkeit des Content-Inventory bewahrt bleibt und andererseits genügend Raum für die unter Umständen umfangreichen Ergebnisse der Inhaltsspezifikation zur Verfügung steht.
Prinzipiell bietet sich eine Dokumentation mittels der in [Bray, Paoli & Sperberg-McQueen 98] spezifizierten Sprache XML (Extensible Markup Language) an, die speziell für derartige Zwecke entwickelt wurde. Aus Gründen der Übersichtlichkeit

sollen diese Möglichkeiten erst in Kapitel 8 detaillierter behandelt und zusätzlich auch eine Lösung in Bezug auf das hier verwendete Modell beschrieben werden.

7.3. Struktur

In diesem Entwurfsschritt geht es darum, die Inhalte der Website zu ordnen und in eine hierarchische Struktur zu überführen, so wie es in Kapitel 2.5 erläutert wurde. Zusätzlich muss in einem Schritt "Chunking" endgültig festgelegt werden, welche Inhalte zu jeweils einer Page zusammengefasst werden. Als Ergebnis sollen auf Basis des Content-Inventories die Struktureinheiten "Page", "Object" und "Container" (vgl. Kapitel 5.2) definiert sein.

Das Strukturieren der Inhalte ist aus zweierlei Gründen von großer Bedeutung: Zum einen wird dadurch Information konstruiert (s. Kapitel 2.5) und zum anderen wird damit auch ein Aspekt der Navigation festgelegt (vgl. Kapitel 2.7). Da es keine "richtige" oder "falsche" Ordnung gibt, sondern nur verschiedene mentale Modelle von einer Ordnung, ist es an dieser Stelle besonders wichtig, partizipativ vorzugehen. Das bedeutet, dass über die Zusammenarbeit mit einzelnen Benutzern deren jeweiliges mentales Modell vom Informationsraum verstanden und berücksichtigt wird.

7.3.1. Vorgehen & Evaluation

Das Vorgehen soll als Abfolge mehrer Schritte beschrieben werden, die iterativ durchlaufen werden. Dadurch entsteht Schritt für Schritt die Struktur.

Card Sorting

Im ersten Schritt geht es darum, die Inhalte des Content-Inventory in eine Ordnung zu bringen, d.h. inhaltlich zusammengehörige Einträge zu gruppieren. Eine gängige Methode, mit der so etwas geleistet werden kann, ist das sog. "Card Sorting", wie es z.B. in [Nielsen & Sano 94] oder in [Fuccella & Pizzolato 98] beschrieben ist. Bei dieser Methode werden die zu ordnenden Begriffe einzeln auf eine Karteikarte geschrieben und dem jeweiligen Teilnehmer in einer zufälligen Reihenfolge vorgelegt. Dessen Aufgabe ist es, diese Karten zu Stapeln zu gruppieren, indem die, seiner Meinung nach, zusammengehörige Inhalte zusammengefasst werden. Nach Beendigung dieser Tätigkeit sollte der Teilnehmer jedem dieser Stapel einen Namen geben bzw. kurz erläutern, nach welchem Schema diese Ordnung stattgefunden hat.
Dieser Vorgang kann separat mit jeder der teilnehmenden Personen anhand eines eigenen Kartenstapels durchgeführt werden, wobei eine zusammenfassende Auswertung gemäß [Fuccella & Pizzolato 98] folgendermaßen aussieht: Ausgehend von den Stapeln des ersten Teilnehmers werden die Ergebnisse mit denen des zweiten verglichen. Alle Stapel, in denen von beiden Teilnehmer ein jeweils ähnliches Ordnungskonzept verfolgt wurde, werden gruppiert, während alle anderen Stapel separat übernommen werden. Dieser Vorgang wird iterativ mit allen Einzelergebnissen durchgeführt, so dass am Ende ein Gesamtergebnis aus mehreren Stapeln vorliegt.
Eine Auswertung dieser Stapel wird ergeben, dass bestimmte Karten untereinander sehr häufig gruppiert werden, während die Zuordnung anderer weniger klar erscheint. Auf diese Weise werden Anhaltspunkte über inhaltlich zusammengehörige Elemente des Content-Inventory erlangt.

Kategoriedefinition

In einem nächsten Schritt müssen eine Reihe von Kategorien festgelegt werden, denen später alle Inhalte zugeordnet werden. Dies sollte in Bezug auf die Ergebnisse des vorherigen Schrittes geschehen, d.h. den dort ermittelten Ordnungskonzepten.

Mit der Definition von Kategorien wird gleichzeitig die erste Hierarchieebene der Struktur festgelegt. Da eine Struktur zugleich auch die Navigation mitdefiniert (vgl. Kapitel 7.4), sollte der Entwickler sich zusätzlich von den Vorgaben aus diesem Bereich leiten lassen. So z.B. von der "7 +/- 2"-Regel (vgl. [Neale 97]), die letztendlich besagt, dass die Anzahl der Kategorien durch ca. 7 Stück begrenzt sein sollte. Jede dieser Kategorien sollte mit einem vorläufigen "Label" versehen werden, mit dem das dort angewandte Ordnungsschema beschrieben wird.

Jede der definierten Kategorien entspricht dem Strukturierungskonstrukt des "Containers" wodurch ein erster Schritt in Bezug auf die Strukturbausteine des Modells aus Kapitel 5 getätigt wurde.

Category Identification
Mittels eines weiteren Schrittes müssen alle Inhalte den Kategorien zugeordnet werden. Dies wird in [Fuccella & Pizzolato 98] als ein partizipativer Akt Namens "Category Identification" beschrieben: Diese Methode verläuft im Prinzip ähnlich zu dem oben beschriebenen "Card Sorting", wobei der wesentliche Unterschied darin besteht, dass der Teilnehmer auf vorgegebene Kategorien zurückgreift, anstatt sie selber zu definieren. Mehrere Durchläufe mit unterschiedlichen Teilnehmern liefern zweierlei Ergebnisse: Zum einen natürlich eine endgültige Gruppierung aller Inhalte und zum anderen eine Bewertung des vorherigen Entwurfsschrittes der "Kategoriedefinition". Anhand der Eindeutigkeit, mit der die Inhalte zu einer bestimmten Kategorie durch verschiedenen Teilnehmern zugeordnet werden, lässt sich die Qualität der Kategorien ablesen. Eine uneindeutige Zuordnung verschiedener Inhalte ist grundsätzlich ein Zeichen für eine schlechte Wahl der Kategorien. In so einem Fall bietet es sich an, den Vorgang mit anderen Kategorien zu wiederholen.
Da auch die Reihenfolge der Inhalte innerhalb einer Kategorie von Bedeutung ist (vgl. Kapitel 5.1), sollte in einem zusätzlichen Schritt eine derartige Sortierung durchgeführt werden. Dies kann wiederum partizipativ geschehen, d.h. dass die Teilnehmer die Einträge einer Kategorie in eine ihnen logische Reihenfolge bringen.

Chunking
Über die vorherigen Schritte ist die Zugehörigkeit einzelner Inhalte zu den "Containern" bestimmt worden. Zusätzlich müssen die Einheiten "Page" bzw. "Object" festgelegt werden. In Kapitel 7.2 ist zunächst davon ausgegangen worden, dass jeder Inhalt genau einer Page entspricht. Genauso gut können aber auch die Inhalte eines Containers zu einer Page zusammengefasst werden, oder aber ein einzelner Inhalt gemäß seiner Chunks in mehrere Pages aufgespalten werden.
Die Frage nach einem korrekten Vorgehen in dieser Richtung entspricht dem Granularitätsproblem (vgl. Kapitel 2.1) und hängt von verschiedenen Faktoren ab. Beispielsweise wird in [Nielsen 99] darauf hingewiesen, dass lange Texte problematisch sind, bei denen im Webbrowser "gescrollt" werden muß. Für eine differenziertere Betrachtung dieses Sachverhaltes sei auf [Hoffmann 97] verwiesen.

Übersichtspages
Als Zwischenergebnis liegt eine einfache Struktur vor, in der Pages und Objects mittels Container gruppiert sind. Für die Entstehung einer Hierarchie aus Pages ist es notwendig, dass die entsprechenden Container mit einer Übersichtspage versehen werden („Container Page“), die jeweils eine Zusammenfassung der gruppierten Inhalte darstellt. Im ersten Durchlauf der in diesem Kapitel beschriebenen Schritte ist in diesem Zusammenhang z.B. auch die Homepage festzulegen.

Obwohl eine derartige Page nicht unbedingt neue Inhalt repräsentiert, sondern eher zusammenfassender Natur sein sollte, muss sie dennoch entsprechend der Ausführungen aus Kapitel 7.2 genauer definiert werden und sollte explizit mit ins Content-Inventory aufgenommen werden. Damit ist sichergestellt, dass in den weiteren Schritten, wie z.B. der Darstellungsdefinition, darauf Bezug genommen werden kann.

Iteration
Nach einem ersten Durchlauf der obigen Schritte sind die Oberkategorien der hierarchischen Struktur festgelegt worden, was allerdings nur als erster Schritt der Strukturbildung angesehen werden kann. Durch wiederholtes Ausführen dieser Schritte in Bezug auf die einzelnen Kategorien kann die Hierarchie noch verfeinert werden. Dies kann iterativ so lange durchgeführt werden, bis eine ausreichende Strukturierung der Inhalte erlangt wurde.

7.3.2. Entwicklungsdokument

Die in diesem Kapitel definierte Struktur gilt es auf geeignete Art und Weise zu dokumentieren. In der gängigen Literatur werden derartige Strukturen als sog. "Flowcharts", "Sitemaps" bzw. "Blueprints" visualisiert und dokumentiert (vgl. [Rosenfeld & Morville 98, S. 154f] oder [Shiple 98]). Dabei handelt es sich jeweils um ein und dasselbe Konzept, nach dem die Knoten einer Hierarchie gemäß ihrer Vorgänger-Nachfolger-Beziehung auf einer Fläche (z.B. einem Blatt Papier) vertikal angeordnet werden. Oftmals wird diese Beziehung zusätzlich durch verbindende Linien repräsentiert, wodurch eine baumartige Struktur entsteht. Sofern die komplette Darstellung der Struktur zu unübersichtlich wird, können auch Teile der Struktur mittels separater Teilbäume dargestellt werden (vgl. [Rosenfeld & Morville 98, S. 154ff]). Zur Verwendung einer besonderen Symbolsprache, in der insbesondere die Konzepte von Page, Object oder Container berücksichtigt werden, sei auf Abbildung 5.3 verwiesen. In Kapitel 8 wird eine Möglichkeit vorgestellt, mit der sich eine Struktur über die Mittel eines Dateisystems beschreiben lässt, und sich somit ebenfalls zur Dokumentation eignet.

7.4. Navigation

Beim Entwurf eines Navigationssystems geht es im Prinzip darum, festzulegen, auf welche Art und Weise die einzelnen Pages miteinander verknüpft sind, d.h. in welcher Form die Hyperlinks zwischen diesen gesetzt sind. Ziel ist es, dass der Benutzer ein Instrument in die Hand bekommt, welches ihn beim Auffinden gesuchter Informationen unterstützt. Dazu ist es nicht in erster Linie entscheidend, dass die Hyperlinks das Mittel zum Wechsel von einer Page zur nächsten darstellen, sondern dass sie in einer Art und Weise präsentiert werden, die eine gradlinige Führung des Benutzers zur erwünschten Information bewirken.
Ein erster Schritt dazu stellt der Entwurf einer Struktur (vgl. Kapitel 7.3) dar, in der die Inhalte in einer für den Benutzer verständlichen Art und Weise geordnet werden. Ein Navigationssystem setzt auf dieser Struktur auf und definiert die Hyperlinks und Kontextinformationen derart, dass zum einen das mentale Modell des Benutzers von dieser Struktur unterstützt wird (vgl. Kapitel 2.7) und zum anderen eine komfortable Bewegung durch diese Struktur gewährleistet ist.

Ziel des Navigationsentwurfs soll es sein, ein allgemeingültiges und konsistentes Schema zu entwickeln, was sich auf mehrere Pages gleichermaßen anwenden lässt. Dieses Schema soll hier entsprechend Kapitel 4.7 als ein Algorithmus behandelt werden, bei dessen Anwendung Bezug auf die Struktur genommen wird. Statt also

separat für jede Page die Hyperlinks und Kontextinformationen einzeln zu definieren, soll dies in einem abstrakten und allgemeingültigen Navigationssystem geschehen.

Zwar ist bei Navigationselementen auch deren Visualisierung von großer Bedeutung (vgl. [Turner 99]), dennoch soll sich der Entwurf eines Navigationssystems an dieser Stelle auf das abstrakte Beschreiben der Hyperlinks und Kontextinformationen beschränken. Die Aspekte der "Darstellung" werden in einem separaten Schritt in Kapitel 7.6 behandelt, während ein spezieller Aspekt der Hyperlinks, nämlich die „Labels“, ebenfalls separat im Kapitel 7.5 beschrieben werden.

7.4.1. Navigationselemente

Es ist üblich, dass sich ein Navigationssystem aus mehreren einzelnen Navigationselementen zusammensetzt, die jeweils ein eigenes konsistentes System darstellen. Diese Untersysteme werden seitens der Darstellung häufig unterschieden, indem ihnen z.B. entsprechend ihrer Wichtigkeit eine mehr oder minder prominente Position im Layout zugewiesen wird.

In [Rosenfeld & Morville 98, S. 53ff] werden eine Reihe derartiger Navigationselemente vorgestellt und diskutiert und können grundsätzlich als Muster oder Beispiele dienen. Allerdings soll darauf nicht näher eingegangen werden, sondern hier nur die Unterscheidung in "Integrated Navigation Elements" und "Remote Navigation Elements" beschrieben werden: Bei ersteren handelt es sich um solche Navigationselemente, die auf jeder Page zusätzlich zum eigentlichen Inhalt enthalten sind und in Form von Hyperlinks und Kontextinformationen auftreten. Sie dienen in Abhängigkeit vom augenblicklichen Kontext dazu, eine Menge von sinnvollen Navigationsalternativen aufzuzeigen und offenbaren damit nur einen Ausschnitt aus allen möglichen Navigationszielen.
Dahingegen stellen die "Remote Navigation Elements" solche Elemente dar, die auf explizit angelegten Pages untergebracht werden und dort als zentrales Element für einen Überblick über alle möglichen Navigationsziele der Website sorgen. Dies kann in Form einer sog "Sitemap" realisiert sein, aber auch über eine "Table of Contents" oder einem "Index". Bemerkenswert sind letztere Elemente deshalb, weil deren Verwendung das Hinzufügen einer neuen Page in die Website bedingt. Ähnlich wie die Übersichtsseiten (vgl. Kapitel 7.3.1) enthalten diese Seiten ebenfalls keinen expliziten Inhalt, sondern dienen ausschließlich der Navigation.
Detailliertere Beispiele für diese beiden Typen von Navigationselementen werden in Kapitel 8.4.1 aufgeführt.

7.4.2. Vorgehen und Probleme

In [Rosenfeld & Morville 98, S. 70f] wird erklärt, dass es für den Entwurf eines Navigationssystems keine Standardrezeptur gibt, sondern ganz speziell auf die aktuellen Umstände wie z.B. der Zielsetzung eingegangen werden muss. Konsequenterweise findet sich daher in der Literatur auch keine befriedigende Beschreibung zum Entwurf.

Insofern soll hier ein Verfahren vorgestellt werden, welches zwar nur sehr abstrakt ist, zumindest aber einen Anhaltspunkt für ein konkretes Vorgehen liefern sollte. Bei diesem Verfahren soll von einer generischen Lösung ausgegangen werden, die dann durch Adaption der konkreten Umstände verfeinert wird. Die generische Lösung sieht vor, dass von jeder Page ein Hyperlink auf die jeweils übergeordnete Page gesetzt wird und außerdem jeweils ein Hyperlink auf alle direkt untergeordneten (vgl. Kapitel

4.7). Damit wird die Struktur aus Containern in eine Struktur aus (strukturellen) Hyperlinks überführt. Als Kontextinformation soll zudem ein Pfad definiert werden, der ausgehend von der Homepage alle Knoten bis zur jeweiligen Page auflistet.

In Bezug auf das Beispiel aus Abbildung 5.2b, würde z.B. von der Page „Personen“ jeweils ein Hyperlink auf die übergeordnete Page „Firma XYZ“ und auf die untergeordneten Pages „Herr Müller“ und „Frau Schulze“ gesetzt werden. Die Kontextinformation des Pfades würde entsprechend die Elemente „Firma XYZ“ und „Personen“ enthalten.

Sofern dieses Schema angewandt wird, ist zumindest jede Page innerhalb der Website erreichbar, auch wenn dies im Einzelfall nicht übermäßig komfortabel erscheint. Ausgehend von dieser Lösung sollte dann für jede einzelne Page festgelegt werden, welche zusätzlichen strukturellen Hyperlinks und Kontextinformationen in Bezug auf die Bedürfnisse der Benutzer sinnvoll erscheinen. Dabei werden sich mit Sicherheit Entwurfsmuster herauskristallisieren, die mittels eines Algorithmus beschrieben werden können.
Die Schwierigkeit eines solchen Navigationsentwurfs besteht in dem Spagat, einerseits für jede Page alle sinnvollen Hyperlinks und Kontextinformationen festzulegen, diese aber andererseits einem möglichst allgemeingültigen und konsistenten System unterzuordnen. Dieses Problem soll anhand eines Beispiels erläutert werden:

Es soll beispielhaft angenommen werden, dass das Navigationssystem über oben beschriebenes Schema definiert wird. Während dies z.B. für die Homepage mit nur einigen wenigen untergeordneten Pages sinnvoll ist, wird dies bei Pages mit vielen "Nachkommen" problematisch werden: Dazu sei angenommen, es liegt eine "News"-Kategorie vor, in der sich über viele Monate hinweg eine große Anzahl von Meldungen angesammelt hat. Sofern von einer Übersichtspage aus das obige Schema angewandt wird, wird dies in einer großen Anzahl gleichwertiger Hyperlinks resultieren, was in dieser Form nicht mehr überschaubar ist.
Stattdessen wäre an dieser Stelle ein modifiziertes Navigationsschema sinnvoll: Dies könnte beispielsweise über eine Untergliederung der Hyperlinks geschehen, indem sie in zwei Gruppen unterteilt werden. Eine Gruppe umfasst die Hyperlinks auf die jeweils neuesten Meldungen, während der anderen Gruppe alle anderen Hyperlinks zugeordnet werden. Über eine entsprechenden Visualisierung dieser beiden Gruppen kann z.B. die erste gegenüber der zweiten hervorgehoben werden und damit für mehr Überblick sorgen.

Der Entwurf eines Navigationssystems wird als Ergebnis eine Menge von Navigationselementen bzw. Navigationsschemata hervorbringen, die jeweils in unterschiedlichen Kombinationen Anwendung auf eine oder mehrere Pages finden.

7.4.3. Entwicklungsdokumente

Bei der Dokumentation des Navigationsentwurfes besteht die Schwierigkeit, die eher abstrakten Schemata in einem Dokument zu beschreiben. Dies wird in [Rosenfeld & Morville 98, S. 158f] und [Fleming 98, S. 94] eher implizit erledigt, indem dies anhand von Beispielen dokumentiert wird. Dabei wird der Entwurf auf eine oder mehrere Pages angewandt und dies mittels eines "Papierprototypen" visualisiert.
Das Problem derartiger Beispiele ist, dass sie nicht ausreichend dokumentieren, was hinter dem jeweiligen Entwurf für ein Konzept steckt und somit Interpretationsspielräume gelassen werden. Sinnvoller wird dies von [Shiple 98] gelöst, indem dort mit-

tels eines Prosatextes beschrieben wird, wie die einzelnen Hyperlinks zu setzen sind. Über sprachliche Mittel können dabei auch Fallunterscheidung formuliert werden, um somit Besonderheiten zu berücksichtigen.
Um die oben erwähnte Sichtweise auf ein Navigationssystem als ein "Algorithmus" aufzugreifen, soll hier weiterführend vorgeschlagen werden, die Dokumentation mittels einer Programmiersprache zu tätigen. Dazu ist im Prinzip jede Programmiersprache oder Pseudoprogrammiersprache geeignet, wobei aber geeignete Beschreibungsmöglichkeiten z.B. für die Bezugnahme auf die Struktur gefunden werden müssen. Eine Lösung dafür soll in Kapitel 8.2.4 vorgestellt werden.

Es ist jedoch nicht ausreichend, nur das algorithmische Schema der Navigationselemente zu definieren, sondern es muss zusätzlich für jede Page festgelegt werden, welche der Elemente dort Anwendung finden sollen. Dazu muss jeweils eine Auswahl aus der Gesamtmenge getroffen und beispielsweise als eigenständiger Inhalt einer Page beschrieben werden (vgl. Kapitel 7.2.5).

7.4.4. Evaluationsprobleme

Die Evaluation der in diesem Schritt getätigten Entwürfe stellt aus verschiedenen Gründen ein Problem dar: Einerseits erschwert die abstrakte Beschreibung dieses Systems das konkrete Erfassen der Umstände und Probleme. Andererseits erscheint es aber auch nicht sinnvoll, die Navigation unabhängig von den anderen Entwurfselementen zu evaluieren: Wenn man sich vor Augen führt, dass die Navigation ein Mittel zum Zweck ist, d.h. sie den Benutzer bei der Suche nach einer Information unterstützt, dann müsste eine Evaluation auch unter Einbeziehung des Inhalts stattfinden. Weiterhin kann auch der visuelle Aspekt eines Navigationssystems nicht berücksichtigt werden, obwohl dieser von entscheidender Bedeutung ist (s.o.). So gesehen sollte die Navigation unter Integration der übrigen Entwurfsschritte stattfinden, weshalb hier lediglich auf Kapitel 7.7 und dem dort beschriebenen integrierten Prototyping verwiesen werden soll.

