Web-Präsenz-Management im Unternehmen
Entwicklung und Einsatz eines Java-basierten Online-Redaktionssystems

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften
am Fachbereich IV der Universität Trier

vorgelegt von

Diplom-Physiker
Andreas Heuer

September 2001

Gutachter: Prof. Dr. sc. Christoph Meinel (Universität Trier)

Prof. Dr. Max Mühlhäuser (Technische Universität Darmstadt)

Datum der Disputation: 26. Februar 2002

Alles Wissen und alles Vermehren unseres Wissens endet nicht mit einem Schlußpunkt, sondern mit einem Fragezeichen.

Hermann Hesse (1877-1962)

Keine Schuld ist dringender,
als die, Dank zu sagen.

Marcus Tullius Cicero (106-43)

Danksagung

Diese Arbeit ist allen Personen gewidmet, die mich auf meinem bisherigen Weg ermutigt und unterstützt haben.

Mich an die Worte Ciceros haltend, möchte ich mich an erster Stelle bei Prof. Dr. sc. Christoph Meinel, dem Direktor des Instituts für Telematik, für die wissenschaftliche Begleitung der Dissertation sowie die Bereitstellung des äußerst fruchtbaren Umfelds in dem diese Arbeit gedeihen konnte bedanken.

Ohne die kameradschaftliche Unterstützung von meinen langjährigen Kollegen Frank Losemann, Dr. Ernst-Georg Haffner und Uwe Roth wäre die Arbeit am Institut nur halb so interessant gewesen. Mein Dank gilt ihnen für die Basis, die durch die zahlreichen, immer spannenden wissenschaftlichen Diskussionen und die erfolgreiche gemeinsame Projektarbeit für diese Arbeit geschaffen wurde.

Für die zahlreichen grammatikalischen und stilistischen Verbesserungen, die sie aufopferungsvoll dieser Arbeit angedeihen ließ sowie die nicht unerhebliche Motivation zur Selbstdisziplin, die sie mir beigebracht hat, möchte ich mich ganz lieb bei Lasia Bloß bedanken.

Ganz herzlich bedanken darf ich mich weiterhin bei Stefan Dewald, ohne dessen tatkräftige Unterstützung die Entwicklung des in dieser Arbeit beschriebenen Systems bei weitem nicht so zügig vorangegangen wäre.

Zu Dank verpflichtet bin ich des weiteren dem ganzen Stab an Institutsmitarbeitern und dabei allen voran Dr. Thomas Engel, dem stellvertretenden Institutsdirektor, da erst ihre freundliche und unauffällige Unterstützung mein Promotionsvorhaben möglich machte.

Ein herzlicher Dank geht im weiteren auch an meine ehemaligen Mitstreiter, die mit mir viel Zeit im Institut und auch außerhalb verbracht haben: Kai Siemonsen, Roland Graßmann, Petra Zimmermann und Claus Schröter.

Meinen Eltern und meinem Bruder, bin ich ganz besonders dankbar für die Unterstützung, ihre Kraft und ihre Liebe, die mich all die Jahre durch mein Studium begleitet und mich in meiner Entscheidung für diese Dissertation bestärkt haben.

Abstract

Die eigene Web-Präsenz hat sich für Unternehmen zu einem nicht mehr wegzudenkenden Instrument, einer flexiblen Schnittstelle zum Kunden, entwickelt. Mit ihr wird eine global erreichbare Anlaufstelle offeriert: Die Chancen der Online-Techniken nutzend ist dem Unternehmen mit seiner Web-Präsenz ein Medium für die „klassische“ Werbung, darüber hinaus ein Direkt-Marketing- und Dialog-Medium sowie weiterhin ein eigenes Marktforschungs-Instrument an die Hand gegeben. Dessen Entwicklung und speziell sein Betrieb stellen für ein Unternehmen in verschiedenen Hinsichten eine ernstzunehmende Herausforderung dar. Aus organisatorischer Sicht ist hier die nicht einfach zu realisierende Integration der Web-Präsenz in das Unternehmen anzustreben: Nicht einem isoliert arbeitenden, technikverliebten Mitarbeiter, dem „Webmaster“, obliegt die Pflege der Web-Präsenz als Allein-Zuständigen, sondern entsprechend ihren Zuständigkeiten übernehmen anteilig die in den Fachabteilungen des Unternehmens tätigen Arbeitskräfte diese Aufgabe. Aus technischer Sicht stellt eine moderne Web-Präsenz ein umfangreiches, und bei hoher Dynamik entsprechend komplexes Produkt dar, für dessen Realisierung nicht unerhebliche Anstrengungen unternommen werden müssen – insbesondere dann, wenn die zugrundeliegende Technik den organisatorischen Anforderungen innerhalb des Unternehmens gerecht werden soll.

Bedingt durch die hohe Innovationsrate im Umfeld der Internet-Technologie befinden sich die Anforderungen an die Betreiber einer Web-Präsenz in einem steten Wandel und erschweren damit den Unternehmen den Einstieg in dieses Medium. Gerade die in vielen Fällen in den Unternehmen noch eingesetzten proprietären Lösungen erfordern aus technischer Sicht einen hohen Integrationsaufwand, wenn die darin vorgehaltenen Informationen für die Web-Präsenz automatisiert einer Zweitverwendung zugeführt werden sollen. Über die zusätzliche Nutzung der Unternehmens-Web-Präsenz bzw. des Systems, mit dem sie erstellt und verwaltet wird, für die unternehmensinterne Arbeit, d.h. für das Intranet, kann der für das Unternehmen entstehende Aufwand relativiert werden.

In das Bild des Lebenszyklus' einer modernen Web-Präsenz eingebettet, steht die Konzeption und Entwicklung eines an die im Umfeld des Web-Präsenz-Managements in Unternehmen vorzufindenden Randbedingungen flexibel anpassbaren Online-Redaktionssystems im Vordergrund dieser Arbeit. Zentraler Bestandteil seiner Konzeption ist die Idee, ein sich aus derart vielen separaten Elementen zusammensetzendes, komplexes Gesamtdokument wie eine Unternehmens-Web-Präsenz durch eine wachsende Mitarbeiterzahl – direkt von deren jeweiligen Arbeitsplätzen aus – gemeinsam aufzubauen und zu pflegen.

Der Praxiseinsatz des bedarfsgerecht in seinen endgültigen Ausprägungen auf die speziellen Belange der Unternehmen maßgeschneiderten Systems begründet die konsequente Nutzung von Internet-Technologien sowie die auf offenen Standards basierende Implementation durch flexible plattformunabhängige Installation und einfache Integration.

Im Anschluss an ein einleitendes Kapitel beschreibt diese Arbeit ausgehend von der Problemstellung sowie technischen Grundlagen die Konzeption, Implementation und den Einsatz des webbasierten Online-Redaktionssystems bevor eine Einordnung und schließlich die Zu-

sammenfassung den Abschluss dieser Arbeit bilden.

Abstract

Any specific website as flexible interface to the customer has nowadays evolved to a valuable tool for business companies that deserves increasing attention - both from the scientific and the practical world. It offers a borderless global communication network: Utilizing the possibilities of modern online techniques through a website, a company is simultaneously given a medium for classical advertisement and direct marketing as well as an interactive instrument to be used as an individualized market analysis tool. The development and, in particular, the operation of websites represent a far reaching challenge for a corporation, and this in several regards: from an organizational point of view, the integration of a website into the general framework of a given business environment and its specific internal structure should aim at avoiding an insulatedly operating, technically oriented associate - the webmaster - being exclusively responsable for the maintenance of the website. Instead, each collaborator working in the specialized departments of the undertaking should proportionately share this webmaster-function according to his/her competences. From a technical point of view, a modern website is an extensive and, due to its extraordinarily high dynamics, an extremely complex product. In the process of its implementation, significant efforts have to be pursued, in particular if the underlying technique attempts to meet the given organizational requirements within an existing enterprise.