7.5. Labels

In Kapitel 2.7 wurde der Begriff des „Labels“ im Kontext von Hyperlinks behandelt und dabei verdeutlicht, dass es sich damit um einen wichtigen Bestandteil der Navigation handelt. Ein Label repräsentiert mittels Text und Icon den Inhalt einer Page und unterstützt den Benutzer zu entscheiden, ob er den Hyperlink benutzen soll oder nicht.
Die Schwierigkeit bei der Gestaltung eines Labels besteht in der Gradwanderung zwischen einerseits der Bereitstellung ausreichender Information und anderseits der beeinträchtigten Übersichtlichkeit. Der Benutzer muss zum einen das Label ausreichend verstehen, darf dabei aber nicht von der Vielzahl aller Informationen überfordert werden.
Ebenso wie in vielen anderen Bereichen auch, ist die Konsistenz von Labels ein entscheidender Faktor für deren Erfolg. Nur wenn sie nach einem einheitlichen Prinzip gestaltet worden sind, ist der Benutzer in der Lage, die dadurch angebotenen verschiedenen Handlungsoptionen in Beziehung zu setzen und sie dadurch zu verstehen [Rosenfeld & Morville 98, S. 74]. Insofern dient der separate Entwurf der Labels dem Herstellen einer solchen Konsistenz.

Die Unterscheidung zwischen assoziativen und strukturellen Hyperlinks soll auch an dieser Stelle Konsequenzen haben: Es soll davon ausgegangen werden, dass die Labels der assoziativen Hyperlinks schon im Rahmen der Inhaltsdefinition (Kapitel

7.2) entworfen worden sind. Da derartige Hyperlinks in den Kontext des direkt umgebenen Inhalts eingebunden sind, ist der Faktor Konsistenz für sie nicht so entscheidend. Wichtiger ist dies für die strukturellen Hyperlinks der Navigation (vgl. Kapitel 7.4), worauf sich dieser Abschnitt letztendlich beziehen soll.

7.5.1. Vorgehen

Der Entwurf eines Labels besteht grob gesehen aus der Festlegung der zwei möglichen Komponenten, nämlich dem Icon und/oder dem Text. Entsprechend muss dies für alle Hyperlinks der strukturellen Navigation innerhalb der Website erledigt werden. Anstatt die Labels für jeden Hyperlink einzeln zu gestalten, soll dies in Bezug auf jede Page getätigt werden: Das bedeutet, dass für jede Page – und nicht für jeden Hyperlink - einheitlich festgelegt wird, welches Label ein Hyperlink auf eben diese Page tragen soll. Das bedeutet in der Konsequenz, dass alle Hyperlinks auf dieselbe Page auch dasselbe Label benutzen. Dies macht für die konsistente Gestaltung eines Labelingsystems Sinn: Andersherum würden nämlich verschiedene Labels für ein- und dieselbe Page einer Konsistenz entgegenlaufen und wären somit potentielle Ursachen für Navigationsprobleme. Ein Vorteil eines derartigen Vorgehens liegt zudem darin, dass der Entwurf weitgehend unabhängig von dem konkreten Navigationsentwurf stattfinden kann und somit eine bessere Arbeitsteilung erlaubt.

Da ein Label den Inhalt einer Page zusammenfassen sollte, muss logischerweise bei dessen Gestaltung auch Bezug auf diesen Inhalt genommen werden. Die Inhaltsangabe einer Page sollte sich an der jeweiligen Zielsetzung bzw. dem Zweck eines Inhalts gemäß der Bedarfsanalyse beziehen. Zusätzliche Orientierungshilfen stellen dabei die Vorgaben aus der Konzeption oder der Gesamtzielsetzung dar. Weiterhin bietet sich die Verwendung von standardisierten Begriffen, wie z.B. "Über Uns", (s. [Rosenfeld & Morville 98, S. 76]) und die Einbeziehung von Heuristiken an. Dabei sei insbesondere auf den Verzicht von Labels wie "click here" hingewiesen [Miller 98].
Ein erster Entwurf von Labels ist implizit über den jeweiligen Bezeichner im Content-Inventory schon gegeben, wobei auch aus dem Schritt der Strukturierung derartige Ergebnisse hervorgegangen sein werden: Über die dort verwendeten partizipativen Maßnahmen liegen u.U. auch seitens der Benutzer Vorschläge für die Labels vor.

In [Rosenfeld & Morville 98, S. 93f] wird beschrieben, dass nach der erstmaligen Festlegung aller Labels ein "Fine-Tuning" in Bezug auf übergeordnete Prinzipien stattfinden sollte. Dazu wird eine Modifikation z.B. in Bezug auf einheitliche Begriffe bzw. einer einheitlichen Sprache vorgenommen.

7.5.2. Evaluation

Für die Evaluation der Ergebnisse sind eine Reihe von Methoden bekannt, anhand derer die Qualität der Labels durch den Benutzer bewertet wird. Diese Methoden sollen hier kurz vorgestellt werden:

Category Description
Dieses Verfahren ist in [Fuccella & Pizzolato 98] beschrieben und dient der Entscheidung, wie gut ein Label jeweils gewählt wurde. Eine Testperson bekommt dazu eine Menge von Labels vorgelegt und soll nur auf Basis dieser Angaben erläutern, welche Inhalte sie dort jeweils erwartet. Dies kann in Form einer Beschreibung aber auch über Beispiele geschehen. Zusätzlich soll noch eine Bewertung anhand einer Skala darüber stattfinden, welches Vertrauen in das eigene Urteil gelegt wird.

Ein prinzipiell ähnliches Prinzip wird im Verfahren des "Icon Intuitiveness" von [Nielsen & Sano 94] vorgestellt und bezieht sich allerdings im Wesentlichen auf die Icons. Ziel ist es, herauszufinden, ob ein Icon auch ohne ein Textlabel ausreichend verständlich ist.
Bei der Auswertung der Ergebnisse sollten Angaben, die deutlich vom tatsächlichen Inhalt abweichen, in einer Überarbeitung des Labels resultieren. Eine subjektive Unsicherheit in Bezug auf das eigene Urteil lässt sich hingegen möglicherweise über geringfügigere Modifikationen am Label beheben.

Category Labeling
Bei der hier wiederum aus [Fuccella & Pizzolato 98] entnommenen Methode geht es darum, dass eine Aussage über die beste von mehreren Labelingmöglichkeiten getroffen wird. Dazu wird der Testperson der jeweilige Inhalt und eine Menge möglicher Labels vorgelegt, wobei das jeweils am besten passende auswählt werden soll. Zusätzlich können die Teilnehmer auch alternative Vorschläge für, ihrer Meinung nach, besser passende Labels machen. Die Ergebnisse dieses Vorgehens lassen sich quantitativ auswerten, d.h. anhand der Menge der Zuordnungen lassen sich direkt die bestgeeigneten Labels ablesen.

Card Distribution to Icons
Dieses von [Nielsen & Sano 94] vorgestellte Verfahren eignet sich dafür, die Kombination eines Icons mit einem Text zu bewerten. Dazu werden der Testperson eine Menge von voneinander getrennte Texten und Icons vorgelegt, bei denen eine Zuordnung zu treffen ist. Das bedeutet, dass jedem Icon der vermeintlich passende Text zugeordnet werden soll. Deutliche und häufige Fehlzuordnungen lassen sich in der Auswertung erkennen und können somit Hinweise zur Überarbeitung der jeweiligen Labels liefern.

7.5.3. Entwicklungsdokumente

Eine Dokumentation der Ergebnisse soll hier unter der Annahme beschrieben werden, dass ein Label als Bestandteil einer Page anstatt als Bestandteil eines Hyperlinks gesehen wird (s.o.). Das bedeutet konsequenterweise, dass ein einzelnes Label als „Inhalt“ (oder genauer: Als Attribut) gesehen werden kann und an entsprechender Stelle dokumentiert wird. Insofern soll hier nur auf das Kapitel 7.2.5 verwiesen werden.
Natürlich kann eine Dokumentation auch direkt im Content-Inventory stattfinden, so wie es implizit über den "Titel" auch schon der Fall ist. In [Rosenfeld 97] wird zusätzlich eine sog. "Labeling Table" erwähnt, die ähnlich einem Content-Inventory aufgebaut ist. Vorteil einer derartigen Dokumentation ist die Übersicht über die Labels, die dabei gewahrt bleibt.

7.6. Darstellung

Der Entwurf der Darstellung hat das Ziel, die Ergebnisse der übrigen Schritte zu visualisieren und somit letztendlich dem Benutzer zugänglich zu machen. Die Ergebnisse dieser Visualisierung beschreiben im Prinzip die Form, in der der Benutzer die Website wahrnehmen wird, d.h. es wird ein "Interface" definiert, welches zwischen den Inhalten und dem Benutzer fungiert. Da die Darstellung letztendlich über HTML definiert wird, muss am Ende des hier beschriebenen Entwurfs eindeutig festgelegt sein, wie jede einzelne Page in HTML überführt werden kann.

Dazu werden für jeden einzelnen Inhalt die Darstellungsaspekte gemäß Kapitel 5.3 festgelegt, was z.B. für einen Text bedeutet, dass dessen Typographie und Farben definiert werden. Alle inhaltlichen Elemente einer Page werden außerdem integriert über ein Layout in Beziehung gesetzt.

7.6.1. Vorgehen

Wie schon angedeutet wurde, müssen für jedes einzelne inhaltliche Element die Komponenten der Darstellung festgelegt werden. Dazu wird sinnvollerweise von der Granularität der "Page" ausgegangen und deren Darstellung Schritt für Schritt verfeinert und festgelegt. Je nachdem, von welchem Datentyp die einzelnen Chunks sind, werden unterschiedliche Darstellungselemente relevant sein. Während es bei Objects im Wesentlichen nur deren Größe und Position ist, kommen bei Texten andere Elemente hinzu, wie z.B. die Typographie.

Unter anderem in [Shiple 98] wird dem Layout eine besondere Rolle zugeschrieben, was sich in einem bestimmten Vorgehen niederschlägt. Dies soll hier in Anlehnung an o.g. Quelle geschildert werden: Ausgegangen werden soll von einer zweidimensionalen Fläche, auf der für die einzelnen Chunks einer Page jeweils Bereiche reserviert werden. Dies geschieht durch Festlegen viereckiger Blöcke, die nichtüberlappend jeweils an einer bestimmten Position und in einer bestimmten Größe auf dieser Fläche angeordnet werden. Jeder einzelnen Block repräsentiert einen Chunk, der über Zwischenräume von den anderen Blöcken bzw. Chunks getrennt ist. Der Tatsache, dass sich Chunks hierarchisch unterteilen lassen, kann an dieser Stelle über ein rekursives Vorgehen Rechnung getragen werden: Jeder Block kann iterativ wiederum nach obigem Schema unterteilt werden, was anhand von folgender Abbildung demonstriert werden soll. Für jeden einzelnen Block können dann die weiteren Parameter, wie z.B. Hintergrundfarbe, typographische Elemente usw. festgelegt werden.

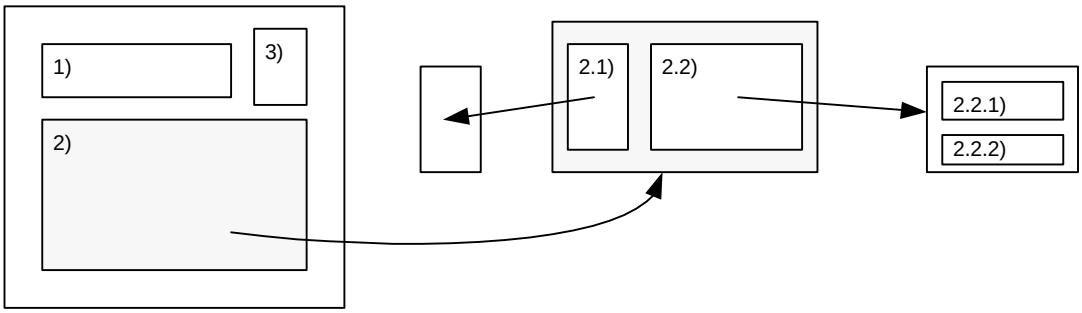

Abbildung 7.1: Beispiel einer hierarchischen Darstellungsdokumentation: Das Feld 2) wird schrittweise in seine Einzelkomponenten zerlegt.

Da die Darstellung der einzelnen Inhalte in Bezug auf Benutzbarkeit konsistent sein sollte, sollte beim Entwurf nicht jede Page separat betrachtet werden, sondern in Hinblick auf die komplette Website stattfinden. Dabei kann das Entwurfsmuster "Dokumenttyp" (vgl. Kapitel 4.7) von hoher Bedeutung sein: Ein derartiges Muster beschreibt eine ähnliche inhaltliche Struktur, die sich konsequenterweise auch in einer ähnlichen Darstellung niederschlagen sollte. Daraus können seinerseits Entwurfsmuster in Form von "Templates" entstehen.

Der Entwurf der Darstellung muss sich an besonderen Umständen orientieren, die durch die technische Umgebung der Zielplattform vorgegeben ist. Unterschiedliche

technische Ausstattungen, z.B. in Form der Grafikkarte oder den verwendeten Browsern, bedingen mehr oder weniger drastische Unterschiede dabei, wie sich die Darstellung letztendlich für den Benutzer präsentiert. Derartige Umstände ergeben sich aus der Bedarfsanalyse (s. Kapitel 6.3), bei der beispielsweise die technischen Voraussetzungen der Benutzer ermittelt wurden.
Besonderheiten wie sie sich durch das "Trägermedium" HTML ergeben, müssen dabei ebenso berücksichtigt werden wie die Umstände, die durch das WWW bedingt werden. So sollte beispielsweise aus Gründen der Benutzbarkeit auf übertragungsintensive Grafiken verzichtet werden (vgl. [Fleming 97]).

7.6.2. Evaluation

Eine Methode, um die visuellen Komponenten einer Page zu evaluieren wird in [Tullis 98] beschrieben. Dazu wird den Teilnehmern die Darstellung einzelner Pages als "Papierprototyp" vorgelegt, wobei konkrete Inhalte mittels sinnleerer Platzhalter repräsentiert werden. Zusätzlich liegt eine separate Liste mit allen inhaltlichen Elemente der jeweiligen Page (z.B. "Überschrift", "Navigation", ...) vor. Aufgabe des Teilnehmers ist es, eine Zuordnung von den Inhalten zu den korrespondierenden darstellenden Bereichen zu treffen. Dies geschieht durch Einzeichnen rechteckiger Felder und deren Beschriftung. Inhalte, denen kein Bereich zugeordnet werden konnte, sollen als fehlend gekennzeichnet werden.
Ziel ist es, herauszufinden, ob die visuellen Elemente deutlich genug herausgearbeitet wurden, so dass sie unmittelbar in ihrer Bedeutung wahrgenommen werden können. Die Ergebnisse ermöglichen eine quantitative Auswertung und lassen für jedes einzelne Element erkennen, ob eine Überarbeitung nötig ist.

Zusätzlich zu diesem Vorgehen kann auch eine subjektive Bewertung des Entwurfs in Hinblick auf den Gesamteindruck bzw. spezieller Aspekte, wie Format, Farbe usw., von den Teilnehmern eingefordert werden. Auf diese Weise lassen sich ansonsten schlecht messbare Entwurfsentscheidungen eines Entwurfs, wie z.B. ästhetische Aspekte, bewerten.

7.6.3. Entwicklungsdokumente

Da die "Darstellung" visueller Natur ist, erscheint es nur konsequent, auch deren Dokumentation in dieser Weise stattfinden zu lassen. Das bedeutet, dass Grafiken erstellt werden, in denen alle relevanten visuellen Aspekte erfasst sind. Herkömmliche Möglichkeiten stellen die sog. "Page Schematics" [Kimen 99] oder "Layout Grids" [Shiple 98] dar, in denen im Wesentlichen das Layout in Form eines Blockdiagramms verzeichnet ist. Andere Möglichkeiten, wie z.B. die "Screens" repräsentieren eine komplette grafische Ausarbeitung einer Page, so dass alle relevanten visuellen Aspekte gleichzeitig dargestellt werden können. Konkrete Inhalte können in einer derartigen Ausarbeitung entweder beispielhaft eingefügt werden, oder aber mittels Blindtext repräsentiert werden.
Zwar muss theoretisch die Darstellung jeder einzelnen Page auf diese Weise dokumentiert werden, was sich aber mittels „Templates“ (vgl. Kapitel 4.7) einschränken lässt: Für jeden definierten Dokumenttyp wird ein Template angelegt, wodurch die Darstellung aller entsprechenden Pages abgedeckt und vereinheitlicht wird.

Ein grundsätzliches Problem der hier beschriebenen Dokumentation liegt darin, dass sich damit bestimmte Umstände nicht dokumentieren lassen. So beispielsweise das dynamische Layoutverhalten von HTML: Je nachdem, wie groß die zur Verfügung stehende Fläche ist, wird ein Layout in HTML vom Webbrowser unterschiedlich dar-

gestellt. Andere dynamische Effekte, wie z.B. das "Rollover" können ebenfalls nicht über die oben genannte statische Möglichkeit dargestellt werden.
Als Lösung bietet es sich an, die Entwürfe direkt in HTML zu dokumentieren, wobei konkrete Inhalte mittels Platzhalter repräsentiert werden können. Dies lässt sich auch mit Techniken wie beispielsweise XSLT (vgl. Kapitel 8.1.2) kombinieren, wo einerseits explizit das Konzept der "Templates" verfolgt wird und andererseits auch ein modularer Ansatz für die Unterstützung von "Dokumenttypen" (vgl. Kapitel 4.7) möglich ist. Zudem setzt diese Technologie auf XML auf, so dass damit eine ideale Kombination mit den in XML beschriebenen Inhalten (vgl. Kapitel 7.2.5) vorliegt.

7.7. Prototyping

In den letzten Kapiteln wurden Schritte beschrieben, in denen jeweils Teilaspekte einer Website entworfen und teilweise auch evaluiert werden. Da sich derartige Evaluationen jeweils nur auf Teile des großen Ganzen beziehen, lässt sich damit nicht ausreichend ermitteln, ob die Entwürfe in Bezug auf eine übergeordnete Zielsetzung bzw. in Bezug auf bestimmte Bedürfnisse der Benutzer ihren Zweck erfüllen. Dies lässt sich prinzipiell nur über die Evaluation in Bezug auf eine reale Umgebung erlangen (vgl. [Nielsen o.D.]), in der die Einzelentwürfe zu einem Gesamtentwurf kombiniert sind. Dazu soll noch einmal vor Augen geführt werden, dass die einzelnen Entwurfsschritte jeweils den Teil eines Ganzen behandeln, bei denen mehr oder minder starke Wechselwirkungen mit den jeweils anderen Schritten bestehen.
Um eine reale Umgebung zu schaffen, müssen die Ergebnisse der Einzelentwürfe zu einem Gesamtsystem integriert werden, d.h. Struktur, Navigation, Labels, Inhalte und Darstellung müssen zu einer kompletten benutzbaren Website kombiniert werden. Anders ausgedrückt muss es dem Benutzer prinzipiell ermöglicht werden, mittels eines Webbrowsers auf eine Struktur aus HTML-Seiten zuzugreifen.

Die Integration der einzelnen Komponenten zu einer kompletten Website ist allerdings in Bezug auf einen Entwurfsprozess problematisch: Eine solche Erstellung entspricht einer Implementation, die u.U. eine zeitaufwendige Angelegenheit darstellt: Jede einzelne Page muss in HTML erstellt und mit den anderen Pages verknüpft werden. Sofern sich bei der Evaluation Entwurfsschwächen offenbaren sollten, die eine Überarbeitung einzelner Aspekte nach sich ziehen, wird dies u.U. in einer wiederum zeitaufwendigen Neuerstellung der Website münden. Dazu sei als Beispiel eine Modifikation des Navigationssystems angeführt, die eventuell eine entsprechende Modifikation der Hyperlinks auf jeder einzelnen HTML-Seite bedingt.
Eine grundsätzliche Lösung dieses Dilemmas stellt das Konzept der Prototypen dar (vgl. Kapitel 3.3), in der jeweils nur gewisse Aspekte zu einem "ablauffähigen Modell" implementiert werden. Das bedeutet z.B. in Bezug auf eine Website, dass nicht die komplette Struktur der Pages implementiert wird, sondern vielleicht nur ein kleiner Ausschnitt, der allerdings für die Bewertung bestimmter Aspekte ausreichend ist.

7.7.1. Arten

In [Fucella & Pizzolato 99] werden in Bezug auf Websites zwei Arten von Prototypen unterschieden, die hier kurz beschrieben werden sollen. Beide Arten repräsentieren zwar jeweils nur Teile aus der Gesamtwebsite, sind aber in Bezug auf die Evaluation bestimmter Sachverhalte ausreichend.

Wire Frames
Ein Prototyp "Wire Frame" ("Drahtgitter") reduziert eine Website auf seine Navigation und den Inhalt, was das Fortlassen möglichst vieler visueller Elemente bedingt. Mit-

hilfe dieser Prototypen lassen sich Navigationsvorgänge simulieren, bei denen es nur um das Auffinden bestimmter Inhalte geht. Dabei ist selbst die Integration der Inhalte nicht unbedingt von Nöten, sondern unter Umständen reichen die Labels und das Navigationssystem aus. Somit besteht der Prototyp aus einer Menge von Seiten, die im Wesentlichen durch die Navigation miteinander verbunden sind. Inhalte werden entweder direkt integriert oder z.B. mittels "Blindtext" simuliert.

Visual Prototype
In diesem Prototypen werden alle inhaltlichen Elemente fortgelassen, so dass nur das visuelle Interface inklusive Navigation zur Verfügung steht. Realisiert wird ein derartiger Prototyp, indem alle Inhalte in Textform durch "Blindtext" ersetzt werden (vgl. Kapitel 7.6.2). Geeignet ist er, um z.B. ein "Look&Feel" der Navigation und der Darstellung zu vermitteln, anstatt dass damit "echte Aufgaben" erfüllt werden können. Aus diesem Grund wird eine Evaluation anhand dieses Prototypen ähnlich wie im o.g. Kapitel stattfinden.

7.7.2. Vorgehen

Das Durchführen einer Evaluation mithilfe eines Prototypen ist grundsätzlich schon in Kapitel 3.4.5 als "Sitereview" beschrieben, weshalb ein Vorgehen hier nur knapp erläutert werden muss. Grundsätzlich sollte es das Ziel sein, die Evaluation anhand "echter Aufgaben" mit "echten Benutzern" in einer "realen Umgebung" durchzuführen.

Bei den "echten Aufgaben" handelt es sich um solche, die von Benutzern potentiell anhand der Website bearbeitet werden würden. Alle diese Aufgaben im Rahmen einer Evaluation zu behandeln, würde mit Sicherheit den Rahmen sprengen, so dass eine repräsentative Auswahl getroffen werden muss. Grundsätzlich muss das Festlegen dieser Aufgaben in Bezug auf die Ergebnisse der Bedarfsanalyse geschehen, in der die Bedürfnisse der Benutzer ermittelt wurden. Aufgrund der Natur der hier betrachteten Website als ein Informationssystem, wird es sich bei den Aufgaben um das Auffinden bestimmter Informationen handeln, wie z.B. "Herausfinden, wer der Ansprechpartner in Bezug auf Produkt xyz ist".
Aus derartigen Untersuchungen heraus sollte ein Aufgabenkatalog aufgestellt werden, den es den Benutzern vorzulegen gilt. Ein Beispiel eines solchen Katalogs findet sich in [Weinreich 97, S. 227].

Obwohl eine "reale Umgebung" tatsächlich erst durch eine komplette Website repräsentiert wird, soll aus o.g. Gründen von einer reduzierten Implementation in Form eines Prototypen ausgegangen werden. Bei der Erstellung des Prototypen muss natürlich darauf geachtet werden, dass dies in Bezug auf die zu behandelnden "Aufgaben" (s.o.) geschieht. Das bedeutet, dass die Aufgaben mittels des Prototypen auch sinnvoll erledigt werden können. Eine Aufgabe "Finde Information xyz" macht natürlich keinen Sinn, wenn der entsprechende Inhalt im Prototypen nicht integriert ist.
Als Medium zur Erstellung der Prototypen bieten sich zweierlei an: Einmal kann eine Erstellung direkt in HTML erfolgen, andererseits stellen auch "Mock-Ups" (vgl. Kapitel 3.3) eine adäquate Lösung dar. Für die Erstellung ersterer Möglichkeit können z.B. Werkzeuge wie die "Web Site Editors" (vgl. Kapitel 4.6) verwendet werden. Deren oftmals mangelnde Möglichkeit zur Feinjustierung bestimmter Aspekte, wie z.B. der Navigation, lässt sich in Bezug auf ein Prototyping eher verschmerzen als in Bezug auf die Implementation. Eine besondere Möglichkeit zur Erstellung von Website-

Prototypen stellt der „SitePrototyper“ dar, der im folgenden Kapitel vorgestellt werden soll.

Aufgabe für die Teilnehmer der Evaluation sollte es sein, anhand des Prototypen den Aufgabenkatalog abzuarbeiten, wobei sich Methoden wie das "Thinking aloud" usw. für diesen Zweck anbieten. Bei der Ausführung dieser Tätigkeit werden die Teilnehmer auf Probleme bei der Benutzung des Prototypen stoßen, deren Gründe sich u.U. unmittelbar erkennen lassen. Grundsätzlich kommen je nach Vorgehen auch andere Auswertungsmethoden in Frage, wie z.B. die Logfileanalyse. Die Teilnehmer können auch direkt nach ihren subjektiven Eindrücken in Bezug auf diverse Aspekte der Website befragt werden, die auf andere Weise nicht ermittelt werden können.

Die Erkenntnisse, die über das hier beschriebene Prototyping gewonnen werden, sollten konsequenterweise in der Überarbeitung des Entwurfs münden. Dazu sollte ein wiederholtes Durchlaufen der entsprechenden Entwurfsschritte stattfinden. Auf diese Weise ergibt sich ein iteratives Vorgehen, bei dem der Entwurf evolutionär weiterentwickelt wird. Sofern die Entwickler der Meinung sind, dass keine schwerwiegenden Benutzbarkeitsprobleme im Entwurf mehr vorliegen, kann anstelle eines Prototypen die Website auch direkt implementiert und über das WWW bereitgestellt werden. Im Rahmen des Entwurfsprozesses ergeben sich daraus ganz neue Möglichkeiten: Zum einen müssen reale Bedingungen nicht mehr simuliert werden, und zum anderen bestehen ganz neue Möglichkeiten, was die Bedarfsanalyse und die Rekrutierung von Evaluationsteilnehmern angeht. In diesem Fall muss nicht mehr auf Repräsentanten und Hypothesen (vgl. Kapitel 6.3) zurückgegriffen werden, sondern es liegen tatsächlich reale Verhältnisse vor.

8. SitePrototyper - Ein Entwurfswerkzeug

In den vorherigen Kapiteln wurde u.a. ein Entwurfsprozess dargestellt, in dem eine Website mittels Einzelkomponenten spezifiziert und dokumentiert wurde. Es wurde im letzten Kapitel festgestellt, dass eine Evaluation der Entwurfsergebnisse in Bezug auf Benutzbarkeit unbedingt integriert stattfinden sollte. Das bedeutet, dass im besten Fall eine vollständig navigierbare Website vorliegt. Die Erstellung einer derartigen Website bzw. eines Prototypen ist mitunter eine zeitintensive Angelegenheit und kann im Fall von "Wegwerfprototypen" (vgl. Kapitel 3.3) zudem nur einmalig verwendet werden. Das bedeutet, dass jede Änderung des Entwurfs unter Umständen in der kompletten Neuerstellung einer Website bzw. eines Prototypen mündet. Ein Grund für den hohen Zeitaufwand stellt die Tatsache dar, dass die Ergebnisse des Entwurfs in eine andere als die vorliegenden Form überführt werden müssen. Beispielsweise müssen die abstrakt schematisierten Navigationselemente jeweils auf jede Page angewandt werden.

Aus den hier genannten Gründen ist im Rahmen dieser Diplomarbeit das Werkzeug "SitePrototyper" entwickelt worden, welches sich dieser Problematik annimmt. Die Grundidee besteht darin, dass die Ergebnisse der einzelnen Entwurfsschritte "Inhalt", "Struktur", "Navigation", "Labels" und "Darstellung" in einer bestimmten Art und Weise dokumentiert werden, in der maschineller Zugriff ermöglicht wird. Als Konsequenz daraus kann eine automatische Integration dieser Komponenten stattfinden, die eine komplett navigierbare Website zum Ergebnis hat. Dadurch besteht die Möglichkeit, ohne zusätzlichen Arbeitsaufwand und zu einem sehr frühen Zeitpunkt, eine Evaluation anhand eines so entstandenen Prototypen durchführen zu können.

8.1. Architektur

Die automatische Integration von Entwurfsergebnissen mittels des SitePrototypers bedingt, dass diese Ergebnisse in einer maschinenlesbaren Form vorliegen. Dazu sollen alle diese Ergebnisse als Dateien innerhalb des Dateisystems dokumentiert werden, wobei je nach Art des Ergebnisses ein spezielles Format verwendet werden muss. Dabei handelt es sich um Folgende:

- Die **Struktur** wird mittels speziell angelegter Dateien und Ordner innerhalb des Dateisystems repräsentiert.
- Die **Inhalte** und **Labels** von Pages werden grundsätzlich mit der Auszeichnungssprache XML beschrieben, wobei Objects als Binärdateien abgelegt werden.
- Die **Navigation** wird über Algorithmen in der Programmiersprache JAVA ausgedrückt.
- Die **Darstellung** wird über die Transformationssprache XSLT beschrieben.

Derartige Formate sowie die übrigen Vorraussetzungen des SitePrototypers bedingen eine bestimmte Architektur von Softwarekomponenten, die im Folgenden erläutert werden soll:

Grob gesehen ist der SitePrototyper in ein System folgender Komponenten integriert: Dem Dateisystem, das als Datenspeicher der Entwurfsergebnisse dient, einem Webserver, sowie einem Webbrowser. Während der Entwickler die Komponenten der Website durch Erzeugen und Modifikation von Dateien und Ordnern innerhalb des

Dateisystems bearbeitet, kann der jeweils aktuelle Stand der Website über das herkömmliche System von Webbrowser und Webserver abgerufen und evaluiert werden.

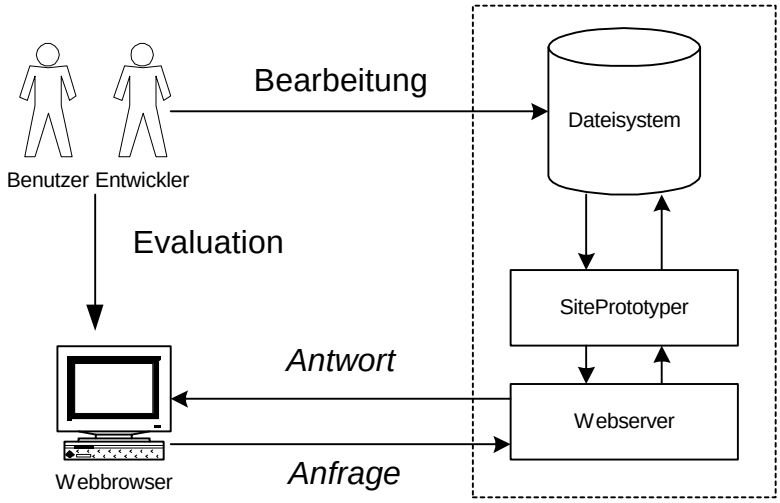

Abbildung 8.1: Der Entwickler greift bearbeitend auf das Dateisystem des SitePrototypers zu, während die Ergebnisse dieser Bearbeitung mittels eines Webbrowsers abgerufen und evaluiert werden können. Dazu werden sie vom SitePrototyper transformiert und über einen Webserver bereitgestellt.

Als Zwischenglied zwischen dem Webserver und dem SitePrototyper wird eine Technologie verwendet, die als "Servlets" bekannt ist. Diese soll zusammen mit den verwendeten Formaten XML und XSLT im Folgenden erläutert werden. Die übrigen Komponenten wie "JAVA" sowie das "Dateisystem" werden von ihrem Prinzip her als bekannt vorrausgesetzt und sollen hier nicht weiter behandelt werden.

Der SitePrototyper besteht insgesamt aus einer Kernkomponente, die im Rahmen dieser Diplomarbeit entwickelt wurde und in der Programmiersprache JAVA implementiert wurde, sowie einigen ebenfalls JAVA-basierten Fremdkomponenten. Damit ist das Gesamtsystem prinzipiell unabhängig von einem speziellen Betriebssystem und setzt nur das Vorhandensein einer JAVA-Umgebung vorraus.

8.1.1. XML

Bei XML (Extensible Markup Language) handelt es sich um eine Auszeichnungssprache, mit deren Hilfe sich Inhalte in Textform beschreiben und strukturieren lassen. Die Strukturierungsmechanismen beinhalten die Möglichkeit der Selbstbeschreibung, wodurch der Struktur grundsätzlich eine Bedeutung zugeordnet werden kann. Beispiel:

```
<name>
       <vorname>Peter</vorname>
       <nachname>Kaul</nachname>
</name>
```

In diesem Beispiel für XML werden zwei Textfragmente definiert ("Peter" und "Kaul"), die durch bestimmte Auszeichnungen (sog. "Tags" oder "Elemente". Im Folgenden wird der Begriff "Element" bevorzugt), nämlich <vorname> und <nachname> eingeschlossen werden. Der Text dieser Marken beschreibt die Bedeutung ihres Inhalts: Die Marke <vorname> lässt sich natürlich so interpretieren, dass es sich bei dem eingeschlossenen Text um einen Vornamen handelt. Beide Marken werden in einer

übergeordneten Marke <name> eingeschlossen, wodurch zum einen ihre Zusammengehörigkeit signalisiert wird und zum anderen deutlich wird, dass es sich hierbei um zwei Komponenten eines Namens handelt.

Elemente können noch mit zusätzlichen Attributen versehen werden, wie an folgendem Beispiel erläutert werden soll:

```
<warnung sprache="deutsch">Bitte sofort den Administrator verständigen</warnung>
```

Hierbei wird ein Text als eine "warnung" ausgezeichnet, der ein zusätzliches Attribut "sprache" zugeordnet ist. Über dieses Attribut wird beispielsweise erklärt, dass der zugehörige Text in deutscher Sprache verfasst ist.

XML wurde vom Konsortium "W3C"[12] als Standard verabschiedet und unter anderem zum Zweck entwickelt, ein universell einsetzbaren Mittels zur strukturierten Beschreibung und dem Austausch von Daten zu haben. Ein wesentlicher Aspekt bei dessen Entwicklung war es, im Gegensatz beispielsweise zu HTML, eine klare Trennung zwischen Inhalt und seiner Darstellung definieren. Mit anderen Worten: Wie und wo die in XML definierten Inhalte letztendlich präsentiert werden, kann unabhängig von XML entsprechend des jeweiligen Einsatzkontextes entschieden werden. Eine Strukturierung der Daten bzw. der Inhalte lässt sich durch verschachtelte Verwendung der Auszeichnungen beliebig fein vornehmen und unterstützt damit das Prinzip der Chunks.

Ein weiterer wichtiger Punkt ist, dass ein XML-Dokument maschinell nicht nur auf syntaktische Korrektheit überprüft werden kann, sondern in Grenzen auch auf semantische: Eine syntaktische Korrektheit ("well-formedness") von XML-Dokumenten bedeutet, dass die Auszeichnungen grundsätzlich richtig gesetzt und z.B. nicht überlappend definiert werden. Bei einer semantische Korrektheit ("Validität") muss zusätzlich gewährleistet sein, dass die Auszeichnungen nach einem bestimmten Schema geschachtelt sind. Dieses Schema wird in einer sog. DTD (Document Type Definition) festgelegt und soll an einem Beispiel erläutert werden:

```
<!ELEMENT name (vorname, nachname)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT nachname (#PCDATA)>
```

Damit wird definiert, dass ein Element <name> jeweils genau ein Element <vorname> und <nachname> in genau dieser Reihenfolge enthalten darf. Somit sind beispielsweise die Auszeichnungen

```
<name>Peter Kaul</name>
```

und

```
<name>
        <vorname>Peter</vorname>
        <vorname>H.</vorname>
        <nachname>Kaul</nachname>
</name>
```

entsprechend dieser DTD ungültig.

Für eine detailliertere Beschreibung dieser Sprache sei auf die Spezifikation von [Bray, Paoli & Sperberg-McQueen 98] oder auf die umfangreich vorhandene Fachliteratur verwiesen.

[12] http://www.w3c.org/ (24.10.2000)

Für die rechnergestützte Verarbeitung von XML werden sogenannte XML-Parser verwendet, die XML aus der Textform beispielsweise in eine objektorientierte Repräsentation überführen können. Eine solche Repräsentation stellt das "Document Object Modell" (DOM) dar (vgl. [Wood 00]), in dem die einzelnen Elemente eines XML-Dokumentes auf eine hierarchische Struktur von Objekten abgebildet werden. Auf diese Objektstruktur kann mittels Software ein manipulierender Zugriff stattfinden.
Einen solchen Parser stellt "Xerces" dar, der unter der "Apache Software License" frei verfügbar bereitgestellt[13] wurde. Dieser Parser stellt eine der integrierten Softwarekomponenten des SitePrototypers dar.

8.1.2. XSLT

XSLT (XSL Transformations) ist Bestandteil von XSL (Extensible Stylesheet Language) und definiert auf Basis von XML eine Sprache, mit der ein XML-Dokument in ein anderes XML-Dokument transformiert werden kann. Dazu werden in einem sogenannten Stylesheet eine Menge von XSL-Kommandos zusammengefasst, die jeweils eine solche spezielle Transformation beschreiben (s. [Clark 99]). Ein sogenannter XSLT-Prozessor sorgt für die Überführung eines Ausgangsdokuments in ein Zieldokument entsprechend der Definition im Stylesheet.

Im Kontext des SitePrototypers ist insbesondere die Eigenschaft von XSLT interessant, als Beschreibung der "Darstellung" einer Website zu dienen. In XSLT lässt sich nämlich die Überführung von XML nach HTML beschreiben, was einer Integration von Inhalt und Darstellung entspricht. In HTML lassen sich seinerseits Darstellungskomponenten wie Layout und Typographie beschreiben, die bekanntermaßen über einen Webbrowser angezeigt werden können.

Eine Implementation eines XSLT-Prozessors stellt der wiederum frei zur Verfügung gestellte "Xalan" dar[14], der ebenso wie der oben erwähnte "Xerces" als eine Komponente in den SitePrototyper integriert ist.

8.1.3. Servlets

Das Konzept der "Servlets" stellt ein Framework zur Entwicklung von webbasierten Applikationen auf Basis der Sprache JAVA dar[15]. Derartig entwickelte Applikationen werden mit einem Webserver gekoppelt und können "Anfragen" an den Webserver übernehmen und in Form spezieller "Antworten" reagieren (vgl. Kapitel 2.2). Im Gegensatz zu einfachen Webservern ergibt sich mit Servlets die Möglichkeit, die Ergebnisse einer Anfrage dynamisch zu erstellen, anstatt statische Dateien zurückzuliefern.

Implementationen der abstrakten Servlet-Spezifikation sind als "Servlet-Engines" bekannt und existieren in verschiedenen Ausprägungen. Die möglicherweise bekannteste dieser Engines ist "Tomcat"[16], die ebenfalls eine frei verfügbar Komponente des "SitePrototyper" darstellt. Innerhalb von "Tomcat" ist ein simpler Webserver integriert, der zusammen mit einem speziell für den "SitePrototyper" entwickelten Servlet als Bindeglied zwischen den Anfragen des Webbrowsers und den Ergebnisses des SitePrototypers dient.

---

[13] http://xml.apache.org/xerces-j/index.html (25.10.2000)
[14] http://xml.apache.org/xalan/index.html (25.10.2000)
[15] http://java.sun.com/products/servlet/index.html (25.10.2000)
[16] http://jakarta.apache.org/tomcat/index.html (25.10.2000)

8.2. Erstellung einer Website

Wie oben erwähnt, erfolgt die Erstellung und Bearbeitung der einzelnen Komponenten einer Website über das Dateisystem. Das bedeutet, dass für die Arbeit an einer Website sowohl Ordner als auch Dateien bestimmter Formate angelegt und modifiziert werden müssen. Der Grund, warum gerade das Dateisystem als Bearbeitungsgegenstand bestimmt wurde, liegt im Wesentlichen darin, dass dafür bereits eine große Menge an Werkzeugen existiert. Für alle dabei verwendeten Formate wie z.B. "XML" und "XSLT" existiert frei verfügbare Software, mit der sich derartige Dateien erstellen und bearbeiten lassen. Im Rahmen dieser Diplomarbeit war es damit eine bewusste Entscheidung allgemein auf diese Werkzeuge zu verweisen, um diese nicht neu erstellen zu müssen. Somit konnte das Augenmerk auf die eigentliche Funktionsweise des SitePrototypers gerichtet und in der begrenzten Zeit ein Prototyp entwickelt werden. Grundsätzlich besteht natürlich die Möglichkeit, eine Benutzungsoberfläche zu entwickeln, in der alle benötigten Werkzeuge integriert sind. Dies soll aber nur als Option für weitergehende Entwicklungen angesehen werden.

Wie die einzelnen Komponenten einer Website über das Dateisystem erstellt werden, ist Gegenstand der folgenden Abschnitte. Ziel ist es dabei, im Wesentlichen die Art und Weise zu beschreiben, in der die Ordner und Dateien angelegt und modifiziert werden müssen. Welche Werkzeuge dafür verwendet werden können, soll dabei nur als ein kleiner Teilaspekt behandelt werden.

8.2.1. Anlegen der Struktur

Page, Object und Container stellen jene Bausteine dar, über die sich die Struktur und die Inhalte einer Website definieren lassen. Hier soll beschrieben werden, wie das Erzeugen dieser Komponenten und der damit verbundenen Struktur mittels des SitePrototypers stattfinden kann. Dazu muss eine Abbildung definiert werden, mit der sich Dateien und Ordner des Dateisystems auf die oben beschriebenen Komponenten übertragen lassen.
Die grundsätzliche Schwierigkeit besteht darin, anhand der zwei unterscheidbaren Einheiten "Datei" und "Ordner" eine Überführung in die drei Einheiten "Page", "Object" und "Container" zu definieren.
Dies wird über eine spezielle Verwendung der Dateinamen gelöst, indem dort nämlich Zusatzinformationen untergebracht werden. Dazu soll ein Dateiname aus drei Teilen bestehend gesehen werden, nämlich "Name", "Position" und "Extension", die über folgendes Muster angeordnet werden: `name$position.extension`

Beispiel: Der Dateiname „personen$5.gif“ hat den Namen "personen", die Position "5", sowie die Extension "gif". Im Folgenden muss sorgfältig zwischen den Bezeichnungen "Dateiname" und "Name" unterschieden werden: Ersteres beschreibt alle drei genannten Komponenten mitsamt ihrer Trennsymbole, während letzteres nur die Komponente "Name" bezeichnet.

Während "Pages" als Dateien mit der Extension „xml“ angelegt werden, werden "Objects" über alle übrigen möglichen Extensions[17], wie z.B. „gif“ oder „jpeg“ identifiziert. Die Extensions der Objects bezeichnen dabei gleichzeitig deren "Mime-Type" (vgl. Kapitel 5.2) und orientieren sich dabei an den üblichen Konventionen zur Bildung von Dateinamen. So steht „jpeg“ zum Beispiel für eine Grafik des JPEG-Formats, während „css“ für eine Textdatei im Format der "Cascading Stylesheets" steht.

[17] abgesehen von der reservierten Extension ‘attributes’ (s.u.)