Due to the high rate of innovation in the surrounding field of the internet technology, the demands for operators of a website are subject to a permanent change and, hence, render it more and more difficult to trespass the entrance barriers into an efficient utilization of the online-medium.

Speaking in technical terms, specifically proprietary solutions, still used in numerous undertakings, ask for a high investment expenditure in order to achieve a useful integration of the internet into an existing business environment, e.g. information processed for the website should, generally speaking, automatically be available for secondary use. Through additional use of company websites or systems with which they are created and administered for enterprise-internal purposes, i.e. the intranet, development expenses can be interrelated and therefore kept reasonable in economic terms.

Enclosed in the scheme of the life cycle of a modern website, where one finds the conception and development of an adjustable system of online editorship which stands at the core of this doctoral thesis. Fundamental requisites in terms of marginal conditions that can be found in the surrounding fields of the website management in corporations are the adaptability and flexibility of such a system of online editorship. Central constituent of its conception is the idea to assemble and maintain a complex final document, synthesized and structured by numerous independent items through the cooperation of an increasing number of employees working directly at their respective workstations. The practical application of the introduced system, custom-made in its final developments according to the special needs of a given enterprise, justifies the consistent use of internet technologies as well as the implementation which is based on open standards through a flexible platform-independent installation and

simple integration into the given business environment.

Following an introductory chapter, this work outlines the conception, implementation and application of the web-based system of online editorship JDaphne on the basis of a definition of the principal problem as well as its technical bases. This study concludes with a product classification and a final summary.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung und Überblick

Während noch vor nicht ganz zehn Jahren nur Eingeweihte den Begriff „Web-Präsenz“ kannten und wussten, welches Konzept sich dahinter verbirgt, ist er heute aus dem täglichen Leben nicht mehr wegzudenken. Sowohl privat als auch beruflich bzw. geschäftlich wird das World Wide Web (WWW, vgl. [BLCGP92]) in großem Maße genutzt. In der kurzen Zeit, die seit der Entwicklung des WWW zum einfacheren wissenschaftlichen Informationsaustausch über das Internet vergangen ist, hat dieses einen primär durch sein Wachstum begründeten Wandel erfahren. Zum einen ist die bloße Anzahl der online verfügbaren Web-Präsenzen mit – nach Angaben der *Internet-Agentur Netcraft* – 31 299 592 Web-Präsenzen im Juli 2001 [Net01] geradezu explodiert, zum anderen hat sich die Gestalt der Web-Präsenzen an sich dramatisch verändert. Die in den letzten Jahren stark fortgeschrittene Kommerzialisierung des WWW mit den damit verbundenen Marketing-Aufwendungen durch die Unternehmen hat die Mutation des ursprünglich wissenschaftlich nüchternen, fachlichen Mediums zu einer mittlerweile sehr „bunten“ und unterhaltsamen, aber nichtsdestotrotz hochfunktionalen Multimedia-Landschaft forciert. Die Simplizität, mit der Informationen aus dem WWW abgerufen werden können, die zudem noch in vielen Fällen „hingebungsvoll“ aufbereitet sind, hat dem WWW einen fest etablierten Platz in der globalen Medienlandschaft verschafft [Ber99]. Das umfangreiche Angebot der auf diese Weise über das WWW international verfügbaren Informationsquellen hat bei dem Benutzer zu einem Anspruchsdenken geführt: Er erwartet hochwertig aufbereitete, personalisierte Inhalte von hoher Qualität und Aktualität. Werden diese ihm von der ersten besuchten Web-Präsenz nicht, zu langsam oder mangelhaft zusammengestellt präsentiert, so stellt es für ihn keinen sonderlichen Aufwand dar, ein konkurrierendes Angebot wahrzunehmen.

Für Unternehmen birgt die eigene Web-Präsenz die Eigenschaften eines multifunktionalen Werkzeugs: Die Chancen der Online-Techniken nutzend ist dem Unternehmen mit seiner Web-Präsenz ein Medium für die „klassische“ Werbung, darüberhinaus ein Direkt-Marketing- und Dialog-Medium sowie weiterhin ein eigenes Marktforschungs-Instrument an die Hand gegeben (vgl. [Ber01]). Nicht nur für Unternehmen der „New Economy“, die von den Auswirkungen des WWW in ihrer wirtschaftlichen Zukunft besonders betroffen sind, leitet sich aus diesem Anspruch an die eigene Web-Präsenz eine aufgrund der globalen Auswirkungen im internationalen Medium „Internet“ überlebenswichtige Herausforderung ab. Dieser gerecht zu werden, erfordert umfassende Anstrengungen von Seiten des Unternehmens; bei dem Aufbau des notwendigen „Know-How“ als Basiswissen für die Entwicklung einer eigenen Web-Präsenz sind neben technischen vor allem organisatorische Aspekte zu berücksichtigen. Bedingt durch die hohe Innovationsrate im Umfeld der Internet-Technologie befinden sich die Anforderungen an die Betreiber einer Web-Präsenz in einem steten Wandel und erschweren damit den Unternehmen den Einstieg in dieses Medium. Gerade die in vielen Fällen noch in den Unternehmen

eingesetzten proprietären Lösungen erfordern aus technischer Sicht einen hohen Integrationsaufwand, wenn die darin vorgehaltenen Informationen für die Web-Präsenz automatisiert einer Zweitverwendung zugeführt werden sollen.

Wie eine Web-Präsenz von einem Unternehmen letztendlich betrieben wird, hängt von vielen Faktoren ab; ausschlaggebend sind hier die mit ihr verbundenen Ziele – sowohl hinsichtlich ihrer Außenwirkung als auch bezüglich ihrer Integration in das Unternehmen. Eine immer bedeutsamere Rolle bei der Einbindung spielt dabei die Nutzung der Web-Präsenz auch für interne Zwecke, im „Intranet“ – ein Mehrwert, der den anfallenden Aufwand relativiert.

1.1 Web-Präsenz-Verwaltung

Mit der historischen Entwicklung haben sich gemäß den gestiegenen Anforderungen an eine Web-Präsenz in Anbetracht ihrer gewachsenen Bedeutung auch die Modelle für ihre Verwaltung geändert. Die umfassenden Anforderungen, die an eine Web-Präsenz gestellt werden, in Kombination mit den Zielen, die ein Unternehmen mit einer eigenen Web-Präsenz verbindet, bedingen die Faktoren, die für die Auswahl und die Umsetzung eines Modells zum Tragen kommen. Aus organisatorischer Sicht ist hier die nicht einfach zu realisierende Integration der Web-Präsenz in das Unternehmen anzustreben: Nicht einem isoliert arbeitenden, technikverliebten Mitarbeiter, dem „Webmaster“, obliegt die Pflege der Web-Präsenz als Allein-Zuständigen, sondern entsprechend ihren Zuständigkeiten übernehmen anteilig die in den Fachabteilungen des Unternehmens tätigen Arbeitskräfte diese Aufgabe.