"Container" werden grundsätzlich als Ordner innerhalb des Dateisystems identifiziert. Der zusätzlichen Eigenschaft von "Pages", nämlich gleichzeitig als "Container" fungieren zu können, wird durch eine Besonderheit Rechnung getragen: Dazu kann ein Ordner angelegt werden, der denselben Namen wie die "Page" trägt. Dadurch werden die zwei Elemente des Dateisystems als ein Element innerhalb des Website-Modells aufgefasst.
Das Ablegen und Erzeugen von Dateien und Ordnern innerhalb bestimmter Ordner entspricht natürlich der hierarchischen Strukturierung von Pages und Objects.

Container und Objects können mit zusätzlichen Attributen versehen werden, indem spezielle Dateien mit der Extension "attributes" angelegt werden. Über den jeweils selben Namen des Containers und des Objects einerseits und der Attributes-Datei andererseits werden diese beiden Dateien bzw. Ordner miteinander assoziiert.

Der innerhalb eines Dateinamens definierten "Position" (s.o.) kommt eine weitere Bedeutung zu: Sie bestimmt die Reihenfolge der Elemente innerhalb eines Containers untereinander, so wie es in Kapitel 5.1 erläutert ist. Die Reihenfolge ergibt sich durch eine aufsteigende Sortierung der jeweils ganzzahligen "Positionen", wobei eine fortlaufende Nummerierung nicht unbedingt notwendig ist. Sofern ein Dateiname keine Positionskomponente beinhaltet, wird diese als unendlich groß angesehen. Auf diese Weise lassen sich Objects, Pages und Container gezielt anordnen, was mittels eines beliebigen Dateisystems nicht ohne weiteres möglich ist.

Dem in Kapitel 7.3.1 beschrieben Vorgang, mehrere Chunks zu einer Page zusammenzufassen ("Chunking"), wird beim SitePrototyper auf eine bestimmte Weise Rechnung getragen: Durch eine besondere Markierung einer Page-Datei, findet intern eine Verschmelzung mit allen untergeordneten Pages und Containern zu einer gemeinsamen Page statt. Dazu muss der entsprechende Ordner der Page-Datei mit der Extension „page“ ausgezeichnet werden. Eine Beschreibung der genauen Funktionsweise folgt in Kapitel 8.3.2.

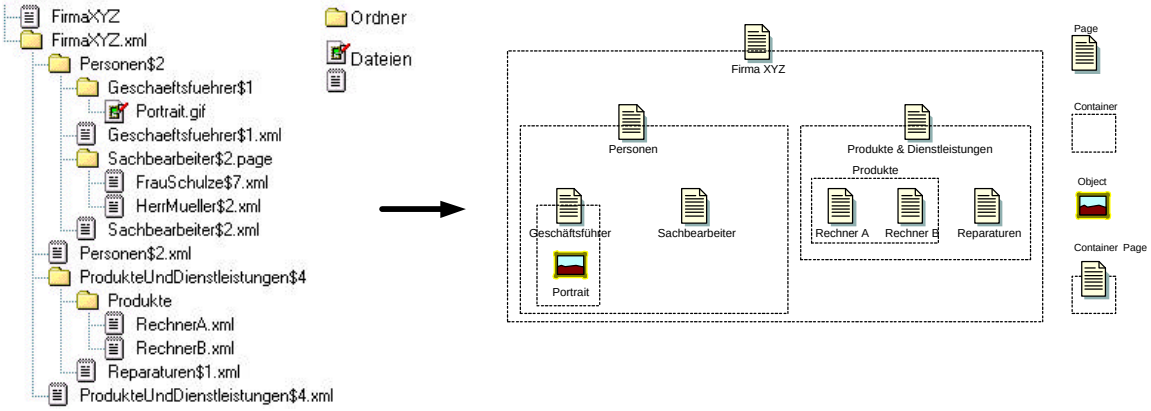

Abbildung 8.2: Die auf der linken Seite dargestellte Struktur aus Dateien und Ordnern entspricht der rechts dargestellten Struktur aus Pages, Objects und Containern.

Wie schon erwähnt, besteht jede Website aus mindestens einer Homepage, die gleichzeitig als Container für eventuell untergeordnete Pages, Objects und Container fungiert. Insofern wird eine Website durch das Anlegen einer entsprechenden Page-Datei sowie eventuell einem gleichnamigen Ordner definiert. Zusätzlich muss noch eine spezielle Konfigurationsdatei angelegt werden, deren Aufbau in Kapitel 8.2.5 beschrieben ist.

Anhand obiger Abbildung soll die hier beschriebene Beziehung zwischen den Dateien und der Struktur der Website abschließend illustriert werden.

8.2.2. Erstellen von Inhalten

Während im vorherigen Kapitel dargestellt wurde, wie Pages und Objects angelegt und mithilfe von Containers in eine Struktur gebracht werden, so soll es sich in diesem Kapitel darum drehen, wie die auf diese Weise erzeugten "leeren Hüllen" der Pages und Objects mit Inhalten gefüllt werden.

Dies soll für Objects nur sehr knapp beschrieben werden: Je nachdem, in welchem der vielen möglichen Formate (bzw. Mime-Types) das Object vorliegt, sind bestimmte Werkzeuge zur Bearbeitung geeignet. Durch die Vielzahl der Möglichkeiten von Objects soll hier nicht speziell auf die verschiedenen Werkzeuge eingegangen werden, sondern nur auf deren allgemeines Vorhandensein hingewiesen werden. Als Beispiel mag dabei "Photoshop" der Firma Adobe gelten, welches sich für die Bearbeitung von Bildern verschiedener Formate eignet.

Das Erstellen der übrigen Inhalte bzw. der Attribute erfolgt auf Basis von XML, für deren Erstellung grundsätzlich ein gewöhnlicher Texteditor ausreicht. Allerdings gibt es explizite XML-Editoren wie z.B. „XMLSpy“ von „Icon Information-Systems“[18], mit denen eine Bearbeitung um einiges komfortabler und benutzungsfreundlicher vonstatten gehen kann. Auch dies soll nicht weiter vertieft werden, sondern es soll stattdessen davon ausgegangen werden, dass der Leser ein grundsätzliches Wissen und Erfahrung bei der "manuellen" Erstellung und Bearbeitung von XML hat.

Grundsätzlich wird XML als Text beschrieben, der mittels Elementen und Attributen zusätzlich ausgezeichnet und somit strukturiert wird. Diesen Strukturierungsmitteln muss zumindest implizit eine Bedeutung zuordnet werden, damit sie an späterer Stelle entsprechend ihrer Bedeutung erkannt und verarbeitet werden können. Dies ist in diesem Kontext insbesondere für die spätere Überführung in eine Darstellung von Bedeutung. Um beispielsweise besonders „wichtige“ Inhalte zu markieren, können diese innerhalb des Elements <wichtig> eingebettet werden. Bei der Definition der Darstellung muss dieser Umstand bekannt sein, damit ein solcher Inhalt beispielsweise an prominenter Position platziert werden kann.

Im Rahmen dieser Diplomarbeit sind eine Menge von Elementen definiert worden, die als Rahmen zur Erstellung von Inhalten dienen sollen. Dabei wurde auf die in Kapitel 5.3 beschriebenen Bausteine Bezug genommen und eine entsprechende XML-Repräsentation entwickelt. Diese Elemente sollen hier beschrieben werden und sind gleichzeitig in einer DTD festgehalten, was eine Validierung der Inhalte in Bezug auf diese Elemente ermöglicht. Diese DTD findet sich als Anhang in Kapitel 10.1.

<text> und <structuredtext>

Mithilfe dieser Elemente kann jeweils ein Text ausgezeichnet werden. Ein einfacher Text, d.h. eine Ansammlung von Zeichen unseres Alphabets, lässt sich mittels <text> auszeichnen. Um allerdings einen strukturierten Text zu markieren, muss das Element <structuredtext> verwendet werden, innerhalb dessen sich mittels weiterer Elemente eine Struktur definieren lässt. Dabei wurden Anleihen an HTML gemacht, wo ein ähnliches Konzept verfolgt wird. Insofern finden sich die dort verwendeten

[18] http://www.xmlspy.com/ (25.10.2000)

Elemente hier zum Teil wieder: So werden mit den Elementen <h1>, <h2>, <h3> Überschriften bzw. Zwischenüberschriften ausgezeichnet und mit <p> ein Absatz gekennzeichnet. Über die Elemente <ol> und <ul> lassen sich Listen von Einträgen definieren, die entweder nummeriert oder unnummeriert dargestellt werden sollen. Für eine exakte Verwendung dieser Elemente soll auf die DTD verwiesen werden, während deren Bedeutung in der Spezifikation von HTML [Raggett, Le Hors & Jacobs 99] nachzulesen ist.

Mittels dieser Elemente kann dem Text eine Struktur gegeben werden, wobei die Einbindung von (assoziativen) Hyperlinks und Objects in den Text ebenfalls zulässig sein soll. Dies wird mittels der im Folgenden beschriebenen Elemente getätigt. Bei der Verwendung des eigentlichen Textes muss auf Besonderheiten von XML geachtet werden, d.h. dass beispielsweise die Zeichen „<“ und „>“ mittels der Entitäten „<“ und „>“ eingegeben werden.

Anhand von zwei Beispielen soll die Verwendung der Elemente demonstriert werden:

```
<text>Dies ist ein simpler Text</text>
```

und

```
<structuredtext>
       <h1>“Strukturierter Text“ -Demo</h1>
       <p>Dieser Text besteht aus zwei Absätzen. Dies ist der Erste ...</p>
       <p>... und dies der Zweite</p>
</structuredtext>
```

<url>

Dieses Element repräsentiert eine Referenz auf eine Page bzw. Object und stellt somit die Basis eines Hyperlinks bzw. eines eingebetteten Objects dar. Separat macht sie keinen großen Sinn, sondern dient ausschließlich als Baustein solcher Elemente.
Diese Referenz wird über das Format der URL [Berners-Lee et al. 94] beschrieben und lässt somit auch die Referenzierung externer Dokumente aus dem WWW zu. Interessanter ist hier allerdings die Referenzierung von Objects und Pages innerhalb der zu erstellenden Website. Diese werden mittels ihres Namens (s.o.) und ihrer Position innerhalb der Struktur identifiziert, was anhand von folgendem Beispiel demonstriert werden soll.

```
<url exists="true">/sites/firmaxyz/personen/geschaeftsfuehrung</url>
```

In Bezug auf obiges Beispiel (s. Abbildung 8.2) wird mit der hier definierten URL eine Referenz auf die "Page "geschaeftsfuehrer" definiert, die innerhalb der Container "firmaxyz" und "personen" abgelegt ist. Die Pfadkomponente „sites“ nimmt in Bezug auf den SitePrototyper eine Sonderrolle ein und wird an späterer Stelle erläutert. Dies findet in der detaillierten Beschreibung des Formats der URL in Kapitel 8.3.1 statt.

Das Attribut "exists" soll ausdrücken, ob die referenzierte Ressource auch wirklich existiert, oder ob es sich dabei um einen "broken link" handelt. Diese Frage lässt sich innerhalb einer mit dem SitePrototyper definierten Website beantworten, während dies bezogen auf das gesamte WWW nicht ohne weiteres möglich ist. Insofern erscheinen drei Möglichkeiten für dieses Attribut sinnvoll: "true", "false" und "unknown". Das Referenzieren von (noch) nicht existierenden Pages und Objects kann für den

Entwicklungsprozess durchaus sinnvoll sein: Somit können Hyperlinks definiert werden, deren Ziel erst später angelegt wird. Mit einer entsprechenden Darstellungskomponente können diese Hyperlink als inaktiv gekennzeichnet werden, so dass bei einer Evaluation zumindest ein Überblick über vorhandene Navigationsalternativen entstehen kann.

<label>
Bei <label> handelt es sich um die Repräsentation eines Labels (vgl. Kapitel 2.1), d.h. einem Bestandteil eines Hyperlinks. Wie schon erwähnt kann ein Label aus einem Text, einem Icon oder aus einem Zusammenspiel beider bestehen, was wiederum durch zwei Unterelemente, nämlich <text> und <icon>, repräsentiert wird. Während in <text> der Text des Labels spezifiziert wird, kann innerhalb von <icon> eine Referenz auf ein (Bild-)Object definiert werden. Wahlweise kann die Kombination beider Komponenten verwendet werden, oder aber auch nur jeweils eines der beiden. Dies kann beispielsweise so aussehen:

```
<label>
	<text>Übersicht</text>
	<icon><url>bilder/uebersicht_icon</url></icon>
</label>
```

oder auch so

```
<label>
	<text>Übersicht</text>
</label>
```

<hyperlink>
Das Element <hyperlink> beschreibt offensichtlich einen Hyperlink, der sich definitionsgemäß aus einem Label und einer Referenz zusammensetzt. Dazu kommen die oben beschrieben Elemente <label> und <url> zum Tragen, was beispielsweise folgendermaßen aussehen kann:

```
<hyperlink self=“false“>
	<url exists=”true”>/sites/firmaxyz</url>
	<label>
		<text>Homepage</text>
	</label>
</hyperlink>
```

Über das Attribute “self” kann angegeben werden, ob es sich bei dem Hyperlink um eine Selbstreferenz handelt, d.h. ob Ausgangspage und Zielpage übereinstimmen. Damit wird einer Konvention Rechnung getragen, nach der Selbstreferenzen auf einer HTML-Seite aus Gründen der Benutzbarkeit deaktiviert dargestellt werden sollen. Einer solchen Markierung kann durch eine entsprechende Darstellung Rechnung getragen werden.

<object>
Mit dem Element <object> wird ein einzubettendes Object deklariert, welches natürlich über eine Referenz <url> beschrieben wird. Beispiel:

```
<object type=“image“>
	<url exists=“true“>/sites/firmaxyz/logo</url>
</object>
```

Hier wird definiert, dass das Object „firmenlogo“ in die jeweilige Page eingebettet werden soll. Durch ein zusätzliches Attribut „type“ kann genauer spezifiziert werden,

um was für ein Object es sich handelt. In diesem Fall wurde „image“ angegeben, um zu signalisieren, dass es sich um ein Bild handelt.

Die bis hierhin erläuterten Elemente beziehen sich prinzipiell auf die in Kapitel 5.3.1 beschriebenen „Datentypen“ in Ausprägung von „Text“, „Object“ und „Hyperlink“. Demgegenüber soll im Folgenden auf die „Inhaltstypen“ (vgl. Kapitel 5.3.2) eingegangen werden, für die jeweils eigene Elemente definiert wurden:

<navigation>
Mithilfe von <navigation> lassen sich eine Menge von Hyperlinks auszeichnen und strukturieren. Das bedeutet, dass ein solches Element sowohl <hyperlink> enthalten darf, als auch wiederum <navigation>. Mit letzterem lassen sich Untergruppen von Hyperlinks bilden, was beispielsweise für eine Unterscheidung verschiedener Navigationselemente (vgl. Kapitel 7.4.1) verwendet werden kann.
Über ein zusätzliches Attribut „name“ lässt sich dem jeweiligen Element <navigation> ein Name zuordnen, mit dessen Hilfe sich die Elemente auch individuell unterscheiden lassen können. Alternativ ist auch ein Label unterhalb von <navigation> zulässig, um damit die Bedeutung der jeweiligen Hyperlinks zu beschreiben. Beispiel:

```
<navigation name=”personen”>
  <hyperlink self=”false”>
    <url exists=”true”>/sites/firmaxyz/personen/geschaeftsfuehrerung</url>
    <label><text>Geschäftsführerung</text></label>
  </hyperlink>

  <navigation>
        <label><text>Sachbearbeiter</text></label>

        <hyperlink self=”false”>
               <url exists=”true”>/sites/firmaxyz/personen/sachbearbeiter/mueller</url>
               <label><text>Herr Müller</text></label>
        </hyperlink>

        <hyperlink self=”false”>
               <url exists=”true”>/sites/firmaxyz/personen/sachbearbeiter/schulze</url>
               <label><text>Frau Schulze</text></label>
        </hyperlink>
    </navigation>
</navigation>
```

<attributes>
Wie der Begriff „attributes“ schon vermuten lässt, können innerhalb dieses Elements eine Menge von Attributen zusammengefasst werden. So kann beispielsweise innerhalb einer Pagedatei das zur Page zugehörige Label abgelegt werden, wie es schon in Kapitel 7.5.3 beschrieben wurde. Beispiel:

```
<attributes>
       <label><text>Homepage</text></label>
</attributes>
```

<macro>
Bei dem Element <macro> handelt es sich nicht direkt um ein inhaltsbeschreibendes Element, sondern es stellt vielmehr einen Stellvertreter für einen beliebigen anderen Inhalt dar. Ein solcher Inhalt muss nicht zwangsläufig manuell festgelegt werden, sondern er kann mittels eines Algorithmus automatisch erzeugt werden. Damit lässt sich beispielsweise die in Kapitel 7.4.3 beschriebene algorithmische Definition eines Navigationssystems erreichen. Eine genauere Verwendung und Funktion dieses Elements wird in Abschnitt 8.2.4 bzw. 8.3.2 geschildert, so dass hier nur folgendes festgestellt werden soll: Mittels eines Attributes „class“ wird der Algorithmus in Form

einer JAVA-Klasse spezifiziert, der durch den SitePrototyper aufgerufen und ausgeführt wird. Verwendet wird das Element beispielsweise so:

```
<macro class=“de.peterkaul.siteprototyper.helpers.SitemapMacro“/>
```

Zusammenspiel

Mithilfe der hier beschriebenen Elemente ist ein Rahmen definiert worden, mit dem sich die inhaltlichen Elemente gemäß Kapitel 5.3 mittels XML ausdrücken lassen. Auf dieser Basis können die Page- bzw. Attribut-Dateien (vgl. Kapitel 8.2.1) grundsätzlich mit Inhalten gefüllt werden, wobei als Konvention zusätzlich folgender Rahmen gelten soll:

Attributes-Dateien sollen als Wurzel-Element das oben genannte Element <attributes> tragen, in dem die einzelnen Attribute eingebettet sind. Pages hingegen können über ein beliebiges Wurzel-Element beschrieben werden, dem jeweils genau ein Element <attributes> und ein Element <navigation> untergeordnet sein soll. Zusätzlich sollen beliebige Elemente als eigentliche Informationselemente (vgl. Kapitel 5.3.2) zulässig sein. Dies soll exemplarisch anhand von folgendem Beispiel demonstriert werden:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE generic SYSTEM "http://localhost:8080/dtds/siteprototyper.dtd">
<generic>
	<attributes>
		<label><text>Homepage</text></label>
	</attributes>

	<navigation name=“primary“>
		<hyperlink self=”false”>
			<url exists=”true”>produkte</url>
			<label><text>Unsere Produkte</text></label>
		</hyperlink>

		<hyperlink self=”false”>
			<url exists=”true”>mitarbeiter</url>
			<label><text>Unsere Mitarbeiter</text></label>
		</hyperlink>
	</navigation>

	<content>
		<structuredtext>
			<h1>Herzlich Willkommen</h1>
			<p>Wir heissen sie willkommen auf der Website der Firma XYZ</p>
		</structuredtext>
	</content>
</generic>
```

Dieses Beispiel einer Page beginnt mit einem Wurzelelement <generic>, dem jeweils ein Element <attributes> und <navigation> untergeordnet sind. Der eigentliche Inhalt wird innerhalb von <content> in Form eines <structuredtext> beschrieben. Dies soll eine allgemeine Lösung zur Definition einer Page darstellen und ist als folgende Definition in der DTD enthalten:

```
<!ELEMENT generic (attributes,navigation,content)>
<!ELEMENT content (structuredtext)+>
```

Jede neu angelegte Page kann somit zunächst über diesen allgemeinen Dokumenttyp „generic“ beschrieben werden und im Bedarfsfall mittels eines spezielleren Typs verfeinert bzw. angepasst werden. So wäre es beispielsweise denkbar, die DTD zu

erweitern und einen neuen Dokumenttyp „homepage“ anzulegen, der sich über folgende Definition beschreiben lässt:

```
<!ELEMENT homepage (navigation,attributes,headline,intro)>
<!ELEMENT headline (text)>
<!ELEMENT intro (structuredtext)>
```

In diesem Dokumenttyp werden zwei neue Elemente <headline> und <intro> definiert, die ihrerseits jeweils ein untergeordnetes Element <text> bzw. <structuredtext> verlangen.

8.2.3. Darstellungsdefinition

Die Darstellung der in den Pages definierten Inhalte wird mithilfe von XSLT (s.o.) realisiert, d.h. dass innerhalb eines sog. „Stylesheets“ Transformationen beschrieben werden, nach denen die Inhalte von XML in eine HTML-Repräsentation überführt werden. Dies geschieht mithilfe einer Menge von Templates, wobei jedes Template die Darstellung eines bestimmten Elementes in einem bestimmten Kontext beschreibt. Jedes Template beinhaltet eine Menge von HTML-Elementen, die dann entsprechend der zusätzlichen XSLT-Vorschriften zu einem Gesamt-HTML-Dokument konstruiert werden können.

Aufgrund der Komplexität von XSLT soll hier nicht weiter auf die konkrete Erstellung von Stylesheetdokumenten eingegangen werden, sondern nur auf die Spezifikation von [Clark 99] bzw. entsprechende Fachliteratur hingewiesen werden. Die Erstellung von Stylesheets wird inzwischen von einer Reihe von Werkzeugen unterstützt, von denen exemplarisch „Stylus“ der Firma „Excelon“[19] genannt werden soll.

Ebenso wenig soll hier auf die genaue Möglichkeiten eingegangen werden, mit denen sich eine Darstellung über HTML beschreiben lässt, d.h. wie sich beispielsweise Farbe oder Layout in HTML ausdrücken lassen. Auch zu diesem Thema existiert eine große Menge an Literatur, sowohl in Buchform, als auch als Ressourcen im WWW, so dass nur allgemein auf derartige Dokumente verwiesen werden soll.

Prinzipiell muss zur vollständigen Definition der Darstellung einer Website für jeden Dokumenttyp ein separates Template erstellt werden, womit dann für jedes Dokument dieses Typs die Darstellung beschrieben wird. Aufgrund der Modularisierbarkeit von XSLT muss die Darstellung einzelner untergeordneter Elemente, wie z.B. den Hyperlinks, nicht mehrfach beschrieben werden, sondern kann zentral festgelegt werden. Für die im vorherigen Kapitel beschriebenen Basiselemente ist im Rahmen dieser Arbeit ein Stylesheet erstellt worden, dass deren Überführung nach HTML beschreibt und somit für eine Basisdarstellung sorgt. Dabei wird davon ausgegangen, dass die einzelnen Pages entsprechend obiger Konvention erstellt worden sind. Auch wenn die resultierende Darstellung visuell nicht sehr ansprechend ist, so kann sie dennoch als Ausgangspunkt für eine Adaption an konkrete Umstände dienen und ermöglicht ein Prototyping der Website zu einem frühen Zeitpunkt.
Das Basis-Stylesheet soll hier nicht weiter erläutert werden, sondern findet sich in Kapitel 10.2. Eine Erweiterung kann stattfinden, indem ein neues Stylesheet erstellt wird, in dem das Basis-Stylesheet „importiert“ wird. Dies soll an folgendem Beispiel erläutert werden:

[19] http://www.exceloncorp.com/products/excelon_stylus.html (10.10.2000)

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" indent="yes"/>
<xsl:include href="http://localhost:8080/styles/siteprototyper.xsl"/>
	<xsl:template match="intro">
		<TABLE BGCOLOR="Red">
		<TR>
			<TD>
			<xsl:apply-templates select=”structuredtext”/>
			</TD>
		</TR>
		</TABLE>
	</xsl:template>
</xsl:stylesheet>
```

In diesem Beispiel wird das Basis-Stylesheet „siteprototyper.xsl“ importiert und durch ein Template zur Darstellung des Elementes <intro> erweitert. Dazu wird der Inhalt des Elementes „structuredtext“, dessen Darstellung schon im Basis-Stylesheet definiert ist, in eine (HTML-)Tabelle mit rotem Hintergrund eingebettet.

Dekorative Elemente, bei denen es sich beispielsweise um Bilder handeln kann, können nicht direkt über XSLT bzw. HTML definiert werden, sondern müssen als separate Dateien angelegt und vom Stylesheet aus referenziert werden. Es handelt sich dabei aus der HTML-Sicht natürlich um denselben Mechanismus wie bei der Einbettung von Objects, wobei hier aber die logische Trennung von Inhalt und Darstellung für eine entsprechende unterschiedliche Behandlung sorgt. Wie ein Einbindung der dekorativen Elemente getätigt werden kann, soll folgendes Beispiel demonstrieren:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:include href="http://localhost:8080/styles/siteprototyper.xsl"/>
<xsl:output method="html" indent="yes"/>

<xsl:template match="/">
<HTML>
	<HEAD>
		<TITLE>Firma XYZ</TITLE>
	</HEAD>

	<BODY BACKGROUND="http://localhost:8080/styles/background.gif">
		<xsl:apply-templates/>
	</BODY>
</HTML>
</xsl:template>
```

Ein dekoratives Element in Form eines Hintergrundbildes wird innerhalb des Elements <BODY> als Referenz auf die Datei „background.gif“ eingebunden. Diese Datei befindet sich ebenso wie das Stylesheet in einem Ordner „styles“, was als weitere Konvention angesehen werden soll.

8.2.4. Erstellen von Macros

Wie schon erwähnt, stellt ein „Macro“ einen Mechanismus innerhalb des SitePrototypers dar, mit dem sich Inhalte automatisch generieren lassen. Dazu wird über einen Platzhalter innerhalb einer Page-Datei ein Algorithmus spezifiziert, dessen Ausführung einen Inhalt in Form eines Elementes zum Ergebnis hat. Auf diese Weise lassen sich Inhalte, die in wiederkehrenden Mustern auftreten, mittels des Algorithmus spezifizieren und stellen damit ein Entwurfsmuster gemäß Kapitel 4.7 dar.
Die Verwendung eines Macros soll zunächst anhand von folgendem Beispiel erläutert werden:

```
<macro class=“de.peterkaul.siteprototyper.helpers.ChildrenNavigationMacro“/>
```

Dieses Macro bezeichnet eine JAVA-Klasse “ChildrenNavigationMacro” (innerhalb eines Paketes „de.peterkaul.siteprototyper.helpers“), deren Verwendung beispielsweise folgendes Element <navigation> als Ergebnis liefert:

```
<navigation type=“children“>
	<hyperlink self=”false”>
		<label><text>Personen</text></label>
		<url exists=”true”>/sites/firmaxyz/personen</url>
	</hyperlink>

	<hyperlink self=”false”>
		<label><text>Produkte</text></label>
		<url exists=”true”>/sites/firmaxyz/produkte</url>
	</hyperlink>
</navigation>
```

Dem SitePrototyper sind einige Macros beigefügt (vgl. Kapitel 8.4.1), die als Lösungen von Standardprobleme bei der Entwicklung von Websites zu verstehen sind. Allerdings besteht auch für den Entwickler einer speziellen Website die Möglichkeit, anwendungsspezifische Macros zu erstellen. Dies geschieht mittels der Programmiersprache JAVA und soll im Folgenden geschildert werden:

Ein Macro ist abstrakt über ein Interface definiert, welches vom Entwickler in Form einer Klasse implementiert werden muss, um so dessen Algorithmus festzulegen. Seitens des SitePrototypers wird diese Klasse zu gegebenem Zeitpunkt instanziiert und eine entsprechende Methode aufgerufen. Dies entspricht der Ausführung des Algorithmus, wodurch ein Ergebnis in Form eines Elementes erzeugt wird.
Der jeweiligen Instanz werden diverse Parameter übergeben, die den aktuellen Kontext, z.B. in Form der aktuellen Page, spezifizieren und somit eine Generierung in deren Abhängigkeit zulassen. Das Interface und dessen kontextabhängige Parameter sind wie folgt definiert:

```
public interface Macro
{
	public abstract org.w3c.dom.Node execute
		(
			org.w3c.dom.Element element,
			org.w3c.dom.Document document,
			de.peterkaul.siteprototyper.Page page,
			de.peterkaul.siteprototyper.Site site
		);
}
```

Bei der Implementierung dieses Interfaces muss die Funktionalität des Macros also innerhalb der Methode „execute“ beschrieben werden, der die Parameter element, document, page und site übergeben werden. Die ersten beiden Parameter beziehen sich auf ein „Application Program Interface“ (API) des „Document Object Modell“ (DOM) (vgl. [Wood 00]), in dem ein XML-Dokument über JAVA-Objekte bereitgestellt wird. Über die nächsten beiden Parameter werden ebenfalls Objekte übergeben, die jeweils Bestandteil der API des SitePrototypers sind. In dieser API wird die Website mittels Objekten repräsentiert, so dass mittels JAVA auf deren Komponenten wie z.B. Pages, Objects und Struktur zugegriffen werden kann.

Im Einzelnen haben die Parameter folgende Bedeutung:

document
Der Parameter „document“ repräsentiert jeweils den Inhalt jener Page, in der das aufgerufenen Macro definiert ist. Es ist vom Typ org.w3c.dom.Document und stellt somit eine komplette DOM-Repräsentation des in XML definierten Inhalts dar.

element
Hierbei handelt es sich um das aufrufende Element <macro>, das innerhalb obigen Dokuments eingebettet ist.

page
Die Struktureinheit „Page“, in der das aufrufende Macro definiert ist, wird durch dieses Objekt repräsentiert. Über die Methoden dieser Page kann auf die komplette Struktur der Website und deren Komponenten zugegriffen werden.

site
Dieses Objekt repräsentiert die komplette Website, in der obige Page als Bestandteil eingebettet ist.

Jedes dieser Objekte stellt eine Menge von Methoden bereit, die für die automatische Generierung eines Inhaltes relevant sein können. So kann beispielsweise mittels des Objekts „page“ auf die Struktur der Website zugegriffen werden. Deren Methode „getChildren()“ liefert alle Nachkommen (in Form von Page, Object und Container), während „getParent()“ Zugriff auf den jeweiligen Vorgänger liefert. Die wesentlichen Klassen der SitePrototyper-API sind im Kapitel 10.3 beigefügt und sollen hier nicht weiter erläutert werden.

Weitere Parameter können dem Macro mittels eingebetteter Elemente übergeben werden, wie es folgendermaßen demonstriert wird:

```
<macro class=”TestMacro”>
	<param>Dies ist ein Parameter</param>
</macro>
```

Das bedeutet, dass das Element <param> mitsamt seinem Inhalt über das Object „element“ (s.o.) zugreifbar ist und somit als Parameter dienen kann.

Jedes Macro muss als Ergebnis seiner Berechnung ein Objekt vom Typ org.w3c.dom.Node zurückliefern, mit dem u.a. XML-Elemente repräsentiert werden. Vom SitePrototyper wird das Element <macro> durch dieses Ergebnis ersetzt, wodurch der Inhalt der jeweiligen Page intern modifiziert wird.

Als Beispiel für die Erstellung eines Macros soll folgende Klasse dienen:

```
import org.w3c.dom.*;
import de.peterkaul.siteprototyper.*;
public class TestMacro implements Macro
{
public TestMacro()
{
	super();
}

public Node execute(Element element, Document document, Page page, Site site)
{
	org.w3c.dom.Node	result =	null;
	try
	{
		result =	document.createElement("parentname");
		Container	parent =	page.getParent();
		org.w3c.dom.Text	text	=	document.createTextNode(parent.getName());
		result.appendChild(text);
	}
	catch(Exception e)
	{
		result =	document.createElement("error");
		org.w3c.dom.Text	message = document.createTextNode(e.getMessage());
```

```
                result.appendChild(message);
        }
        return result;
}
}
```

In diesem Beispiel wird ein Macro mit dem Namen „TestMacro“ definiert, das als Ergebnis der Berechnung jeweils den Namen des übergeordneten Containers in der Form <parentname>[name]</parentname> zurückliefert. Sofern bei der Ausführung ein Fehler auftritt, wird stattdessen eine Fehlermeldung erzeugt, die in das Element <error> eingebettet wird.

Das Abfangen und Behandlung von Fehlern in Form der „Exceptions“ liegt in der Verantwortlichkeit des Programmierers und soll über folgende Konvention geregelt werden: Ein nicht behandelbarer Fehler soll in der Rückgabe eines Elementes <error> resultieren, in das, möglichst im Klartext, weitere Fehlerbeschreibungen eingebettet werden.

8.2.5. Konfiguration einer Website

Im vorherigen Kapiteln ist beschrieben worden, auf welche Weise sich eine Website mittels verschiedener Dateien und Formate erstellen bzw. definieren lässt. Damit das System des SitePrototypers von der Existenz dieser Dateien in Kenntnis gesetzt wird, ist es notwendig, dass diese einerseits an einer bestimmten Stelle im Dateisystem abgelegt werden und andererseits für jede Website eine explizite Konfigurationsdatei angelegt wird.
Bei der Installation des SitePrototyper wird implizit ein sog. „Projektordner“ festgelegt, der als Basis für alle Dateien der Website dienen soll. Dort befinden sich mehrere Unterordner, nämlich `sites/`, `styles/`, `dtds/` und `macros/`, in denen die einzelnen Dateien abgelegt werden. Die Stylesheets und die dekorativen Elemente werden dabei in `styles/`platziert, während dies bei den DTDs natürlich im Ordner `dtds/` der Fall ist. Analog werden die JAVA-Klassen bzw. die Inhalts- und Strukturdateien in den Ordnern `macros/` sowie `sites/` abgelegt. Der Macro-Ordner ist dabei so zu verstehen, dass er Teil des JAVA-Klassenpfades ist und somit eine entsprechende Behandlung bedingt. Direkt im Ordner `sites/` hingegen werden die Homepages aller Websites platziert, während alle weiteren Dateien in entsprechenden Unterordnern liegen müssen (vgl. Kapitel 8.2.1).

Die Konfigurationsdateien werden ebenfalls direkt in `sites/` abgelegt und müssen entsprechend folgender Ausführungen bearbeitet werden. Dort werden nach dem Prinzip „schlüssel = wert“ nacheinander alle Parameter festgelegt, die für die jeweilige Website von Relevanz sind. Es handelt sich dabei um sog. „Properties“-Dateien, deren Format aus dem Umfeld von JAVA (vgl. [Middendorf, Singer & Strobel 96, S. 754f]) als bekannt vorrausgesetzt wird.
Die Konfigurationsdatei wird nach dem Muster [name].properties benannt, wobei [name] zur weiteren Identifikation der jeweiligen Website dient. Innerhalb dieser Dateien sind folgende Einträge von Interesse:

site.homepage
Hiermit wird die Datei festgelegt, die die Homepage der entsprechenden Website darstellt. Dies kann über einen Verzeichnispfad relativ zum Installationsverzeichnis des SitePrototypers angegeben werden. Über die Homepage wird die Position der übrigen Dateien der Website implizit mitbestimmt (vgl. Kapitel 8.2.1).
Beispiel: `site.homepage = ./projects/sites/FirmaXYZ.xml`

stylesheet.url
Über diesen Parameter wird das zu verwendende Stylesheet über eine URL festgelegt. Über den im SitePrototyper integrierten Webserver kann auch auf ein lokal abgelegtes Stylesheets zugegriffen werden, indem dieser Server entsprechend spezifiziert wird.
Beispiel: `stylesheet.url = http://localhost:8080/styles/siteprototyper.xsl`

stylesheet.mimetype
Eine über das Stylesheet definierte Transformation des Inhalts wird in der Regel HTML zum Ergebnis haben. Grundsätzlich kann aber auch ein anderes Format spezifiziert werden, wie z.B. XML oder WML. In diesem Zusammenhang ist prinzipiell nur wichtig, dass ein entsprechender Browser zur Verfügung steht. Das Format wird mittels des „MIME-Type“ (vgl. [Freed & Borenstein 96]) beschrieben und lautet z.B. für HTML text/html.
Beispiel: `stylesheet.mimetype = text/html`

parsers.validate
Über diesen Parameter kann festgelegt werden, ob die XML-Dokumente der Website grundsätzlich gegen eine DTD validiert werden sollen oder nicht. Entsprechend lauten die beiden Konfigurationsmöglichkeiten „true“ und „false“.
Beispiel: `parser.validate =  true`

Mittels der Konfigurationsdateien ist es u.a. möglich mehrere Websites zu definieren, die zwar ein und dieselben Inhalte haben, aber unterschiedliche Darstellungen. Dazu reicht es aus, mehrere Konfigurationsdateien mit jeweils derselben Homepage aber unterschiedlichen Stylesheets anzulegen. Sinnvoll ist dies, wenn z.B. mehrere Darstellungsentwürfe parallel evaluiert werden sollen.

8.3. Funktionsweise

Im vorherigen Kapitel ist geschildert worden, auf welche Weise eine Website innerhalb vom SitePrototyper erstellt werden kann. In diesem Kapitel soll nun beschrieben werden, wie auf diese Daten sinnvoll zugegriffen kann bzw. wie sich die Funktionalität des SitePrototypers darstellt.

Grundsätzliches Ziel ist es, dass seitens des SitePrototypers alle Komponenten einer Website, wie Inhalt, Darstellung, Struktur usw., zu einer kompletten und navigierbaren Struktur aus HTML-Seiten zusammengefügt werden. Es soll möglich sein, mittels eines Webbrowsers auf diese Seiten zuzugreifen, um somit den aktuellen Stand der Entwicklung evaluieren zu können.

8.3.1. Zugriff und Logging

Über den im SitePrototyper integrierten Webserver wird die Website bereitgestellt und lässt sich mittels eines Webbrowsers abrufen und navigieren. Je nachdem, wie der SitePrototyper installiert bzw. konfiguriert wurde, wird sich die URL unterscheiden, unter der die Website zu erreichen ist. Der Einfachheit halber soll hier davon ausgegangen werden, dass sich der Server als „localhost“ auf dem Port „8080“ ansprechen lässt, so dass die Basis-URL des SitePrototypers „http://localhost:8080/“ lautet. Mittels einer Pfadkomponente „sites/[name]“ lässt sich dann auf die Homepage der jeweiligen Website zugreifen, d.h. dass z.B.

„http://localhost:8080/sites/firmaxyz“ als URL für die Homepage der Website „firmaxyz“ dient. Mittels zusätzlicher Pfadkomponenten lässt sich jede einzelne Page und jedes Object eindeutig ansprechen. Grundsätzlich wird eine solche Ressource über einen Pfad, von der Homepage bis zur eigentlichen Ressource, identifiziert. Dabei werden die Namen der jeweiligen Container getrennt durch das Symbol „/“ aneinandergefügt. Beispiel: Über die URL