1.1.1 Unternehmensziele für eine Web-Präsenz

Anhand der Ziele, die ein Unternehmen für die Ausrichtung seiner Web-Präsenz vorgibt, wird auf der einen Seite ihre Ausprägung und auf der anderen Seite ihre Verwaltung determiniert. Grundsätzlich lassen sich bei der Ausprägung einer Unternehmens-Web-Präsenz zwei verschiedene Ansätze differenzieren: die Web-Präsenz als Marketing-Instrument oder als „Profit-Center“. Während im ersten Fall Marketing-Aspekte [Ren00] im Vordergrund stehen und der finanzielle Aufwand sich nur schwer in Relation zu dem betriebswirtschaftlichen Nutzen setzen lässt, ist im zweiten Fall der mit der Web-Präsenz erzielte Umsatz direkt messbar und somit eine Gewinn- und Verlustrechnung möglich.

Die im Folgenden aufgeführten Ziele sind repräsentativ für die vorrangig aus dem Banken- und Versicherungsumfeld stammenden Projektpartner, mit denen zusammen das Institut für Telematik Web-Präsenzen aufgebaut hat, gelten jedoch für den Großteil der sich im WWW engagierenden Unternehmen analog.

1.1.1.1 Präsenz zeigen

Die für alle Unternehmen primär mit der eigenen Web-Präsenz verfolgte Zielsetzung ist die damit verbundene Inanspruchnahme eines Platzes in der virtuellen Welt des WWW und daraus resultierend die Erhöhung des Bekanntheitsgrades und die Positionierung als zukunftsträchtiges Unternehmen. Auch wenn für viele Unternehmen der „New Economy“ dieser virtuelle Standort von deutlich größerer Bedeutung ist als für die Unternehmen der „Old Economy“, ist seine Relevanz, gerade in Anbetracht der internationalen Konkurrenz, in jedem Fall eine hinreichende Motivation für eine eigene Web-Präsenz. Die „Präsenz“ im WWW ist hierbei als ein mehrstufiger Prozess zu sehen: In einem ersten Schritt wird mit dem Erwerb eines geeigneten Domain-Namens verhindert, dass dieser durch andere genutzt werden kann. In einem zweiten Schritt wird eine Web-Präsenz mit einem Grundbestand an Inhalten auf-

gebaut. Basierend auf den so gewonnenen Erfahrungen wird das Angebot modifiziert, d.h. umgeschichtet und ergänzt.

1.1.1.2 Kontaktaufnahme

Sekundäres Ziel bei dem Betrieb einer Web-Präsenz ist die durch sie bedingte Erhöhung der Neukunden-Anfragen innerhalb der kommenden Jahre. Dieser Aspekt wird insbesondere realisiert durch die Bereitstellung von Kontaktinformationen: der direkte Zugriff auf Übersichten von Ansprechpartnern, E-Mail-Adressen und Telefonnummern zu den einzelnen Unternehmensbereichen, Abteilungen und Produkten erleichtert den Kunden die direkte Kontaktaufnahme, oft von Beginn an, z.B. wenn ein Foto des Ansprechpartners im WWW veröffentlicht ist, auf einer persönlicheren Ebene. Für den Kunden entfällt auf diese Weise in vielen Fällen der sonst mühsame Weg über das Telefon von einer zentralen Vermittlungsstelle über diverse zwischengeschaltete Gesprächspartner bis hin zu dem Ansprechpartner für das ihn interessierende Produkt. Die direkten Wege im WWW helfen somit, die Distanz zwischen dem Unternehmen und seinen Kunden zu verringern.

1.1.1.3 Nutzung als externes und internes Informations-System

Bereits die mit der Web-Präsenz angestrebte Kontaktaufnahme nutzt den Informations-System-Aspekt einer Web-Präsenz. Ziel der Unternehmens-Web-Präsenz ist es, Informationen bereitzustellen, sowohl nach außen für die Kunden als auch nach innen für die im Unternehmen Beschäftigten. In Abhängigkeit von der Branche werden hier vorrangig Produktinformationen dargeboten. Sie erlauben den Kunden beispielsweise, vor der Kontaktaufnahme auf einem konventionellen Weg, einem Telefongespräch, sich zu Hause in der gewohnten Umgebung mit den Produkten auseinanderzusetzen. Über dynamische Abfragen können die Kunden in spezifische Daten zu den Produkten – zum Teil direkt aus dem internen Datenbestand des Unternehmens – Einsicht nehmen. Ausgestattet mit diesem Wissen sind anschließend bei einer im Normalfall telefonischen Kontaktaufnahme produktive Verkaufsgespräche möglich. Ein Mehrwert entsteht für das Unternehmen auch durch die interne Nutzung der Web-Präsenz. Bei Bedarf modifiziert und erweitert durch einen internen Datenbestand können Arbeitsabläufe im Unternehmen effizienter gestaltet werden.

Eine Nutzung einer Web-Präsenz als Informations-System ergibt sich insbesondere durch den darüber abgewickelten Kunden-Support. Durch die Bereitstellung von Antworten auf häufig in Zusammenhang mit den Produkten des Unternehmens auftretende Fragen können viele ansonsten telefonisch auflaufende Anfragen dank der vierundzwanzig Stunden am Tag verfügbaren Web-Präsenz unabhängig von den Arbeitszeiten der Support-Mitarbeiter im Vorfeld vermieden werden. Diese können sich, da sich die Benutzer anhand der Web-Präsenz „Standard-Fragen“ selbsttätig beantworten können, vollständig auf die Beantwortung kritischer Fragen konzentrieren.

1.1.1.4 Geschäftliche Transaktionen

Im Gegensatz zu den als Marketing-Instrumenten oder als Werbe-Medium betriebenen Web-Präsenzen, die im Allgemeinen nur Kosten verursachen, jedoch keine eigenständigen Einnahmen erzielen und daher als „Cost-Center“ betrachtet werden müssen, wird mit den auf „E-Commerce“, d.h. als Plattform für geschäftliche Transaktionen, ausgerichteten Web-Präsenzen direkt versucht, durch über die Web-Präsenz generierte Umsätze Gewinne zu erzielen (“Profit-Center“) [Bra00].

Zumindest in der heutigen Zeit wird noch versucht – bedingt durch die bis dato verfügbaren integrierenden Technologien und dabei vor allem aus Performance-Gründen – die Web-Präsenzen in transaktionelle Bereiche mit einer hohen Sicherheitsstufe sowie „konventionelle“ Bereiche aufzuteilen und auf diese Weise neben dem Profit- auch den Marketing-Aspekten gerecht zu werden.

1.1.1.5 Marktforschung

Mit wenigen anderen Medien ist der Kontakt eines Unternehmens zu seinen Kunden so eng und mit so direkter Rückkopplung versehen, wie mit der eigenen Web-Präsenz. Von daher liegt es nahe, die aus der Nutzung der Web-Präsenz durch den Kunden gewonnenen Informationen für die Marktforschung einzusetzen. Die Auswertung der über die Web-Präsenz generierten Log-Dateien lässt nicht nur Aussagen darüber zu, wie die Web-Präsenz von den Kunden angenommen wird. Darüber hinaus erlaubt sie bei geeigneter Auswahl der Inhalte Rückschlüsse auf die Kundeninteressen und -vorlieben. Oft in Kombination mit konkreten Kundenprofilen (vgl. [Sch01]) ist ein generelles, entweder zentrales oder zumindest nebengeordnetes Ziel bei dem Aufbau einer Web-Präsenz die Erhebung von Marktforschungsdaten über ihre Benutzer.