```
http://localhost:8080/sites/firmaxyz/personen/sachbearbeiter/mueller
```

lässt sich die Page „mueller“ ansprechen, die sich im Container „sachbearbeiter“ und diese wiederum im Container „personen“ befindet.

Der Zugriff auf die Pages und Objects muss aber für gewöhnlich nicht durch das Eingeben der jeweiligen URL in den Webbrowser erfolgen, sondern ergibt sich automatisch als Bestandteil des Navigationsvorganges. Das bedeutet, dass die Ressourcen über die Verwendung der Hyperlinks automatisch vom Webbrowser angefordert werden. Einzig der erstmalige Zugriff, z.B. auf die Homepage, muss manuell erfolgen.

Jeder Zugriff auf eine beliebige Ressource einer Website wird in einem Logfile im „Common Logfile Format“ vermerkt, wodurch grundsätzlich eine spätere Analyse mittels geeigneter Werkzeuge ermöglicht wird (vgl. Kapitel 3.4.3). Dieses Logfile wird als Textdatei unter dem Namen „access.log“ im Ordner „logs“ des SitePrototypers angelegt und bei jedem Zugriff um eine neue Zeile gemäß des obigen Formates erweitert.

8.3.2. Verarbeitung der Daten

Die vom Entwickler erstellten Daten repräsentieren ein Website, die intern vom SitePrototyper verarbeitet werden muss, bevor eine einzelne Page aufgrund einer Anfrage als HTML zurückgegeben werden kann. Neben dem Parsing der XML-Dateien und der Darstellungstransformation mittels XSLT, die mithilfe der integrierten Komponenten „Xerces“ und „Xalan“ (s. Kapitel 8.1) vorgenommen wird, sind hier insbesondere die Ausführung der Macros und die Verschmelzung der Inhalte von Interesse. Dabei handelt es sich um Funktionalitäten, die speziell in dieser Diplomarbeit entwickelt wurden und somit an dieser Stelle genauer geschildert werden sollen:

Ausführung der Macros
Die Ausführung der Macros (vgl. Kapitel 8.2.4) bedeutet, dass vor der Darstellungstransformation einer Page alle ihre Elemente <macro> durch entsprechend generierte Inhalte ersetzt werden. Dazu wird bei jedem einzelnen dieser Elemente die entsprechend spezifizierte Klasse instanziiert und deren Methode „execute“ aufgerufen. Dabei wird der augenblickliche Kontext in Form verschiedener Parameter übergeben, so dass dieser bei der Berechnung einbezogen werden kann. Das dabei resultierende Element wird dann anstelle des Elements <macro> in die augenblickliche interne Repräsentation der Page eingefügt.

Verschmelzung von Inhalten
Über die Funktion der „Verschmelzung“ wird der Vorgang des „Chunking“ (vgl. Kapitel 8.2.1) reflektiert, in dem es darum geht, mehrere Einzelinhalte zu einer Page zusammenzufassen. Der Hintergedanke der im Folgenden beschriebenen Funktionalität ist der, dass innerhalb des Entwurfsvorganges zu Beginn nicht unbedingt endgültige Klarheit darüber herrscht, welche Inhalte zu einer Page zusammengefasst werden sollen. Insofern ist es durchaus möglich, dass, nachdem Inhalte als separate Page-Dateien definiert wurden, diese zu einer einzelnen zusammengefasst werden sollen. Anstatt dies mittels der manuellen Bearbeitung und Löschung dieser Dateien

tätigen zu lassen, was durchaus einen kritischen Zeitfaktor darstellt, soll dies automatisiert vom SitePrototyper geschehen.

Hier soll nun erläutert werden, wie der Mechanismus der Verschmelzung funktioniert und wie der dadurch entstehende Inhalt der Page aussieht. Zum allgemeinen Verständnis sei noch einmal klargestellt, dass diese Verschmelzung im Prinzip nichts an dem Inhalt und der Struktur ändert, sondern diese im Prinzip nur durch eine andere Granularität bereitstellt. Es findet also eine Überführung der hierarchischen Struktur aus Pages zu einer hierarchischen Struktur aus Chunks statt. In Bezug auf den SitePrototyper lässt sich dies auch so formulieren: Eine Struktur aus Dateien und Ordnern wird in eine Struktur in XML überführt.

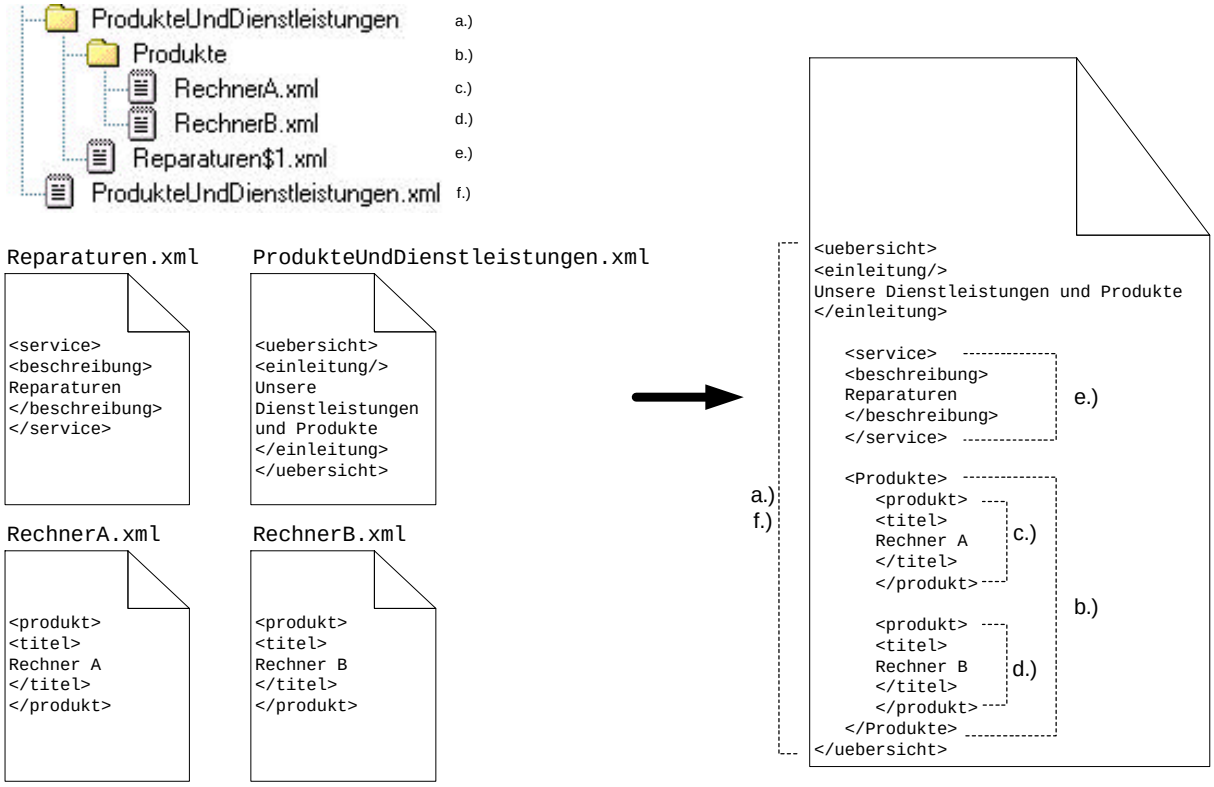

Abbildung 8.3: Eine Dateistruktur (links oben), bestehend aus zwei Ordnern und vier Dateien, wird zu einem einzelnen Page-Dokument (rechts) verschmolzen. Die Inhalte der einzelnen Dateien (links unten) und ein Element „Produkte“ (analog zum Container „Produkte“) werden dabei zusammengeführt.

Dies findet mittels eines rekursiven Verfahrens statt, bei dem beginnend mit der Ausgangspage die Inhalte der nachkommenden Pages und Container jeweils in diese Page eingefügt werden. Während der Inhalt der nachkommenden Pages über deren Wurzelelement direkt eingefügt wird, geschieht dies bei gewöhnlichen Containern durch Einfügen eines Elementes mit dem Namen des Containers. Eingefügt werden diese Elemente direkt unter das Wurzelelement ihres jeweils übergeordneten Containers. Als Ergebnis liegt dann eine Struktur aus ineinandergeschachtelten Elementen vor, wie es anhand von obiger Abbildung erläutert werden soll.

Bei der hier beschriebenen Funktionalität wird praktisch ein Teilbaum eines hierarchischen Informationssystems zu einem einzigen Knoten zusammengefasst. Dies beschreibt allerdings nur einen einzelnen Aspekt des „Chunking“: Einerseits wird weder das Aufspalten einer Datei in mehrere Unterdateien unterstützt, andererseits

stellt auch das beliebige Zusammenfassen von Chunks eine Hürde dar. Insofern ist die hier geschilderte Funktion als ein erster Entwurfsschritt der grundsätzlichen Möglichkeiten zu sehen.

8.3.3. Ablauf

Zum besseren Verständnis der Funktionsweise, soll der zeitliche Ablauf innerhalb des SitePrototypers geschildert werden, wie er sich ab der Anfrage seitens des Webbrowsers bis zur Antwort in Form eines HTML-Dokuments abspielt.

Die Anfrage an den SitePrototyper über den Webserver geschieht mittels einer URL seitens des Webbrowsers. Sofern die Anfrage eine Website spezifiziert, wird sie an ein entsprechendes Servlet übergeben. Dieses Servlet kontrolliert zunächst, ob die gewünschte Website existiert, indem nach einer entsprechenden Konfigurationsdatei gesucht wird, und versucht dann ggf. die angeforderte Page oder das Objekt mittels des Pfades und des Namens zu finden. Sollte einer dieser Schritte missglücken, wird dies als Fehler gemeldet.

Sollte sich die Anfrage auf ein Object beziehen, wird dessen Mime-Type mittels der internen Extension bestimmt, und dessen Binärdaten werden zurückgeliefert. Andernfalls handelt es sich um eine Page, so dass als nächster Schritt das in der Konfiguration definierte Stylesheet mittels Parsing vom XSLT-Prozessor auf syntaktische Richtigkeit geprüft wird und eventuelle Fehler gemeldet werden.

Sollte es bis hierhin keine Fehler gegeben haben, wird der Inhalt der Page in eine interne Repräsentation überführt, indem die Page-Datei geparsed wird und danach alle Macros ausgeführt werden. Sofern sich die Page aus mehreren Dateien zusammensetzt, wird jede einzelne Datei geparsed und integriert. Beim Auftreten eines Fehlers (bezüglich der Syntax oder der Validität) während des Parsings werden diese mittels speziell integrierter Elemente gekennzeichnet. Am Ende dieser Prozedur liegt der Inhalt der Page komplett vor und kann in seine Darstellung überführt werden.

Dies geschieht in einem letzten Schritt, in dem das Stylesheet auf den Inhalt angewandt wird und das Resultat an den Webbrowser zurückgeliefert wird. Da auch die Ausführung des Stylesheets Fehler bedingen kann, werden diese ggf. Anstelle des eigentlichen Resultats zurückgeliefert.

8.3.4. Fehler und Fehlersuche

Bei der Arbeit mit dem SitePrototyper können an vielen Stellen Fehler auftreten, wie schon im vorherigen Kapitel zu erkennen war. Neben Fehlern, die durch Programmierfehler am SitePrototyper verursacht werden, sind hier besonders solche interessant, die bei der Erstellung einer Website seitens der Benutzer des SitePrototypers entstehen.
Viele dieser Fehler können vom System erkannt und somit gemeldet werden. Das ist in Bezug auf deren Beseitigung Gegenstand dieses Kapitels. Zunächst sei festgestellt, dass ein Fehler während der Erstellung einer Website durch das Anlegen oder Modifizieren einer Datei bzw. des Zusammenspiels mehrerer solcher Dateien ausgelöst wird. Insofern sei angenommen, dass sich die Beseitigung eines Fehlers in der Regel durch die Modifikation einer speziellen Datei beseitigen lässt. Dies kann vom System unterstützt werden, indem möglichst exakt die Position des Fehlers mitgeteilt wird, z.B. durch Angabe dieser Datei.

Da die Website über einen Webbrowser visualisiert wird, bietet es sich an, dass auch Fehlermeldungen über diesen Kanal übermittelt werden. Eine Möglichkeit der Fehlermeldung besteht darin, anstelle der angeforderten Ressource eine speziell erzeugte HTML-Seite zurückzuliefern, in der die Fehlermeldung enthalten ist. Davon muss dann Gebrauch gemacht werden, wenn der Fehler so tiefgreifend ist, dass eine Fortführung des Verarbeitungsprozesses nicht möglich ist. Sofern der Fehler weniger tiefgreifend ist und sich die Anfrage auf eine Page bezieht, kann die Fehlermeldung in den Inhalt integriert und somit gemeldet werden. Dies macht beispielsweise dann Sinn, wenn nur die Ausführung eines einzelnen Macros missglückt ist. Über eine entsprechend definierte Darstellung kann die Fehlermeldung in der resultierenden HTML-Seite unterdrückt oder durch eine besondere Hervorhebung kenntlich gemacht werden.

Für die erste beschriebene Möglichkeit wird ein Mechanismus verwendet, der innerhalb der Servlet-Engine integriert ist. Fehler können auf diese Weise intern als sog. „ServletException“ gemeldet werden, was in einer speziellen Behandlung der Ausgabe mündet. So wird als Konsequenz eine HTML-Seite generiert, in der diese „Exception“ inklusive einer möglichen Fehlermeldung im Klartext enthalten ist.

Bei der zweiten Möglichkeit wird vom System ein Element <error> erzeugt und darin die Fehlermeldung eingebettet. Dieses Element wird dann an entsprechender Stelle in den Inhalt integriert. Ein solches Element kann beispielsweise so aussehen:

```
<error type=”parser”>Parsingerror in file ‘c:\test.xml’ at line#12 column#1</error>
```

Über ein zusätzliches Attribut „type“ wird der Fehler über die drei Möglichkeiten „parser“, „assembler“ und „macro“ noch genauer spezifiziert. Ein Fehler beim Parsing wird über den Typ „parser“ gemeldet, während „macro“ und „assembler“ für einen Fehler bei der Ausführung eines Macros bzw. bei dem Verschmelzen mehrerer Inhalte stehen.

Im beigefügten Stylesheet (vgl. Kapitel 10.2) ist festgelegt, dass Fehlermeldungen über <error> auf besonders auffällige Art dargestellt werden. Insofern kann vom Benutzer problemlos festgestellt werden, dass überhaupt ein Fehler aufgetreten ist. Da es aber unter Umständen entscheidend ist zu erfahren, an welcher Stelle im Inhalt er aufgetreten ist, besteht die Möglichkeit die Überführung des Inhalts nach HTML zu unterdrücken und sich stattdessen den Inhalt als XML anzeigen zu lassen. Zu diesem Zweck kann ein spezielles Stylesheet verwendet werden, in dem eine Identitätstransformation des Inhalts beschrieben wird.

Aufgrund ihres starken Bezugs zur Umgebung von JAVA und XML wird für die Interpretation der Fehlermeldungen ein gewisses Maß derartiger Kenntnisse vorrausgesetzt. Damit sollten zumindest grobe Rückschlüsse auf die Natur und Position des Fehlers möglich sein.
Im Folgenden sollen die möglichen Fehler klassifiziert und Möglichkeiten für deren Beseitigung aufgezeigt werden:

Anfragefehler
Hierbei handelt es sich um Fehler, die dann auftreten, wenn mittels einer URL eine nicht vorhandene Ressource angefordert wird. Entweder ist die spezifizierte Website nicht vorhanden, oder aber an der angegeben Position gibt es kein Object oder keine

Page mit dem entsprechenden Namen. Durch das Überprüfen der Richtigkeit der Anfrage oder durch das Anlegen der entsprechenden Ressource sollte sich der Fehler beheben lassen.

Parsingfehler
Der SitePrototyper macht in hohem Maße von der Möglichkeit gebrauch, XML-Dateien zu parsen. Es kann sich dabei um ein Stylesheet, eine Inhaltsdatei oder eine DTD handeln. Wie schon erwähnt wurde, können dabei zwei Arten von Fehlern entdeckt werden: Einerseits kann die Datei syntaktisch fehlerhaft sein, während andererseits auch eine Invalidität in Bezug auf eine DTD vorliegen kann.
Der Vorteil dieser Art von Fehlern liegt darin, dass jeweils ziemlich exakt spezifiziert werden kann, in welcher Datei bzw. welcher Zeile und Spalte dieser Datei der Fehler aufgetreten ist.
Insofern sollte eine Fehlerbeseitigung in Bezug auf die konkrete Position nicht schwierig sein und kann durch Modifikation der entsprechenden Datei erfolgen.

Ausführungsfehler
Zu den Ausführungsfehlern zählen solche, bei denen ein spezieller Prozess nicht vollständig ausgeführt werden konnte. Zu solchen Prozessen zählen das Ausführen von Macros, das Verschmelzen mehrerer Pages und das Anwenden eines Stylesheets. Da es sich hier um eine sehr allgemeine Fehlerklasse handelt, kann kein allgemeingültiger Hinweis gegeben werden, wie diese Fehler zu beseitigen sind. Es soll dabei nicht ausgeschlossen werden, dass Fehler nicht auch durch den SitePrototyper selbst bzw. eine seiner integrierten Komponenten ausgelöst worden sein kann.

8.4. Anwendungsbeispiele

Nachdem in den vorherigen Kapiteln eine grundsätzliche Funktionsweise und Handhabung des SitePrototypers beschrieben wurde, sollen an dieser Stelle Beispiele aufgeführt werden, die sich aus der Benutzung des Kernsystems seitens des Autors ergeben haben. Dabei wurden die wesentlichen Eigenschaften des SitePrototypers zur Anwendung gebracht und in der Praxis ausgetestet.

8.4.1. Entwurfsmuster der Navigation

Es sind in dieser Diplomarbeit einige Navigationselemente implementiert worden, die als Entwurfsmuster zur Lösung von Standardproblemen dienen können. Es handelt sich um solche, die häufiger auf diversen Websites zu finden sind und somit grundsätzlich als bekannt vorrausgesetzt werden können.
Diese Entwurfsmuster sind in Form von Macros erstellt und können somit in jede beliebige Website innerhalb dieses Systems eingebunden werden. Es handelt sich dabei einerseits um zwei „Integrated Navigation Elements“ (vgl. Kapitel 7.4.1), die als generische Lösung einer Navigation zu verstehen sind (vgl. Kapitel 7.4.2) und andererseits um das „Remote Navigation Element“ der Sitemap.
Die hier erläuterten Macros erzeugen jeweils ein Element <navigation>, in das beispielsweise die Elemente <hyperlink> und <label> eingebettet sind. Dabei werden z.B. auch die entsprechenden Attribute von <hyperlink> gesetzt (s.o.), die ihrerseits eine Unterstützung in der Darstellung über das Basis-Stylesheet finden.

ChildrenNavigationMacro
Das „ChildrenNavigationMacro“ dient dazu, automatisiert Hyperlinks auf die jeweils untergeordneten Pages zu generieren und in einem Element <navigation> zusammenzufassen. Dies geschieht unter Berücksichtigung von zweierlei Besonderheiten:

Einerseits wird davon ausgegangen, dass das Label der jeweiligen Zielpage als ein Attribut definiert ist, so dass es automatisch in den jeweiligen Hyperlink übernommen werden kann. Sollte dies nicht der Fall sein, wird als Label der Name (s. Kapitel 8.2.1) der jeweiligen Page verwendet.
Andererseits werden auch Gruppierungen berücksichtigt, die über die Verwendung „einfacher“ Container entstehen: Pages, die sich innerhalb eines solchen Containers befinden, werden in einem eigenen Element <navigation> zusammengefasst und mit dem Label des jeweiligen Containers versehen.
Dies soll an folgendem Beispiel demonstriert werden, in dem das Ergebnis einer solchen Berechnung für die Page „Produkte & Dienstleistungen“ aus Abbildung 8.2 dargestellt wird:

```
<navigation name="children">
    <navigation>
        <label>
            <text>Produkte</text>
        </label>
        <hyperlink self="false">
            <url exists="true">Produkte/RechnerA</url>
            <label>
                <text>Rechner A</text>
            </label>
        </hyperlink>
        <hyperlink self="false">
            <url exists="true">Produkte/RechnerB</url>
            <label>
                <text>Rechner B</text>
            </label>
        </hyperlink>
    </navigation>
    <hyperlink self="false">
        <url exists="true">Reparaturen</url>
        <label>
            <text>Reparaturen</text>
        </label>
    </hyperlink>
</navigation>
```

PathNavigationMacro

Dieses Macro generiert einen Pfad über alle Pages, ausgehend von der Homepage bis zur jeweils aktuellen Page, so wie es prinzipiell in Kapitel 2.7 beschrieben ist. Obwohl es sich dabei einerseits um eine Kontextinformation handelt, dient es andererseits durch die Verwendung der Pfadelemente als Hyperlinks zusätzlich als echtes Navigationselement. Dies soll anhand von folgendem Beispiel demonstriert werden, das wiederum auf Abbildung 8.2 Bezug nimmt und das Ergebnis für die Page „Rechner A“ darstellt:

```
<navigation name="path">
       <hyperlink self="false">
       <url exists="true">/sites/firmaxyz</url>
       <label>
              <text>Firma XYZ</text>
       </label>
       </hyperlink>

       <hyperlink self="false">
       <url exists="true">/sites/firmaxyz/ProdukteUndDienstleistungen</url>
       <label>
              <text>Produkte &amp;amp; Dienstleistungen</text>
       </label>
       </hyperlink>

       <hyperlink self="true">
       <url exists="true">/sites/firmaxyz/ProdukteUndDienstleistungen/Produkte/RechnerA</url>
       <label>
       <text>Rechner A</text>
       </label>
       </hyperlink>
   </navigation>
```

SitemapMacro
Wie schon erwähnt, stellt das SitemapMacro ein sog. „Remote Navigation Element“ dar, in dem komplett alle Pages einer Website aufgelistet werden und somit für einen Überblick über den gesamten Inhalt sorgen. Durch die zusätzlich integrierten Hyperlinks ist über die Sitemap auch das direkte Springen zu jeder einzelnen Page möglich.
Das Macro erzeugt eine Struktur von Elementen, die prinzipiell der des „ChildrenNavigationMacro“ (s.o.) entspricht, wobei in diesem Fall aber von der Homepage ausgegangen wird und der komplette Hierarchiebaum der Struktur abgebildet wird. Dies soll anhand von folgendem rekursiven Schema erläutert werden:

```
1.)
<navigation name=“sitemap“>
        <hyperlink>[Hyperlink auf die Homepage]</hyperlink>
        <navigation>
                [<hyperlink> Elemente auf untergeordnete Pages, gemäß 3.)]
                [<navigation> Elemente für untergeordnete Container, gemäß 2.)]
        </navigation>
</navigation>

2.)
<navigation>
        <label>[Label des Containers]</label>
        [<hyperlink> Elemente auf untergeordnete Pages, gemäß 3.)]
        [<navigation> Elemente für untergeordnete Container, gemäß 2.)]
</navigation>

3.)
<hyperlink>[Hyperlink auf die Page]</hyperlink>
<navigation>
        [<hyperlink> Elemente auf untergeordnete Pages, gemäß 3.)]
        [<navigation> Elemente für untergeordnete Container, gemäß 2.)]
</navigation>
```

Ausgehend vom Schema 1.) werden rekursiv die Elemente aus Schema 2.) und 3.) ineinander eingesetzt und dabei die Struktur der Website über ineinandergeschachtelte Elemente abgebildet.

8.4.2. Zwei Beispiel-Websites

Um Erkenntnisse zur Praxistauglichkeit des SitePrototypers zu gewinnen, wurden mit diesem System zwei Beispiel-Websites erstellt. Dabei handelt es sich einerseits um die Umsetzung des hier häufiger verwendeten Beispiels der „Firma XYZ“, was als Internetpräsenz einer fiktiven Firma zu verstehen ist. Andererseits wurde mit dem SitePrototyper auch ein Teilausschnitt der Website des Arbeitsbereiches ASI[20] nachmodelliert.
Beide Beispiele sind nicht als vollständige Websites zu sehen, sondern sollen als Demonstration wesentlicher Eigenschaften des SitePrototypers verstanden sein. Sie sind dieser Diplomarbeit in digitaler Form als Bestandteil des SitePrototypers beigefügt und können damit auch in der Praxis begutachtet werden.

Firma XYZ-Website
Die Website „Firma XYZ“ entspricht strukturell dem Informationsraum aus Abbildung 5.1, wobei die Inhalte komplett über den allgemeinen Dokumenttyp „generic“ (s.o.) definiert und mit vergleichsweise inhaltsleeren Phrasen gefüllt wurden. Bei der Darstellung wurde ebenfalls auf die vorgefertigte Lösung des Basis-Stylesheet zurückgegriffen, während die Navigation über die oben beschriebenen Navigationselemente „ChildrenNavigation“, „PathNavigation“ und „Sitemap“ beschrieben wird. Für letzteres

[20] „Angewandte und Sozialorientierte Informatik“, Fachbereich Informatik, Universität Hamburg

Element wurde eine zusätzliche Page in die Struktur eingefügt, die ausschließlich zur Darstellung der Sitemap dienen soll. Weiterhin wurden die Inhalte des Teilbaumes „Produkte & Dienstleistungen“ als „verschmolzen“ gekennzeichnet (vgl. Kapitel 8.2.1), um diese Funktionsweise in der Praxis testen und demonstrieren zu können.

Ziel dieses Beispieles war es, die hier genannten Elemente bzw. auch den SitePrototyper selbst auf korrekte Funktionsweise zu testen und gleichzeitig auch einen Demonstrationsgegenstand vorzuweisen. Als wesentliches Ergebnis sei prinzipiell die korrekte Funktionsweise der Elemente festgestellt.
Allerdings entstand der Eindruck, dass das Konzept der „Verschmelzung“ alles andere als ausgereift ist. Beispielsweise werden dadurch Hyperlinks ungültig, in denen auf vormals unverschmolzene Pages referenziert wird. Außerdem wurde festgestellt, dass insbesondere die Verschmelzung der einzelnen Attribute und Navigationselemente durchaus problematisch ist, da in letzterem Fall oftmals Navigationsalternativen entstehen, die in dem neuen Kontext keinen Sinn machen. Dies lässt sich allerdings über eine Unterdrückung dieser Inhalte mittels der Darstellung lösen.

ASI-Website
Die Erstellung der ASI-Website sollte der Kontrolle darüber dienen, ob sich eine real existierende Website auch wirklich mithilfe der Mittel des SitePrototypers beschreiben lassen kann. Dazu wurden vier Pages angelegt, nämlich die „Homepage“, die Übersichtsseite „Personen“ und innerhalb von „Personen“ die Pages „Horst Oberquelle“ und „Wiebke Oeltjen“. Viele der übrigen Pages dieser Websites wurden zusätzlich als „Dummies“ erzeugt und dienen nur der Vervollständigung der Struktur bzw. der Navigationsalternativen.
Zur Beschreibung der Navigation wurde wieder ausschließlich auf die Navigationselemente „ChildrenNavigation“ und „PathNavigation“ zurückgegriffen, womit zwar die wesentlichen Navigationselemente wiedergegeben, aber speziellere nicht berücksichtigt wurden.
Die Homepage und die „Personenübersicht“-Page wurden wieder als „Generic“-Dokumenttyp erstellt, während für die beiden „Personen“-Pages ein neuer Dokumenttyp angelegt wurde. Dies macht deshalb Sinn, weil die ersten beiden Pages eine vergleichsweise universale inhaltliche Struktur aufweisen, während die letzteren beiden einen sehr speziellen Inhalt haben. Da alle „Personen“-Pages auf der Website eine sehr ähnliche inhaltliche Struktur aufweisen, stellt ein expliziter Dokumenttyp auch ein Entwurfsmuster dar und kann als Vorlage für alle weiteren eingefügten „Personen“ dienen.
Zur Darstellung wurde ein neues Stylesheet erstellt, das die wesentlichen optischen Merkmale der Original-Website wiederspiegelt. Gleichzeitig kann damit auch die Entkoppelung von Inhalt und Darstellung demonstriert werden, da sich die Website sowohl über das Basis-Stylesheet, als auch über das speziellere Stylesheet darstellen lässt.

Als Ergebnis sei an dieser Stelle festgehalten, dass es keine Schwierigkeiten gemacht hat, die bestehende Website ausschnittsweise in eine, dem SitePrototyper konforme, Repräsentation zu überführen. Dies sei insbesondere in Hinblick auf die Verwendung der Standard-Navigationselemente festgestellt, die sich problemlos integrieren ließen und offenbar genau einen Teil des ursprünglichen Navigationskonzeptes wiederspiegeln. Allerdings sei auch erwähnt, dass dazu eine leichte aber unmerkliche Änderung an der Struktur notwendig war.

Insgesamt ist also zu vermuten, dass eine Modellierung von beliebigen Websites mittels des SitePrototypers möglich ist und über Standardelemente auch erleichtert wird.

9. Fazit und Diskussion

Das Ziel dieser Diplomarbeit war es, sich des Entwurfs von Informationssystemen im WWW unter dem Aspekt der Benutzerorientierung anzunehmen. Dabei sollten einerseits im weiteren Sinne Möglichkeiten zum Vorgehen aufgezeigt und andererseits ein Werkzeug entwickelt werden, mit dem sich ein solches Vorgehen unterstützen lässt.

Unter Einbeziehung von Erkenntnissen aus der grundsätzlichen Thematik ist dabei einerseits ein Modell einer Website entwickelt worden, das als Entwicklungsgrundlage für einen Entwurfsprozess dienen kann. Weiterhin ist ein solcher Entwurfsprozess in seinen einzelnen Schritten und Aspekten dargelegt worden, wobei immer nach Möglichkeiten gesucht wurde, benutzerorientiert vorzugehen.
Mittels des SitePrototypers ist dann ein Werkzeug entworfen und auch als Prototyp implementiert worden, welches zumindest theoretisch ein Hilfsmittel für den Entwurf von Websites darstellt. Die Art und Weise, wie dabei eine Unterstützung des Entwicklers stattfinden kann, soll hier zusammengefasst werden:

Einerseits orientiert sich der SitePrototyper an dem Modell einer Website, indem sich die dort beschriebenen Komponenten problemlos auf dessen Entwurfseinheiten abbilden lassen. Da sich der beschriebene Entwurfsprozess ebenfalls auf diese Komponenten bezieht, lässt sich der SitePrototyper dort nahtlos integrieren. Für die Dokumentation der einzelnen Entwurfsergebnisse ergeben sich dabei gleichzeitig ganz neue Möglichkeiten, da der SitePrototyper spezielle Lösungen dafür bereithält. Beispielsweise lässt sich die in Kapitel 7.3 beschriebene Strukturierung mittels der in Kapitel 8.2.1 geschilderten Möglichkeiten dokumentieren.

Die Funktionalität des SitePrototypers, diese Ergebnisse auch unmittelbar mittels einer navigierbaren Website bereitzustellen, lässt die Möglichkeit entstehen, eine entsprechend frühe Evaluation der Zwischenergebnisse stattfinden zu lassen. Weiterhin ist eine Modifikation der Entwurfsergebnisse durch simple Manipulation der entsprechenden Dateien jederzeit möglich, so dass z.B. mit minimalem Aufwand Entwurfsalternativen evaluiert werden können.
Durch bereitgestellte Standardlösungen, z.B. in Bezug auf die Navigation oder die Darstellung, kann die Evaluation eines Prototypen sogar ohne einen vorherigen expliziten Entwurf dieser Komponenten durchgeführt werden.

Auf den Punkt gebracht, sind dadurch folgende Vorteile für den Entwurf einer Website zu erwarten:

- *Zeitersparnis*: Prototypen müssen nicht mehr manuell erstellt werden, sondern werden automatisiert aus den aktuellen Entwurfsergebnissen generiert.
- *Wiederverwendbarkeit*: Die Überarbeitung der einzelner Entwurfsergebnisse bedingt keine komplette Überarbeitung der Entwicklungsdokumente, sondern es kann auf vorherige zurückgegriffen und partielle Änderungen können durchgeführt werden.
- *Benutzerorientierung*: Durch die Möglichkeit der schnellen Erstellung von Prototypen können unmittelbarer Benutzbarkeitsprobleme entdeckt und behoben werden.
- *Verteiltes Arbeiten*: Aufgrund der Integration mehrerer unterschiedlicher Entwicklungsdokumente wird das verteilte Arbeiten gemäß der Einzelschritte unterstützt.

- Nahtlose Implementation: Die Ergebnisse der automatischen Generierung stellen eine vollständige Website dar, die sich als implementiertes Endergebnis verwenden lässt.

Allerdings sollen auch einige negative Aspekte des SitePrototypers nicht unerwähnt bleiben: Durch das Fehlen einer vernünftigen Benutzungsoberfläche, ist der Entwickler gezwungen, die Website mittels Dateien anzulegen und zu bearbeiten. Die dabei verwendeten Formate wie XML, XSLT und JAVA werden dabei als bekannt vorrausgesetzt und müssen entsprechend beherrscht werden. Dies bezieht sich ebenso auf die Art und Weise, mit der vom SitePrototyper Fehler gemeldet werden. Zur Interpretation dieser Fehler ist nämlich oftmals ein ebensolches Wissen notwendig.
Somit ist der damit verbundene Lernaufwand seitens der Entwickler ein nicht zu unterschätzender Faktor, der sich aber aufgrund der hohen Verbreitung dieser Formate als möglicherweise nicht so dramatisch darstellt. An dieser Stelle sei für weiterführende Arbeiten auf die Möglichkeit hingewiesen, ein entsprechend benutzungsfreundliches Interface zu entwickeln.

Es ist zu erwarten, dass sich der Entwurf von webbasierten Informationssystemen mittels der Ergebnisse dieser Arbeit in vielerlei Hinsicht verbessern lässt. Das bedeutet einerseits, dass das Ergebnis des Entwurfs bei Anwendung der hier geschilderten Möglichkeiten von hoher Qualität im Sinne von Benutzergerechtheit sein wird. Dies hat seinerseits Konsequenzen in Bezug auf den Erfolg einer solchen Entwicklung und der damit verbundenen Zielsetzung.
Andererseits erfährt auch der Entwickler eine Unterstützung, indem ihm ein Rahmen gegeben wird, nach dem der Entwurf stattfinden kann. Durch das Werkzeug des SitePrototypers lassen sich zudem bestimmte Entwurfsschritte automatisieren, was in Bezug auf die gesamte Entwicklungszeit als vorteilhaft vermerkt werden kann.

Allerdings ist weder das hier geschilderte Verfahren, noch der SitePrototyper in Bezug auf den praktischen Einsatz seitens der potentiellen Benutzer, den Entwicklern von Websites, evaluiert worden. Eine Evaluation fand ausschließlich seitens des Autors und damit des Entwicklers des SitePrototypers statt, was bekanntermaßen (s.o.) grundsätzlich nicht unproblematisch ist. Insofern sind die hier geschilderten Ergebnisse zunächst unter Vorbehalt zu sehen. Für die genauere Bewertung dieser Resultate und Erfahrungen sei damit auf eventuell weiterführende Arbeiten verwiesen.

10. Anhang

10.1. Basis-DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Datentyp: Text -->
<!ENTITY % inline "#PCDATA|br|hyperlink|object|macro">

<!ELEMENT text (#PCDATA)>
<!ELEMENT structuredtext (h1|h2|h3|p|ol|ul|object|macro)*>

<!ELEMENT p (%inline;)*>
<!ELEMENT br EMPTY>

<!ELEMENT h1 (#PCDATA)>
<!ELEMENT h2 (#PCDATA)>
<!ELEMENT h3 (#PCDATA)>

<!ELEMENT ol (li)*>
<!ELEMENT ul (li)*>

<!ELEMENT li (%inline;)*>


<!-- Datentyp: Hyperlink -->
<!ELEMENT label (text?,icon?)>
<!ELEMENT icon (url|macro)>

<!ELEMENT url (#PCDATA)>
<!ATTLIST url exists (true|false|unknown) "unknown">

<!ELEMENT hyperlink (label,url)>
<!ATTLIST hyperlink self (true|false) "false">


<!-- Datentyp: Object -->
<!ELEMENT object (url|macro)>
<!ATTLIST object type CDATA #IMPLIED>


<!-- Inhaltstyp: Attribute -->
<!ELEMENT attributes ANY>


<!-- Inhaltstyp: Navigation -->
<!ELEMENT navigation (label?,(navigation|hyperlink|macro)+)>
<!ATTLIST navigation name (CDATA) #IMPLIED>



<!-- Macro -->
<!ELEMENT macro       (#PCDATA)>
<!ATTLIST macro class CDATA #REQUIRED>

<!-- Fehlermeldung -->
<!ELEMENT error (#PCDATA)>
<!ATTLIST error type (parser|assembler|macro|unknown) "unknown">

<!-- Allgemeiner Dokumenttyp -->
<!ELEMENT generic (attributes,navigation,content)>
<!ELEMENT content (structuredtext|text)*>
```

10.2. Basis-Stylesheet

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" indent="yes"/>


<!-- page in htmlskelett wandeln -->
<xsl:template match="/">
<HTML>
       <HEAD>
              <TITLE><xsl:value-of select="/*/attributes/label/text"/></TITLE>
       </HEAD>
```

```
        <BODY BGCOLOR="#E0E0E0">

                <TABLE BORDER="1" WIDTH="100%">
                <TR>
                        <TD WIDTH="80%" VALIGN="TOP">
                        <!-- Inhalt -->
                        <xsl:apply-templates select="/*/*[local-name()!='navigation' and
                                                        local-name()!='attributes']"/>
                        </TD>

                        <TD WIDTH="20%" VALIGN="TOP">
                                <!-- Navigation -->
                                <xsl:for-each select="/*/navigation/navigation">
                                        <xsl:apply-templates select="."/>
                                        <HR/>
                                </xsl:for-each>


                                <xsl:for-each select="/*/navigation/hyperlink">
                                        <xsl:apply-templates select="."/>
                                        <BR/>
                                </xsl:for-each>

                                <xsl:apply-templates select="/*/navigation/error"/>
                        </TD>
                </TR>
                </TABLE>
        </BODY>

        <xsl:apply-templates select="error"/>
</HTML>
</xsl:template>


<!-- hyperlink -->
<xsl:template match="hyperlink">
        <xsl:choose>
                <!-- Wenn es ein Hyperlink auf sich selbst ist,
                dann nur das Label darstellen -->

                <xsl:when test="@self='true'">
                        <B><xsl:apply-templates select="label"/></B>
                </xsl:when>

                <!-- ansonsten ... -->
                <xsl:otherwise>
                        <xsl:choose>
                        <!-- Wenn die url explizit als nicht existent gekennzeichnet ist,
                             dann nur das Label darstellen -->
                          <xsl:when test="url[@exists='false']">
                                  <I><xsl:apply-templates select="label"/></I>
                          </xsl:when>

                          <!-- Ansonsten ganz normal darstellen -->
                          <xsl:otherwise>
                                  <A HREF="{url}"><xsl:apply-templates select="label"/></A>
                          </xsl:otherwise>
                        </xsl:choose>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>


<!-- label -->
<xsl:template match="label">
        <xsl:if test="icon">
                <IMG SRC="{icon/url}" BORDER="0" ALIGN="BOTTOM"/>
                <BR/>
        </xsl:if>

        <xsl:value-of select="text"/>
</xsl:template>


<!-- navigation -->
<xsl:template match="navigation[@name='path']">
        <xsl:if test="*[last()&gt;1]">
```

```
        <!-- Nur darstellen, wenn es mehr als einen Eintrag gibt -->


        <TABLE BORDER="0">
        <TR>
                <TD ALIGN="center"><B><xsl:apply-templates select="label"/></B></TD>
        </TR>

        <xsl:for-each select="hyperlink">
        <TR>
                <TD ALIGN="center">
                        <xsl:apply-templates select="."/>
                </TD>
        </TR>
        <xsl:if test="not(position()=last())">
        <TR>
                <TD ALIGN="center"><B>\/</B></TD>
        </TR>
        </xsl:if>

                </xsl:for-each>
        </TABLE>
        </xsl:if>
</xsl:template>

<xsl:template match="navigation">
        <xsl:if test="*"><!-- Nur darstellen, wenn es Nackommen gibt -->
                <TABLE BORDER="0">
                <TR>
                        <TD COLSPAN="2">
                        <B><xsl:apply-templates select="label"/></B>
                        </TD>
                </TR>

                <xsl:for-each select="hyperlink|navigation">
                <TR>
                        <TD> </TD>
                        <TD>
                                <xsl:apply-templates select="."/>
                        </TD>
                </TR>
                </xsl:for-each>

                </TABLE>
        </xsl:if>
</xsl:template>


<!-- error -->
<xsl:template match="error">
        <TABLE BGCOLOR="red"><TR><TD>
                <B>Error (<xsl:value-of select="@type"/>)</B>
                <xsl:text> </xsl:text><xsl:apply-templates/>
        </TD></TR></TABLE>
</xsl:template>


<!-- object[image] -->
<xsl:template match="object[@type='image']">
        <TABLE BORDER="0">
        <TR>
                <TD><IMG SRC="{url}" BORDER="0"/></TD>
        </TR>
        </TABLE>
</xsl:template>



<!-- b,i,p,br,h1,h2,h3,ol,ul,li -->
<xsl:template match="p">
        <P><xsl:apply-templates/></P>
</xsl:template>

<xsl:template match="b">
        <B><xsl:apply-templates/></B>
</xsl:template>

<xsl:template match="i">
```

```
        <I><xsl:apply-templates/></I>
</xsl:template>

<xsl:template match="br">
        <BR/>
</xsl:template>

<xsl:template match="h1">
        <H1><xsl:apply-templates/></H1>
</xsl:template>

<xsl:template match="h2">
        <H2><xsl:apply-templates/></H2>
</xsl:template>

<xsl:template match="h3">
        <H3><xsl:apply-templates/></H3>
</xsl:template>

<xsl:template match="ol">
        <OL><xsl:apply-templates/></OL>
</xsl:template>

<xsl:template match="ul">
        <UL><xsl:apply-templates/></UL>
</xsl:template>

<xsl:template match="li">
        <LI><xsl:apply-templates/></LI>
</xsl:template>


<!-- Default Content-->
<xsl:template match="*">
        <xsl:choose>
                <xsl:when test="local-name='navigation' or local-name='attributes'">
                        <!-- "attributes" und "navigation" ignorieren -->
                </xsl:when>

                <xsl:otherwise>
                        <!-- Standardmäßig alle Inhalte zeilenweise auflisten -->
                        <xsl:value-of select="text()"/>
                        <BR/>
                        <xsl:apply-templates select="*[local-name()!='navigation' and
                                                       local-name()!='attributes']"/>
                </xsl:otherwise>
        </xsl:choose>
</xsl:template>