1.1.2 Modelle der Web-Präsenz-Verwaltung

Zum Teil beeinflusst von den mit der Web-Präsenz verbundenen Zielen haben sich bei den Unternehmen bislang zwei gängige Modelle für den Aufbau, die Pflege, den Betrieb und die Verwaltung etabliert, die mit Einschränkungen häufig auch in entsprechenden Abstufungen kombiniert vorzufinden sind: die Web-Präsenz wird in ihrer Verwaltung entweder an eine Agentur ausgelagert oder direkt in das Unternehmen integriert. Letzteres geschieht i.d.R. unter Zukauf von externer Unterstützung in technischen und konzeptionellen Fragen durch Agenturen wie z.B. Design-Agenturen oder Unternehmensberatungen. Beide Varianten haben jeweils Vor- und Nachteile, die in den folgenden beiden Abschnitten beleuchtet werden; der allgemein zu beobachtende Trend geht aber definitiv in die Richtung der Integrationslösung.

1.1.2.1 Auslagerung an eine Agentur

Die Auslagerung der Web-Präsenz-Entwicklung und -Pflege an eine Agentur ist ein in der Vergangenheit häufig gewählter Weg. Bei dieser Variante stimmt die Marketing-Abteilung des Unternehmens mit der beauftragten Agentur den Internet-Auftritt ab und liefert die für die Entwicklung notwendigen inhaltlichen Eingaben. Die Agentur kümmert sich um die technischen und grafischen Details, nimmt aber häufig auch Einfluss auf die inhaltliche Architektur der Web-Präsenz. Dies führt zu professionellen, in sich stimmigen Internet-Auftritten, hat bei näherer Ansicht jedoch oft den Makel der inhaltlichen Leere und bietet häufig nicht den notwendigen Aktualitätsgrad.

Der Weg von den im Unternehmen verfügbaren Informationen über die „Engstelle“ Marketing-Abteilung und die Internet-Agentur auf die Web-Präsenz ist zum einen allein aufgrund der großen Abstimmungsarbeit sehr lang, zum anderen sind in solche „runden“ Web-Präsenzen neue Inhalte nicht einfach zu integrieren, da sie in ihrer gestalterischen Ausprägung vielfach konsequent auf die initialen Inhalte ausgerichtet wurden. Für die Präsentation anderer Inhalte wird nicht selten ein vollständiges Re-Design des Internet-Auftritts erforderlich.

Bei in solchen Auftragsarbeiten entstandenen Web-Präsenzen steht generell das Design im Vordergrund; sie dienen vorrangig der Werbung für ein bestimmtes Produkt und sind in ihrem Umfang entsprechend darauf abgestimmt.

Vorteilhaft bei der Auslagerung der Web-Präsenz an eine Agentur ist die hohe Transparenz der für den Betrieb der Web-Präsenz anfallenden Kosten, denn gerade zu Beginn der Umorientierungsphase von Unternehmen, in der die Möglichkeiten des neuen Mediums ausgelotet werden sollen, stellen die klar erfassbaren und normalerweise fest begrenzten Kosten ein wichtiges Argument für die Erprobung einer Unternehmens-Internet-Präsenz dar.

Bei ihrer erfolgreichen Platzierung mit entsprechend positiven Ergebnissen für das Unternehmen können, basierend auf den Erkenntnissen aus der von der Agentur betriebenen Web-Präsenz, Prognosen für die anfallenden Arbeiten, die Kosten und die Notwendigkeit einer Integration in das Unternehmen veranschlagt werden.

1.1.2.2 Integration in das Unternehmen

Mit der Integration der Web-Präsenz verbindet sich die Anforderung an den Internet-Auftritt, neben den bereits durch den Betrieb über eine Agentur (vielleicht sogar besser und effizienter) erreichbaren Marketing-Aspekten auch tiefgreifenderen inhaltlichen Aspekten (Informations-System) Rechnung zu tragen. Für die Bereitstellung inhaltlicher Tiefe ist das Unternehmen aus organisatorischer Sicht gut beraten, große Teile der Web-Präsenz-Verwaltung in den normalen Tagesablauf seiner Mitarbeiter zu integrieren.

Wie einleitend beschrieben, ist der für die Integration zu betreibende Aufwand, gegebenenfalls erweitert durch ein Intranet-Angebot, erheblich, da von der Integration sowohl die technische Infrastruktur als auch das Personal betroffen sind. Bei ihrer erfolgreichen Integration sind die Weichen für eine lebendige Web-Präsenz gestellt, und es wird zudem noch der Mehrwert als interne Informationsquelle generiert.

Im Gegensatz zu der vorhergehenden Variante der Auslagerung lassen sich die bei der Integration der Web-Präsenz anfallenden Kosten nur sehr schwer erfassen. Während sich die Investitionen in Soft- und Hardware noch vergleichsweise klar berechnen lassen, bleiben die Aufwendungen für die benötigte Aus- und Weiterbildung der Mitarbeiter eine nur schwer erfassbare Größe. Deren Berechnung fällt um so schwerer, je stärker die Integration der Web-Präsenz in das Unternehmen umgesetzt wird, denn mit dem Wachsen der Verzahnung verschwimmen die Grenzen zwischen der Arbeit an der Web-Präsenz und den bis dato ohnehin angefallenen Arbeiten. Beispielsweise obliegt es den Mitarbeitern unabhängig von der Web-Präsenz, Produktinformationen, z.B. für Broschüren, zusammenzustellen. Der Mehraufwand der Publizierung dieser Informationen auf der Web-Präsenz ist bei dem Einsatz entsprechender Werkzeuge gering, der Mehrwert jedoch enorm. Während die beispielhaft erwähnte Broschüre nur in größeren zeitlichen Abständen erstellt wird, kann durch eine entsprechend gepflegte Web-Präsenz permanent eine aktuelle Version der darin zu publizierenden Informationen bereitgestellt und auf diese Weise die Zwischenzeit überbrückt werden. Im Idealfall wird schließlich die Web-Präsenz als aktuelle Basis für die Printausgabe der Broschüre genutzt.

1.1.3 Werkzeuge zur Verwaltung von Web-Präsenzen

Die beschriebenen unterschiedlichen Vorgehensweisen und Zielsetzungen bei der Erstellung und Verwaltung einer Web-Präsenz legen es nahe, in den jeweiligen Fällen unterschiedliche Werkzeuge, ausgewählt anhand der spezifischen Anforderungen, einzusetzen. In einer Agentur beispielsweise wird in den meisten Fällen die gestalterische Komponente der Web-Präsenz im Vordergrund stehen; die Werkzeuge mit denen sie eine Web-Präsenz aufbaut, lehnen sich in ihrem Ergebnis dementsprechend an Desktop-Publishing-Programme und Hypermedia-Authoring-Tools an. Um eine bestimmte Bildschirmaufteilung zu erzielen, werden pixelgenaue

Platzierungen von Designelementen vorgenommen, mit der Folge hoher manueller Einflussnahme bei der Gestaltung jedes einzelnen Dokuments: die Entwürfe orientieren sich an einer vorgegebenen Bildschirmauflösung, manuell werden gezielte Zeilenumbrüche erzeugt, werbewirksame Effekte, erzielt durch dynamische multimediale Elemente, werden in hohem Maße eingesetzt. Charakteristisch bei der Web-Präsenz-Verwaltung über Agenturen sind die kleinen und zudem im Umgang mit den Werkzeugen hochqualifizierten Teams. Durch den täglichen Umgang geübt und unterstützt durch entsprechende Schulungen, sind die Mitarbeiter der Agentur in der Lage, die dort genutzten Editor- und Design-Programme sachgerecht und effizient einzusetzen, insbesondere auch dann, wenn komplexere Aufgaben anstehen.