</xsl:stylesheet>
```

10.3. API des SitePrototypers

Mittels der Macros besteht beim SitePrototyper die Möglichkeit, Programme zu entwickeln, mit denen automatisiert Inhalte innerhalb von Pages generiert werden können. Um dabei auf den aktuellen Kontext der jeweiligen Page bezugnehmen zu können, wurde ein „Application Program Interface“ (API) entwickelt, mit dessen Hilfe z.B. auf die Struktur und die Inhalte der Website zugegriffen werden kann. Dazu werden eine Menge von Klassen und Methoden bereitgestellt, die an dieser Stelle beschrieben werden sollen. Es handelt sich dabei um eine eher technische Notation, die sich an das Format JavaDoc[21] anlehnt.
Alle hier beschriebenen abstrakten Klassen befinden sich in einem Paket „de.peterkaul.siteprototyper“ und nehmen neben den üblichen JAVA-Paketen teilweise auch Bezug auf ein weiteres Paket „org.w3c.dom“, das als Bestandteil des „Document Object Model“ zu verstehen ist [Wood 00].

[21] http://java.sun.com/products/jdk/javadoc (25.10.2000)

Resource
public abstract interface Resource

Eine Klasse zur übergeordneten Beschreibung der drei Website-Einheiten "Page", "Object" und "Container". Die bereitgestellten Methoden lassen einerseits den Zugriff auf die Struktur der Website und andererseits auf den Inhalt der jeweiligen Resource zu. Sie werden an die entsprechenden Klassen "Page", "Object" und "Container" vererbt und dort durch weitere Methoden teilweise verfeinert.

Methoden	`public boolean equals(Resource resource)`
	Testet, ob die übergebende Resource jeweils der aktuellen entspricht, d.h. ob jeweils diesselbe Page, Container oder Object repräsentiert wird. Je nachdem wird als Ergebnis einer der booleschen Werte „true“ oder „false“ zurückgeliefert.
	`public java.io.File getFile()`
	Liefert die Datei zurück, in der die jeweilige Resource beschrieben ist (vgl. Kapitel 8.2.1)
	`public String getName()`
	Liefert den Namen der Resource zurück, der sich aus dem Dateinamen der repräsentierten Datei ergibt. Entsprechend dem Schema der Dateinamen (vgl. Kapitel 8.2.1) entspricht dies der Komponente „name“.
	`public Container getParent()`
	Bestimmt den jeweils übergeordneten Container der Resource und liefert dessen Repräsentation zurück. Sofern es sich bei der Resource um die Wurzel-Page („Homepage“) handelt, ist das Ergebnis „null“.
	`public Site getSite()`
	Liefert die Repräsentation der jeweiligen Website (s.u.) zurück, in der sich die Resource befindet.
	`public boolean isContainer()`
	Bestimmt, ob die Resource einen Container darstellt und liefert einen entsprechenden booleschen Wert zurück. Im Fall, dass die Resource eine Container-Page ist (vgl. Kapitel 5.2), wird hier „false“ zurückgegeben.
	`public boolean isObject()`
	Bestimmt analog, ob die Resource ein Object darstellt.
	`public boolean isPage()`
	Bestimmt analog, ob die Resource eine Page darstellt.

Container
public abstract interface Container implements Resource

Repräsentation der Website-Einheit "Container" als Ableitung des allgemeineren Konstrukts der "Resource".

Methoden	`public org.w3c.dom.Document getAttributes()`
	Diese Methode liefert die Attribute des Containers in Form eines Dokumentes des "Document Object Model" zurück. Das Wurzel-Element wird entsprechend einer Konvention den Namen "attributes" tragen (vgl. Kapitel 8.2.2).
	`public Resource[] getChildren()`
	Liefert alle untergeordneten Ressourcen (Pages, Objects und Container) des Containers in einem Array zurück, wobei die Reihenfolge des ihrer vorgegebenen Ordnung (vgl. Kapitel 8.2.1) entspricht.

Page
public abstract interface Page implements Container

Repräsentation der Website-Einheit "Page" als Ableitung des übergeordneten Konstrukts des "Containers".

Methoden	`public org.w3c.dom.Document getContent()` Liefert den kompletten Inhalt der Page zurück, d.h. die Informationselemente inklusive der Attribute und der Navigation. Das Ergebnis ist ein Dokument des „Document Object Model“.

Object
public abstract interface Object implements Resource

Repräsentation der Website-Einheit "Object" als Ableitung des allgemeineren Konstrukts der "Resource".

Methoden	`public org.w3.dom.Document getAttributes()`
	Diese Methode liefert die Attribute des Objects analog zu denen des Containers (s.o.) zurück.
	`public java.io.InputStream getBinary()`
	Liefert den Inhalt des Objects als Strom von Binärdaten zurück.
	`public String getMimeType()`
	Liefert den MIME-Type des Objects als String zurück, sofern er anhand der Extension der entsprechenden Datei (s.o.) ermittelt werden kann. Ansonsten wird „null“ zurückgeliefert.

Site
public abstract interface Site

Die Repräsentation einer Website: Es werden eine Menge von Methoden bereitgestellt, die in Bezug auf den programmiertechnischen Umgang mit der Website von Bedeutung sein können.

Methoden	`public Resource getResource(Container context, String path)`
	
	Mit dieser Methode lassen sich die Ressourcen der Website (d.h. Pages, Objects und Container) anhand ihres Pfades bestimmen. Dabei wird die Website, ausgehend vom übergebenen Container, entsprechend des übergebenen Pfades durchsucht.
	Der Pfad besteht aus einzelnen Komponenten, die mittels "/" voneinander getrennt sind. Dabei sind als Komponenten neben den Namen der entsprechenden Ressourcen auch die Ausdrücke ".." und "." zulässig. Mit letzteren wird jeweils der übergeordneten Container, bzw. der aktuelle Container bezeichnet. Sofern dem Pfad ein "/" vorrangestellt ist, wird von der Homepage als Ausgangscontainer ausgegangen, ansonsten vom zusätzlich als Parameter übergebenen Container.
	Als Ergebnis wird die Resource zurückgeliefert, oder „null“, sofern sie nicht existiert.
	
	`public Page getRoot()`
	Liefert die Homepage der Website zurück, d.h. das Wurzel-Element der hierarchischen Struktur.
	
	`public String getUrl(Resource resource)`
	Mit dieser Methode lässt sich die URL der übergebenen Resource im Kontext der jeweiligen Website bestimmen.
	
	`public void log(int level, String message)`
	Eine Möglichkeit, programmierspezifische Ereignisse als Nachricht zu verzeichnen. Dies ist insbesondere als Mittel gedacht, um das Debugging von "Macros" zu unterstützen. Es sind drei Levels von Meldungen definiert, nämlich "ERROR", "WARNING" und "MESSAGE", die in einer unterschiedlichen Art und Weise verzeichnet werden.
	Die Nachrichten werden über die Standardausgabe des Systems ausgegeben und können so vom Benutzer eingesehen werden.

Felder	`public static final MESSAGE`
	
	`public static final WARNING`
	
	`public static final ERROR`
	
	Die drei verschiedenen Levels des Logging-Mechanismus (s.o.)

11. Literaturverzeichnis

[Bailey 97]
Samantha Bailey
Navigating the Information Architecture Maze
November 1997
http://webreview.com/wr/pub/97/11/14/arch/index.html (23.9.99)

[Balasubramanian & Bashian 98]
V. Balasubramanian und Alf Bashian
Document Management and Web Technologies : Alice Marries the Mad Hatter
In: Communications of the ACM, Juli 1998, Vol. 41, No. 7, S. 107-115

[Berners-Lee et al. 94]
T. Berners-Lee, L. Masinter und M. McCahill (Hrsg.)
RFC 1738 - Uniform Resource Locators (URL)
Dezember 94
http://rfc.fh-koeln.de/rfc/html/rfc1738.html (10.8.2000)

[Bray, Paoli & Sperberg-McQueen 98]
Tim Bray, Jean Paoli und C. M. Sperberg-McQueen (Hrsg.)
Extensible Markup Language (XML) 1.0
Januar 1998
http://www.w3.org/TR/1998/REC-xml-19980210 (15.10.2000)

[Challenger, Iyengar & Witting 00]
Jim Challenger, Arun Iyengar und Karen Witting
A Publishing System for Efficiently Creating Dynamic Web Content
März 2000
http://www.research.ibm.com/people/i/iyengar/infocom2000.ps (12.10.2000)

[Clark 99]
James Clark (Hrsg.)
XSL Transformations (XSLT) Version 1.0
1999
http://www.w3.org/TR/1999/REC-xslt-19991116 (15.10.2000)

[Deshpande, Hansen & Murugesan 99]
Yogesh Deshpande, Steve Hansen und San Murugesan
Web Engineering: Beyond CS, IS and SE
Mai 1999
http://fistserv.macarthur.uws.edu.au/san/icse99-webe/ICSE99-WebE-Proc/yogesh.doc (25.8.2000)

[Dougherty 97]
Dale Dougherty
Don't Forget To Write
Oktober 1997
http://webreview.com/97/10/10/imho/index.html (16.12.99)

[Duschek 89]
Karl Duschek
Format, Raster, Layout
In: Visuelle Kommunikation – Ein Design-Handbuch (S. 112-140)
Anton Stankowski und Karl Duschek (Hrsg.)
Berlin: Dietrich Reimer Verlag, 1989

[Eco 77]
Umberto Eco
Zeichen – Einführung in einen Begriff und seine Geschichte
Frankfurt am Main: Suhrkamp Verlag, 1977

[Engesser 93]
Hermann Engesser
Duden Informatik, 2. Auflage
Mannheim, Leipzig, Wien, Zürich: Dudenverlag, 1993

[Fleming 97]
Jennifer Fleming
In Defense of Web Graphics
Juli 1997
http://webreview.com/wr/pub/97/07/25/feature/index4.html (27.11.99)

[Fleming 98]
Jennifer Fleming
Web Navigation – Designing the User Experience
Sebastopol: O'Reilly, 1998

[Fleming 98b]
Jennifer Fleming
User Testing – How to find out what users want
Juni 1998
http://www.ahref.com/guides/design/199806/0615jefprintable.html (17.11.99)

[Floyd 84]
Christiane Floyd
A Systematic Look at Prototyping
In: Approaches to Prototyping (S. 1-18)
R. Budde, K. Kuhlenkamp, L. Mathiassen und H. Züllighoven (Hrsg.)
Berlin, Heidelberg, New York, Tokya: Springer-Verlag, 1984

[Franke 93]
Herbert W. Franke
Informationstheorie und Ästhetik
In: Die erträgliche Leichtigkeit der Zeichen. Ästhetik, Semiotik, Informatik (S. 103-119)
Frieder Nake (Hrsg.)
Baden-Baden: AGIS Verlag GmbH, 1993

[Freed & Borenstein 96]
N. Freed und N. Borenstein
Multipurpose Internet Mail Extensions (MIME) Part One
November 1996
http://rfc.fh-koeln.de/rfc/html/rfc2045.html (15.10.2000)

[Fuccella & Pizzolato 98]
Jeanette Fuccella und Jack Pizzolato
Creating Web Site Designs Based on User Expectations and Feedback
Juni 1998
http://www.sandia.gov/itg/newsletter/june98/web_design.html (20.04.2000)

[Fucella & Pizzolato 99]
Jeanette Fuccella und Jack Pizzolato
A divided approach to Web site design: Separating content and visuals for rapid results
Juni 1999
http://www-4.ibm.com/software/developer/library/wireframe/wireframe.html (22.11.1999)

[Fuccella, Pizzolato & Franks 99]
Jeanette Fuccella, Jack Pizzolato und Jack Franks
Finding out what users want from your Web site: Techniques for gathering requirements and tasks
Juni 1999
http://www-4.ibm.com/software/developer/library/moderator-guide/requirements.html (31.1.2000)

[Fuller 96]
Rodney Fuller
Measuring User Motivation from Server Log Files
1996
http://is.twi.tudelft.nl/~graaff/webuse96/ (16.04.2000)

[Gamma, Helm, Johnson & Vlissides 95]
Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides
Design Patterns – Elements of Reusable Object-Oriented Software
Reading, Massachusetts [u.a.]: Addison Wesley, 1995

[Gould, Boies & Lewis 91]
John D. Gould, Stephen J. Boies und Clayton Lewis
Making Usable, Useful, Productivity-Enhancing Computer Applications
In: Communications of the ACM, Januar 1991, Vol. 34, No. 1, S. 74-85

[Hoffmann 97]
Michael Hoffmann
Enabling Extremely Rapid Navigation in Your Web or Document (Version 2.0)
Dezember 97
http://www.well.com/user/hoff/infoaxcs.htm (30.3.98)

[Instone 97]
Keith Instone
User Test Your Web Site
1997
http://www.webreview.com/97/04/25/usability/index.html (18.04.2000)

[Isakowitz, Stohr & Balasubramanian 95]
Tomas Isakowitz, Edward A. Stohr und P. Balasubramanian
RMM: A Methodology for Structured Hypermedia Design
In: Communications of the ACM, August 1995, Vol. 38, No. 8, S. 34-44

[Jones & Lynch 99]
David Jones & Teresa Lynch
A Model for the Design of Web-based Systems that supports Adoption, Appropriation, and Evolution
Mai 1999
http://fistserv.macarthur.uws.edu.au/san/icse99-webe/ICSE99-WebE-Proc/jones-30.doc (25.8.2000)

[Jul & Furnas 97]
Susanne Jul und George W. Furnas
Navigation in Electronic Worlds – A CHI 97 Workshop
In: SIGCHI Bulletin, October 1997, Volume 29, No. 4, S. 44-49

[Kalin 99]
Sari Kalin
Mazed and Confused
April 1999
http://www.cio.com/archive/webbusiness/040199_use_content.html (17.11.1999)

[Kaul 98]
Peter Kaul
Navigation im WWW
Studienarbeit, Universität Hamburg, Fachbereich Informatik, 1998

[Kimen 99]
Shel Kimen
10 Questions about Information Architecture
Juni 1999
http://www.builder.com/Authoring/AllAboutIA/index.html (15.10.2000)

[Kornatzki 89]
Peter von Kornatzki
Text & Bild
In: Visuelle Kommunikation – Ein Design-Handbuch (S. 177-208)
Anton Stankowski und Karl Duschek (Hrsg.)
Berlin: Dietrich Reimer Verlag, 1989

[Kreitzberg 98]
Charles Kreitzberg
The LUCID Framework
1998
http://www.cognetics.com/lucid/index.html (15.10.2000)

[Kuhlen 91]
Rainer Kuhlen
Hypertext: Ein nichtlineares Medium zwischen Buch und Wissensbank
Berlin: Springer, 1991

[La Roche 95]
Walther von La Roche
Einführung in den praktischen Journalismus, 14. Auflage
München: Paul List Verlag, 1995

[Lindgaard 94]
Gitte Lindgaard
Usability Testing And System Evaluation - A guide for designing useful computer systems
London; Glasgow (u.a.): Chapman & Hall, 1994

[Macy, Anderson & Krygier 99]
Interactivity and Meaning
Sheryl Macy, Elizabeth Anderson und John Krygier
In: Information Design (S. 293-299)
Robert Jacobsen (Hrsg.)
Cambridge, Massachusetts: The MIT Press, 1999

[Merholz 98]
Peter Merholz
User-Centered Information Design
Juli 1998
http://home.netscape.com/computing/webbuilding/studio/feature19980729-2.html (20.11.99)

[Middendorf, Singer & Strobel 96]
Stefan Middendorf, Reiner Singer und Stefan Strobel
Java – Programmierhandbuch und Referenz
Heidelberg: dpunkt, 1996

[Miller 98]
Drue Miller
How Not to Write for the Web
1998
http://home.netscape.com/computing/webbuilding/studio/feature19980827-2.html (28.11.99)

[Morkes & Nielsen 97]
John Morkes und Jakob Nielsen
Concise, SCANNABLE, and Objective: How to Write for the Web
1997
http://www.useit.com/papers/webwriting/writing.html (15.10.2000)

[Morville 99]
Peter Morville
Information, Architecture, and Usability
März 1999
http://webreview.com/wr/pub/1999/03/12/arch/index.html (10.9.99)

[Murugesan, Deshpande, Hansen & Ginige 99]
San Murugesan, Yogesh Deshpande, Steve Hansen und Athula Ginige
Web Engineering: A New Discipline for Development of Web-based Systems
Mai 1999
http://fistserv.macarthur.uws.edu.au/san/icse99-webe/ICSE99-WebE-Proc/San.doc (25.8.2000)

[Nanard, Nanard & Kahn 98]
Marc Nanard, Jocelyne Nanard und Paul Kahn
Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates
In: Proceedings of the ninth ACM conference on Hypertext and hypermedia, 1998, S. 11-20

[Neale 97]
Wayne Neale
Designing Usable & Visually Appealing Web Sites
März 1997
http://www.acm.org/sigchi/chi97/proceedings/tutorial/wn.htm (27.2.1998)

[Nielsen 94]
Jakob Nielsen
Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier
1994
http://www.useit.com/papers/guerrilla_hci.html (18.04.2000)

[Nielsen 96a]
Jakob Nielsen
The Rise of the Sub-Site
September 1996
http://www.useit.com/alertbox/9609.html (26.4.2000)

[Nielsen 96b]
Jakob Nielsen
Users First: Web Usability: Why and How User Testing
August 1996
http://www.zdnet.com/devhead/stories/articles/0,4413,2137666,00.html (18.8.2000)

[Nielsen 96c]
Jakob Nielsen
Multimedia, Hypertext und Internet – Grundlagen und Praxis des elektronischen Publizierens
Braunschweig, Wiesbaden: Vieweg, 1996

[Nielsen 97a]
Jakob Nielsen
Why Web Users Scan Instead of Read
Oktober 1997
http://www.useit.com/alertbox/9710a.html (15.10.2000)

[Nielsen 97b]
Jakob Nielsen
The Use and Misuse of Focus Groups
1997
http://www.useit.com/papers/focusgroups.html (18.4.2000)

[Nielsen 98b]
Jakob Nielsen
Web Usability: Why and How
September 1998
http://www.zdnet.com/devhead/stories/articles/0,4413,2137433,00.html (4.9.2000)

[Nielsen 98c]
Jakob Nielsen
Failure of Corporate Websites
Oktober 1998
http://www.useit.com/alertbox/981018.html (6.11.99)

[Nielsen 99]
Jakob Nielsen
"Top Ten Mistakes" Revisited Three Years Later
Mai 1999
http://www.useit.com/alertbox/990502.html (4.9.2000)

[Nielsen o.D.]
Jakob Nielsen
How to Conduct a Heuristic Evaluation
(ohne Datumsangabe)
http://www.useit.com/papers/heuristic/heuristic_evaluation.html (25.8.2000)

[Nielsen o.D.b]
Jakob Nielsen
Ten Usability Heuristics
(ohne Datumsangabe)
http://www.useit.com/papers/heuristic/heuristic_list.html (25.8.2000)

[Nielsen & Sano 94]
Jakob Nielsen und Darrell Sano
SunWeb: User Interface Design for Sun Microsystem's Internal Web
1994
http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/HCI/nielsen/sunweb.html (21.04.2000)

[Nievergelt 83]
Juerg Nievergelt
Die Gestaltung der Mensch-Maschine-Schnittstelle
In: Gesellschaft für Informatik: GI-Jahrestagung: Proceedings, S. 41-50
Ingbert Kupka (Hrsg.)
Berlin: Springer, 1983

[Norton 99]
Kenneth S. Norton
Applying cross-functional evolutionary methodologies to Web development
1999
http://fistserv.macarthur.uws.edu.au/san/icse99-webe/ICSE99-WebE-Proc/norton-29.doc (17.8.2000)

[Oppermann & Reiterer 94]
Reinhard Oppermann und Harald Reiterer
Software-ergonomische Evaluation
In: Einführung in die Software-Ergonomie, 2. Auflage, S. 335-355
Edmund Eberleh, Horst Oberquelle und Reinhard Oppermann (Hrsg.)
Walter de Gruyter: Berlin, New York, 1994

[Outing 99]
Steve Outing
Some Advice on Writing, Web-style
Juni 1999
http://www.mediainfo.com/ephome/news/newshtm/stop/st061899.htm (28.11.99)

[Pflüger 95]
Jörg Pflüger
Leitbilder der Programmiersprachenentwicklung
In: Informatik und Gesellschaft, S. 196-210
Jürgen Friedrich, Thomas Herrmann, Max Peschek und Arno Rolf (Hrsg.)
Heidelberg [u.a.] : Spektrum Akadademischer Verlag, 1995

[Pomberger & Blaschek 96]
Gustav Pomberger und Günther Blaschek
Software Engineering – Prototyping und objektorientierte Software-Entwicklung, 2. Auflage
München, Berlin: Carl Hanser Verlag, 1996

[Raggett, Le Hors & Jacobs 99]
Dave Raggett, Arnaud Le Hors und Ian Jacobs (Hrsg.)
HTML 4.01 Specification
Dezember 1999
http://www.w3.org/TR/html4/ (24.10.2000)

[Rahardja 99]
Agus Rahardja
Designing Interactivity: Who are the users and what are the techniques
1999
http://ausweb.scu.edu.au/aw99/papers/rahardja/paper.html (5.11.99)

[Reisin 94]
Fanny-Michaela Reisin
Software-Ergonomie braucht Partizipation
In: Einführung in die Software-Ergonomie, 2. Auflage, S. 299-328
Edmund Eberleh, Horst Oberquelle und Reinhard Oppermann (Hrsg.)
Walter de Gruyter: Berlin, New York, 1994

[Rosenfeld 97]
Lou Rosenfeld
Particles Waves and Site Visualization
Juli 1999
http://webreview.com/pub/97/07/11/arch/index.html (2.10.99)

[Rosenfeld 98]
Lou Rosenfeld
Bottom-Up Architecture
August 1998
http://webreview.com/wr/pub/98/08/14/arch/index.html (20.9.1999)

[Rosenfeld & Morville 98]
Louis Rosenfeld und Peter Morville
Information Architecture for the World Wide Web
Sebastopol: O'Reilly & Associated, 1998

[Rosenfeld 99a]
Lou Rosenfeld
Brief „Data Does Not Equal Information" Digression
Februar 1999
http://webreview.com/wr/pub/1999/02/05/feature/index2.html (20.9.2000)

[Rosenfeld 99b]
Lou Rosenfeld
Information Architecture and Corporate Strategy: The Tail Wags the Dog
Juni 1999
http://webreview.com/wr/pub/1999/06/04/arch/index.html (15.10.2000)

[Rosenfeld 99c]
Lou Rosenfeld
Cuisinarts, E-Commerce, and … Controlled Vocabularies
Juli 1999
http://webreview.com/wr/pub/1999/07/09/web_architect/index.html (23.11.99)

[Rossi, Schwabe & Garrido 97]
Gustavo Rossi, Daniel Schwabe und Alejandra Garrido
Design Reuse in Hypermedia Application Development
In: Proceedings of the eighth ACM conference on Hypertext, April 1997, S. 57-66

[Scharl 99]
Arno Scharl
A Conceptual, User-Centric Approach To Modelling Web Information Systems
1999
http://ausweb.scu.edu.au/aw99/papers/scharl/paper.html (8.11.99)

[Schneider & Raue 98]
Wolfgang Schneider und Paul-Josef Raue
Handbuch des Journalismus
Reinbek: Rowohlt Verlag GmbH, 1998

[Schulmeister 96]
Grundlagen hypermedialer Lernsysteme: Theorie – Didaktik - Design
Rolf Schulmeister
Bonn; Paris [u.a.]: Addison-Wesley, 1996

[Schwabe & Rossi 95]
Daniel Schwabe und Gustavo Rossi
The Object-Oriented Hypermedia Design Model
In: Communications of the ACM, August 1995, Vol. 38, No. 8, S. 45-46

[Schwabe & De Almeida Pontes 98]
Daniel Schwabe und Rita de Almeida Pontes
OOHDM-WEB: Rapid Prototyping of Hypermedia Applications in the WWW
August 1998
http://www.inf.puc-rio.br/~schwabe/papers/MCC-08-98.pdf.gz (1.6.2000)

[Seitz 89]
Fritz Seitz
Farbe und Entwurf
In: Visuelle Kommunikation – Ein Design-Handbuch (S. 141-176)
Anton Stankowski und Karl Duschek (Hrsg.)
Berlin: Dietrich Reimer Verlag, 1989

[Shedroff 99]
Nathan Shedroff
Information Interaction Design: A Unified Field Theory of Design
In: Information Design (S. 267-292)
Robert Jacobsen (Hrsg.)
Cambridge, Massachusetts: The MIT Press, 1999

[Shiple 98]
John Shiple
Information Architecture Tutorial
1998
http://hotwired.lycos.com/webmonkey/design/site_building/tutorials/tutorial1.html (25.8.2000)

[Shneiderman & Kearsley 89]
Ben Shneiderman und Greg Kearsley
Hypertext Hands-On! An introduction to a new way of organizing and accessing information
Reading, Massachusetts: Addison-Wesley, 1989

[Shneiderman 98]
Ben Shneiderman
Designing the User Interface - Strategies for Effective Human-Computer-Interaction (Third Edition)
Reading, Massachusetts: Addison-Wesley, 1998

[Siegel 97]
David Siegel
The Web Is Ruined and I ruined it
Oktober 1997
http://www.xml.com/pub/w3j/s1.people.html (21.3.2000)

[Sisson 99]
Derek Sisson
Getting to Know Your Audience
Juni 1999
http://www.philosophe.com/audience/understanding_users.html (20.11.99)

[Snyder 96]
Carolyn Snyder
Using Paper Prototypes to Manage Risk
1996
http://world.std.com/~uieweb/paper.htm (18.04.2000)

[Strothotte & Strothotte 97]
Christine Strothotte und Thomas Strothotte
Seeing Between the Pixels. Pictures in Interactive Systems.
Berlin: Springer-Verlag, 1997

[Sullivan 96a]
Terry Sullivan
User Testing Techniques - A Reader-Friendliness Checklist
September 1996
http://www.pantos.org/atw/35317.html (25.8.2000)

[Sullivan 96b]
Terry Sullivan
User Testing Techniques – Site Reviews
August 1996
http://www.pantos.org/atw/35283.html (25.8.2000)

[Turner 99]
Margaret Turner
Fundamental Web Design Principles?
1999
http://ausweb.scu.edu.au/aw99/papers/turner/paper.html (5.11.99)

[Tenny 96]
Matthew J. Tenny
Ameritech Graphical User Interface Standards and Design Guidelines: User-Centered Design
Januar 1996
http://www.ameritech.com/corporate/testtown/library/standard/std-guix.html (30.5.2000)

[Tullis 98]
Thomas S. Tullis
A Method for Evaluating Web Page Design Concepts
In: Proceedings of the conference on CHI 98 summary, April 1998, S. 323-324

[Vogt 99]
Thomas Vogt
The use of guidelines for designing user interfaces in commercial software development
Studienarbeit, Universität Hamburg, Fachbereich Informatik, 1999

[Ward & Codrai 98]
Robert Ward und Chris Codrai
Designing information for the World-Wide Web
1998
http://www.istc.org.uk/design.htm (28.11.99)

[Weinreich 97]
Harald Weinreich
Ergonomie von Hypertext-Systemen und das World Wide Web – Evaluation und Überarbeitung des WWW-Informationssystems des Fachbereichs Informatik
Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 1997

[Wildman 95]
Daniel Wildman
Getting the Most from Paired-User Testing
1995
http://www.acm.org/interactions/vol2no3/columns/mandt/methtool.htm (18.4.2000)

[Wood 00]
Lauren Wood (Hrsg.)
Document Object Model (DOM) Level 2 Specification
2000
http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510 (17.8.2000)

[Yu, Prabhu & Neale 98]
Jack J. Yu, Prasad V. Prabhu und Wayne C. Neale
A User-Centered Approach to Designing a New Top-Level Structure for a Large and Diverse Corporate Web Site
1998
http://www.research.att.com/conf/hfweb/proceedings/yu/index.html (17.8.2000)

[Züllighoven 93]
Heinz Züllighoven
Grundzüge der Informatik A1, Teil 1
Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, 1993

[Züllighoven et al. 98]
Heinz Züllighoven (Hrsg.)
Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz
Heidelberg: dpunkt-Verlag, 1998

Ich versichere hiermit, die vorliegende Arbeit selbstständig und unter ausschließlicher Zuhilfenahme der im beigefügten Verzeichnis angegebenen Hilfsmittel erstellt zu haben.

Hamburg, d. 16.12.2000

(Peter Kaul)