Konträr gestaltet sich hingegen die Arbeit an der Web-Präsenz bei ihrer Integration in das Unternehmen. Hier sind zum einen bei den Mitarbeitern andere, aus der Arbeit in den Fachabteilungen motivierte und nicht auf das grafische Design ausgerichtete Qualifikationen vorhanden, zum anderen ist die Zahl der an der Pflege der Web-Präsenz beteiligten Personen deutlich größer, da jeder einzelne nur anteilig mit ihr beschäftigt ist. Beide Faktoren zusammen bedingen andere Anforderungen an das zu nutzende Werkzeug: einfache Bedienbarkeit, verteiltes Arbeiten und Mehrbenutzerfähigkeit. Bei der letztendlichen Gestaltung der publizierten Dokumente wird auf Grund der hohen Dokumentenzahl verstärkt den Automatismen des Werkzeugs vertraut; das Design der Web-Präsenz muss sich der Funktionalität in einem viel größeren Maße unterordnen, da allein aus Zeitgründen keine manuelle Nachbearbeitung, eine Art Fine-Tuning, der fertigen Web-Präsenz möglich ist.

Während Kapitel sieben einen umfassenden Überblick über Konzepte und Systeme liefert (vgl. Seite 173), werden im Folgenden die beiden grundsätzlichen Varianten zusammen mit einer aus ihnen konzipierten Synthese einführend besprochen: die HTML-Editoren/Multimedia-Authoring-Tools, die Content-Management-Systeme und schließlich die Web-Präsenz-Content-Management-Systeme.

1.1.3.1 HTML-Editoren/Multimedia-Authoring-Tools

Für die Erstellung und den Aufbau von kleinen, gestalterisch anspruchsvollen Web-Präsenzen werden HTML-Editoren und Programme für die Erstellung der multimedialen „Effekt“-Elemente eingesetzt (vgl. Seite 175); spezielle Werkzeuge haben diese beiden Komponenten zusammengebracht und ermöglichen auf diese Weise die integrierte Erstellung gestalterisch äußerst ansprechender Internet-Auftritte.

Die Arbeit mit solchen Werkzeugen erfolgt vorrangig durch einzelne Benutzer oder kleine Teams. Während diese Programme bezüglich der Gestaltung einer Web-Präsenz schwer zu überbieten sind, werden von ihnen Mechanismen zur Verwaltung der Web-Präsenz nur eingeschränkt angeboten. Insbesondere bei wachsenden Dokumentenzahlen und damit verbunden vielen an der Verwaltung der Web-Präsenz beteiligten Personen kommen ihre Schwächen zum Vorschein: Dokumenten-Workflows und rollenbasierter Zugriff beispielsweise sind, wenn überhaupt vorhanden, nur rudimentär ausgeprägt.

1.1.3.2 Content-Management

Gerade die zuletzt genannten Mechanismen werden hingegen von Content-Management-Systemen (vgl. Seite 181) als Basisfunktionen geliefert. Auf der Grundlage medienneutraler Formate werden mit ihnen Inhalte verteilt erstellt und verwaltet; die Ausgabe der ihnen bekannten Informationen kann in vielen verschiedenen Formaten und Medien, Kontexten und Kombinationen erfolgen.

Grundsätzlich ist demnach auch die Möglichkeit einer Publikation auf eine Web-Präsenz gegeben, und mit dem wachsenden Bedarf an entsprechenden Werkzeugen gehen die Anbie-

ter von Content-Management-Lösungen dazu über, für ihre Produkte zusätzliche, auf den Web-Export ausgelegte Module bereitzustellen. Für ein Unternehmen, das bereits über ein solches Content-Management-System verfügt, ist die Anschaffung eines derartigen Moduls, aus ganzheitlicher Sicht die effizienteste Lösung. Für ein Unternehmen, das noch nicht über ein so umfangreiches und komplexes System verfügt, rechnet sich die Neuinstallation zum Zweck des Web-Präsenz-Managements nur bedingt im Vergleich zu den speziell für diesen Zweck entworfenen „schlankeren" Web-Präsenz-Content-Management-Systemen.

1.1.3.3 Web-Präsenz-Content-Management

Eine Mischlösung aus oben beschriebenen „Editor"-Programmen und Content-Management-Systemen mit einem Web-Export-Modul stellen die als Web-Präsenz-Content-Management-Systeme (WCMS, vgl. Seite 182) bezeichneten Produkte dar. Anders als die „Editor"-Programme verfügen sie über Dokumenten-Workflows, sind entsprechend aber in dem Design der damit erstellten Web-Präsenzen aufgrund ihrer höheren Automatisierung nicht so flexibel in der Ausgestaltung einzelner Dokumente. Ihre medienneutralen Content-Management-Fähigkeiten kommen im Normalfall aufgrund ihrer konkreten Ausrichtung auf die Verwaltung von Web-Präsenzen und damit der Festlegung auf ein bevorzugtes Ausgabeformat nicht an die von den umfassenderen Content-Management-Systemen heran.

Während bereits von der Konzeption her bei den „Editor"-Programmen eine starke Integration von Inhalt und finaler Gestaltung der fertigen Web-Präsenz vorgesehen ist, bieten die Content-Management-Programme in diesem Punkt den höchsten Abstraktionsgrad, bei ihnen ist das Format des eingespielten Content von dem letztendlichen Aussehen der exportierten Web-Präsenz, dem Export-Medium, unabhängig.

Dieser Vorteil wirkt sich jedoch gleichzeitig als Nachteil aus, denn durch die spezielle Aufbereitung von Inhalt mit der vorrangigen Zielrichtung „Web-Präsenz", können für die Publikation im WWW besser geeignete Dokumente erstellt werden. Dies betrifft sowohl den Umfang als auch den Inhalt und die Verknüpfung der vorhandenen Informationen. Dementsprechend nutzen die WCMS den Vorteil der konkreten Ausrichtung auf Web-Präsenzen als Zielmedium, bedienen sich dabei aber der aus dem Content-Management bekannten Mechanismen zur Verwaltung der Informationen getrennt von dem finalen Layout. In Verbindung mit speziell auf die Verwaltung von Web-Präsenzen ausgerichteten Workflows bieten sie zusammen mit verteiltem Zugriff auf die verwalteten Informationen bei größeren Informations-Beständen deutliche Vorteile gegenüber den „Editor"-Programmen. In ihrer Architektur sind sie aufgrund der von ihnen gebotenen Workflow- und Verwaltungskomponenten aufwendiger als die „Editor"-Programme; an die umfassende Komplexität eines vollwertigen Content-Management-Systems kommen sie i.d.R. jedoch nicht heran.

Die spezielle Komplexität und die besonderen Fähigkeiten des WCMS liegen indes darauf gerichtet, ein oder mehrere Gesamt-Dokumente, die Web-Präsenz(en), bestehend aus vielen Einzelteilen, den Teil-Dokumenten bzw. Web-Seiten, über eine weitgehend hierarchische Struktur zu produzieren. Dabei liegt das Hauptaugenmerk auf den ständig notwendigen Aktualisierungen der Web-Seiten: Sie werden erstellt, liegen (teilweise) fertig vor, werden modifiziert oder komplett ersetzt; neue Dokumente werden eingefügt und veraltete aus der Web-Präsenz entfernt. Dies erfordert eine aufwendige Verwaltung der zwischen den Dokumenten in den verschiedenen Versionen existierenden Hyperlinks. Für die finale, aktuell publizierte Version, die Web-Präsenz, muss trotz der hohen Dynamik des Dokumentenbestandes konsequent Hyperlink-Konsistenz bestehen.

1.2 Zielsetzung und Thema dieser Arbeit

Das Ziel dieser am Institut für Telematik angefertigten Arbeit ist die Entwicklung eines an die individuellen Bedürfnisse der Projektpartner anpassbaren, bezüglich seiner Komponenten offenen Online-Redaktionssystems für das Web-Präsenz-Content-Management. Motiviert durch die initiale Konzeption für die Produktion einer Online-Zeitung – daher auch die Bezeichnung als Online-Redaktionssystem – sollen damit entsprechend den Randbedingungen des Projektumfelds Web-Präsenzen mittlerer Größe, d.h. ca. 1000 - 5000 Dokumente, effizient verwaltet werden können.

Kerngedanke der Entwicklung ist das Publizieren auf die Web-Präsenz von den Arbeitsplätzen aus: Mit dem System sollen die Mitarbeiter des Unternehmens aus den Fachabteilungen heraus in die Lage versetzt werden, mit einer beliebigen Editor-Komponente Dokumente für die Unternehmens-Web-Präsenz zu erstellen. Nach dem Durchlaufen einer Qualitätssicherung sollen diese Dokumente dann in der von dem Unternehmen gewünschten Gestaltung auf der Web-Präsenz veröffentlicht werden. Dabei muss das System die Verwaltung des ständig zu aktualisierenden Dokumentenbestandes genauso zuverlässig gewährleisten wie die Konsistenz der Verknüpfungen zwischen seinen Dokumenten. Die Einbindung von Datenquellen, hauptsächlich relationalen Datenbanken, vervollständigt die auf der Web-Präsenz zu publizierenden Inhalte.

Mit dem konkreten Einsatz im Unternehmen verbunden, soll so ein portables, auf Internet-Technologie basierendes System entwickelt werden, das sich leicht in ein Unternehmensumfeld integrieren lässt und in der von ihm produzierten Web-Präsenz der Individualitätsforderung des Unternehmens gerecht wird. Seine Komponenten und die auf die Verwaltung von Web-Präsenzen ausgerichteten Workflows müssen die Basis für ein effizientes Web-Präsenz-Management bereitstellen. Über vielfältige in das System integrierte Export-Mechanismen sollen sich abgesehen von vielgesichtigen statischen Web-Präsenzen auch komplett dynamische Präsenzen betreiben lassen.

Aus der Forderung nach Portabilität des Gesamtsystems bedingt sich der Verzicht auf die Nutzung bereits in Web- oder Application-Servern implementierter Funktionalitäten; für den Einsatz in einer beliebigen Umgebung ist es notwendig, Komponenten zu entwickeln, die gegebenenfalls auch ohne eine bestehende Server-Umgebung eingesetzt werden können. Bei der konkreten Installation im Unternehmen können diese dann entweder alleinstehend eingesetzt oder in die Infrastruktur, d.h. die Server-Umgebung, eingebunden werden.

1.2.1 Konzeption und Realisierung

Im technischen Konzept des am Institut für Telematik entwickelten Online-Redaktionssystems mit dem Namen „JDaphne“ liegt implementatorisch das Hauptaugenmerk auf der Plattformunabhängigkeit und auf den offenen Standards. Alle Komponenten, sowohl auf Server- als auch auf Client-Seite, sollen, gemäß den heterogenen Betriebssystemumfeldern im Unternehmen, auf allen gängigen Betriebssystemplattformen arbeiten. Um diese Plattformunabhängigkeit zu gewährleisten, sind die Kernkomponenten auf Server-Seite in Java (vgl. Seite 50) implementiert. Alle auf Server-Seite eingesetzten Komponenten, beispielsweise die Datenbank, sind für den Server konfigurierbar und können bei Bedarf einfach durch konkret auf einem Betriebssystem verfügbare Komponenten ersetzt werden.

Auf der Client-Seite wird Plattformunabhängigkeit erreicht durch den Einsatz von HTML (siehe Seite 42) und einem Java-Applet (siehe Seite 51) als Benutzerschnittstelle; der Client muss dementsprechend lediglich über einen Web-Browser (siehe Seite 38) verfügen.

1.2.1.1 Publizieren vom Arbeitsplatz aus

Besonderer Wert wird im Rahmen dieser Arbeit auf die Pflege der Web-Präsenz durch die Mitarbeiter des Unternehmens gelegt – im Unterschied zu bislang weit verbreiteten Vorgehensweisen, wie:

- Auslagern der Web-Präsenz (Agentur)
- Webmaster als Allein-Zuständiger für Technik und Inhalte
- Web-Team arbeitet als kleine Gruppe analog zu einer Online-Redaktion.

Eine Web-Präsenz lebt von ihren Inhalten; diejenigen, die in einem Unternehmen über die Inhalte verfügen, sind die Mitarbeiter der Fachabteilungen. Die obenstehend aufgeführten Vorgehensweisen haben gemeinsam, dass diejenigen, die mit der Pflege der Web-Präsenz beauftragt sind, nicht identisch mit denen sind, die über die dafür erforderlichen Inhalte verfügen. Folglich wird bei allen drei Varianten ein zusätzlicher Informationsbeschaffungsprozess notwendig: Als „Bittsteller“ auftretend fordern die Web-Präsenz-Betreuer von den Mitarbeitern der Fachabteilungen deren Hilfe bei der Erstellung von Inhalten ein; wie die Praxis zeigt, ein mühsames Vorgehen.

Effizienter erscheint hier der Ansatz, direkt für die Mitarbeiter der Fachabteilungen, die aufgrund ihrer Tätigkeit über die für die Unternehmens-Web-Präsenz relevanten Informationen verfügen, die Möglichkeit zu schaffen, diese selbsttätig für die Publikation im WWW aufzubereiten. Nutzung der Internet-Technologie hilft hier, ein WCMS aufzubauen, das den Mitarbeitern eines Unternehmens die aktive Teilnahme an der Außendarstellung des Unternehmens erlaubt. Dabei kann sowohl vom Arbeitsplatz im Unternehmen aus als auch, bei entsprechender externer Zugriffsmöglichkeit auf das System, von einem beliebigen Arbeitsplatz im Internet aus, an der Web-Präsenz gearbeitet werden.

1.2.1.1.1 Einfache Bedienbarkeit – Auslagerung der Eingabekomponente

Damit ein beliebiger Mitarbeiter aus einer Fachabteilung in der Lage ist, diejenigen Informationen, die er bereits im Rahmen seiner normalen Tätigkeit zusammengetragen hat, für die Web-Präsenz zur Verfügung zu stellen, muss das eingesetzte System einfach in seiner Bedienbarkeit sein. Aus diesem Grund wird bei dem im Rahmen dieser Arbeit beschriebenen System die Strategie verfolgt, den Benutzer die Dokumente mit dem ihm bekannten und vertrauten Editor seiner Wahl erstellen und bearbeiten zu lassen; einzige Voraussetzung stellt dabei das Vorhandensein eines HTML-Exports aus diesem Editor heraus, eine Funktion, die mittlerweile die Mehrzahl aller im Unternehmen eingesetzten Office-Anwendungen beherrscht, dar.

An dieser Stelle eine eigenständige Eingabekomponente zu entwickeln, deren Bedienbarkeit an die kommerzieller Produkte herankommt, würde aufgrund der aus den gängigen Produkten bekannten, aufwendig zu implementierenden Funktionsanforderungen (z.B. Rechtschreibkontrolle, Drag&Drop, Syntax-Highlighting etc.) den Rahmen der projektierten Arbeit sprengen und hätte zudem den Nachteil, dass die Benutzer für die Web-Präsenz-Pflege nicht mit ihrer gewohnten Standardanwendung arbeiten könnten.

Aus technischer Sicht hat die Entscheidung, dem Benutzer seinen gewohnten Editor als Eingabekomponente zur Verfügung zu stellen, gravierende Auswirkungen auf die Konzeption von JDaphne:

1. Aus Gründen der Bedienerfreundlichkeit ist ein Mechanismus erforderlich, der die komfortable Integration des Editors übernimmt.

2. Damit ein beliebiger Editor eingesetzt werden kann, muss das System dateibasierte Dokumente verwalten.

3. Um auf eine breite Auswahl an Editor-Programmen zurückgreifen zu können, ist HTML als Datenformat für die Inhalte zu nutzen.

4. Das System muss auch mit potentiell semi-strukturierten Inhalten arbeiten können, da keine Kontrolle über die Erstellung und Aufbereitung der Dokumente im Editor möglich ist.

5. Die Verarbeitung der Dokumente kann nicht im Editor stattfinden, d.h. beim Bearbeitungsvorgang wird immer oben beschriebener Dateitransfer zwischen System und Editor notwendig.

1.2.1.1.2 Komfort

Verbunden mit der einfachen Bedienung des Systems ist der Bedienkomfort. Nur wenn der Aufwand, der dem Mitarbeiter bei der Erstellung von Dokumenten für das WWW zugemutet wird, akzeptabel ist, gibt es bei ihm die Bereitschaft, von sich aus zu der Außenrepräsentation, die im übrigen auch einen Teil seiner Eigenrepräsentation darstellt, beizutragen.

Mit einem System, das die Eingabekomponente auslagert, übergreifenden Bedienkomfort bereitzustellen, ist schwer, da diese letztendlich das dem Benutzer direkt zur Verfügung stehende Werkzeug ist. Vom System aus besteht auf dieses jedoch keine Eingriffsmöglichkeit, da es nicht Teil des Systems ist und von diesem nicht kontrolliert, sondern nur integriert wird. Für das Online-Redaktionssystem bedeutet dies, dass sämtliche Verarbeitungsschritte, die notwendig sind, um ein wirkungsvolles Content- und Hyperlink-Management bereitzustellen, auf einer internen Komponente, d.h. serverseitig stattfinden müssen: Dokumente durchlaufen somit während eines Bearbeitungsdurchgangs sowohl lokal am Benutzerarbeitsplatz als auch auf der Server-Seite stattfindende Bearbeitungsschritte. Mit einer wachsenden Zahl für einzelne Arbeitsgänge notwendiger Schritte sinkt der Benutzerkomfort erheblich. Ziel muss es folglich sein, ihre Anzahl bei der Bearbeitung eines Dokuments mit serverseitigen „WCMS-Eingriffen“ zu minimieren.

1.2.1.1.3 Qualitätssicherung

So reizvoll der Gedanke erscheint, jeden Mitarbeiter im Rahmen seiner normalen Tätigkeit direkt für das WWW publizieren zu lassen, – ohne eine Qualitätssicherungskomponente ist dieses Konzept nur wenig erfolgsversprechend. Nicht jeder Mitarbeiter, der über die notwendigen Informationen verfügt, ist in der Lage, diese kundengerecht für das WWW aufzubereiten. Wird deshalb hinter den Erstellungsprozess und vor den eigentlichen Publikationsvorgang noch eine Qualitätssicherungskomponente geschaltet, ein Mehr-Augen-Prinzip für alle Dokumente realisiert, so ist eine höhere Qualität der Dokumente zu erwarten und zudem kann auf diese Weise verhindert werden, dass Informationen veröffentlicht werden, die nach der Unternehmensstrategie nicht die Freigabe für eine Publikation besitzen.

1.2.1.2 Verteilte Architektur

Als webbasierte Anwendung ist JDaphne zwangsläufig auf einen Web-Browser als Client festgelegt. Durch den Einsatz eines Java-Applets wird hier dem Benutzer eine interaktive Schnittstelle geboten. Auf der Server-Seite ist JDaphne's Funktionalität auf einzelne Server-Komponenten verteilt. Funktionalitäten werden sowohl von dem Applet als auch von den Server-Komponenten untereinander über das Netzwerk abgerufen.

Die Entscheidung, eine verteilte Architektur zu wählen, liegt zu einem großen Teil in der Interaktivität des Java-Applets begründet: als Kontrolleinheit für die Editor-Anwendung übernimmt es den Aufruf der serverseitig verfügbaren Funktionen, beispielsweise die Hyperlink-Umsetzung in den Dokumenten. Zusätzlich erlaubt der verteilte Aufbau flexibel die lastverteilende Installation des Systems auf mehrere Server-Rechner.

1.2.1.3 Rollenbasierte Zugriffskontrolle

Die inhaltliche Zugriffskontrolle innerhalb von JDaphne erfolgt über die Kombination von Rollen und Verzeichnissen [ZHH+99]. Über die Rollen werden die Berechtigungen zu bestimmten Aktionen, z.B. Schreiben, Lesen oder Publizieren vergeben; für die inhaltlichen Berechtigungen, d.h. für diejenigen Objekte, auf die die Aktionen wirken, – im Rahmen von JDaphne Dateien – werden Verzeichnisse, im weiteren auch als Ressorts bezeichnet, herangezogen. Diese Verzeichnisse sind nicht direkt an Benutzer gekoppelt, sondern an Gruppen. Über die Zugehörigkeit zu einer oder mehreren Gruppen, erhält jeder Benutzer somit die Berechtigung, in den zugeordneten Ressorts zu agieren. Welche Arbeitsschritte im einzelnen erlaubt sind, ist von der Rolle abhängig, die der Benutzer innehat.

Über die Rollen und Verzeichnisse lässt sich eine, beispielsweise an die Unternehmenshierarchie angepasste, Rechtevergabe bewerkstelligen. Während die Erfassung und Modifikation von Inhalten durch die Sacharbeiter, mit der zugehörigen Rolle *Autor*, erledigt wird, sorgt die übergeordnete Ebene, im Unternehmen die Ebene mit inhaltlicher Verantwortung, bei JDaphne als *Freigabe-Berechtigter* bezeichnet, für die inhaltliche Kontrolle und damit die Qualitätssicherung.

Die Gesamtübersicht über den Stand der Web-Präsenz wird von der Rolle des *Webmaster* erwartet; mit Zugriff auf alle Ressorts versehen, obliegt es den Inhabern dieser Rolle, die von den *Freigabe-Berechtigten* abgezeichneten Dokumente ins Internet zu publizieren.

1.2.1.4 Dokument-Workflows

Angepasst an die hohe Dynamik des Dokumentenbestandes einer Web-Präsenz muss das Online-Redaktionssystem die Dokumente in verschiedenen Status-Versionen vorhalten. Unterschieden wird beispielsweise zwischen Dokumenten, die sich noch in der Bearbeitung befinden, Dokumenten, die gerade zum Abzeichnen durch eine berechtigte Stelle vorliegen, Dokumente, die auf die Web-Präsenz publiziert werden können, und schließlich der gerade im WWW sichtbaren Version.

Die Implementation von JDaphne sieht verschiedene solcher Workflows für die zu verwaltenden Dokumente vor. Da diese stark mit den implementierten Rollen verknüpft sind, ist ihre Konfiguration über die Vergabe entsprechender Berechtigungen an die Rollen realisierbar. Beispielsweise stellt sich der normale Workflow für ein Dokument dreistufig dar: Das Dokument wird bearbeitet und schließlich von dem Autor fertiggestellt, dieses fertiggestellte Dokument wird von dem *Freigabe-Berechtigten* abgezeichnet, das abgezeichnete Dokument wird von dem *Webmaster* publiziert.

Der hier exemplarisch aufgegriffene Workflow bildet ein Sechs-Augen-Prinzip ab; um ein Vier-Augen-Prinzip abzubilden, kann es beispielsweise dem *Freigabe-Berechtigten* erlaubt werden, eigenständig Dokumente zu publizieren. Alternativ können von dem Autor als „fertig“ gemeldete Dokumente von diesem selbst zusätzlich auch abgezeichnet und dann direkt von dem *Webmaster* publiziert werden.

Wie in dem Beispiel skizziert, lassen sich für alle im Rahmen des Online-Redaktionssystems anfallenden Workflows entsprechende Abbildungen auf die Rollen vornehmen. Im Zusammenspiel mit „Aufgaben-Listen“ (siehe Seite 166), die über die Workflows

für die Rolleninhaber generiert werden, sind flexible Mechanismen realisiert, die durch einfache Konfigurationsänderungen angepasst werden können.

1.2.1.5 Hyperlink-Management

Eng verbunden mit den Dokumenten-Workflows ist das Management der Hyperlinks zwischen den Dokumenten. Abgesehen von für die Autoren hilfreichen Funktionen wie beispielsweise „Hyperlink-Vorschlag auf Anfrage“ werden umfassende Algorithmen benötigt, die die Konsistenz der Hyperlinks in Abhängigkeit von dem Dokumentenstatus garantieren. Zu diesem Zweck wird die vollständige Kenntnis aller Referenzen zwischen den Elementen des Dokumentenbestands vorausgesetzt. Die hohe Dynamik, die diesem innewohnt, führt dabei zu aufwendigen Verwaltungsmechanismen, die nachhalten, ob ein referenziertes Dokument gerade überarbeitet wird, fertig, abgezeichnet oder publiziert ist, ob es gerade aus der publizierten Version entfernt oder in ein anderes Verzeichnis verschoben wird oder komplett durch ein neues Dokument ersetzt wird.

Innerhalb einer Statusebene, z.B. innerhalb der publizierten Version des Dokumentenbestandes, muss jeweils Konsistenz erhalten werden: dynamisch passt JDaphne bei jeder Aktion mit einem Dokument alle dieses Dokument referenzierenden Dokumente an die neue Situation an. Beim Einspielen eines Dokumentes werden alle Referenzen extrahiert und in der zum System gehörigen Datenbank gesichert; dokumentstatusabhängig werden sie dort aktualisiert.

Die statusabhängige Konsistenz der Hyperlinkstruktur wird im Redaktionssystem über das automatisierte (De-)Aktivieren von Hyperlinks gewährleistet: In jedem Dokument werden diejenigen Hyperlinks, die gerade nicht funktionstüchtig sind, da das referenzierte Dokument statusbedingt nicht verfügbar ist, deaktiviert. Sobald das Dokument in dem entsprechenden Status wieder verfügbar ist, wird der Hyperlink erneut aktiviert.

Auf der Bearbeiter-Ebene sind Broken-Links, Verweise auf nicht existierende Dokumente, nach wie vor möglich. Ein Statuswechsel ist für ein Dokument nur erlaubt, wenn es ausschließlich funktionstüchtige Hyperlinks besitzt.

1.2.1.6 Ressort-Struktur

Wie im Kontext der rollenbasierten Zugriffskontrolle bereits erwähnt, erfolgt die inhaltliche Strukturierung der Web-Präsenz über Ressorts analog zu Verzeichnissen im Dateisystem. Diese können ineinander geschachtelt werden und bieten somit die Möglichkeit zur feingliedrigen Verwaltung der Informationen.

Jeder Informationsbaustein wird fest einem Ressort zugeordnet. Diese primäre Ordnung wird analog einem Dateisystem in einem hierarchischen Baum präsentiert, für die Informationsbausteine können zusätzlich sekundäre Zugehörigkeiten vergeben werden. Anhand dieser sekundären Zugehörigkeiten, die nicht wie die Ressorts Teil der rollenbasierten Zugriffskontrolle sind, lassen sich alternative Darstellungen der Web-Präsenz beim Publizieren aufbauen. Während in der Normaleinstellung die Ressortstruktur auch in der fertigen Web-Präsenz sichtbar wird, z.B. in der Navigation, lassen sich über die sekundären Zugehörigkeiten von der Verwaltungsstruktur unabhängige Darstellungen erzeugen.

1.2.2 Der Aufbau dieser Arbeit

Die Entwicklung einer Web-Präsenz ist ein facettenreiches interdisziplinäres Unterfangen. Neben den technischen Aspekten, die mit der Entwicklung des Online-Redaktionssystems „JDaphne“ zum Web-Präsenz-Content-Management im Vordergrund dieser Arbeit stehen, sind in großem Maße auch organisatorische, verwaltungstechnische, soziale und künstlerische

Aspekte betroffen. Alle mit dem Web-Präsenz-Management verbundenen Themengebiete in ihrer vollen Komplexität zu erfassen, sprengt den Rahmen dieser Arbeit; soweit möglich, wurde aber an denjenigen Stellen, an denen sich ihre Diskussion sinnvoll kontextualisieren lässt, darauf eingegangen. Eingebettet in die Metapher des Lebenszyklus' einer Web-Präsenz beschreiben die folgenden sechs Kapitel ausgehend von der Problemstellung und den technischen Grundlagen die Konzeption, Implementation sowie den Einsatz des webbasierten Online-Redaktionssystems bevor eine Einordnung und schließlich die Zusammenfassung diese Arbeit abschließen.

Das zweite Kapitel greift ausführlich die bereits in dieser Einleitung andiskutierte Problemstellung des Web-Präsenz-Managements auf. Insbesondere wird dort einleitend der Frage nachgegangen, wie sich die bereits für Printmedien etablierten Vorgehensweisen von denen bei der Publikation im WWW unterscheiden: Der zentrale Unterschied zwischen einer Printausgabe und einer Web-Präsenz ist die hohe Dynamik der letzteren. Während es für jede Print-Ausgabe einen geregelten Redaktionsschluss gibt, zu dem das fertige Printdokument vorliegen muss, handelt es sich bei einer Web-Präsenz um ein umfangreiches Gesamtdokument, aufgebaut aus wechselnden Teildokumenten, das einem ständigen Wandel unterliegt – wenn die oben besprochenen Aspekte betrachtet werden – sogar zwangsweise unterliegen muss.

Diesen Zustand dynamischer Konsistenz einer Web-Präsenz in einem ersten Schritt zu erreichen und in einem zweiten Schritt zu bewahren, ist die Aufgabe eines Web-Präsenz-Content-Management-Systems. Die Anforderungen an ein solches System sowie die aus dem konkreten Einsatz im Unternehmen erwachsenen Randbedingungen stellen die Problemstellung für diese Arbeit dar, auf deren Grundlage die Entwicklung des Online-Redaktionssystems am Institut für Telematik erfolgte.

Bereits bei der Lektüre der Einleitung wird deutlich, dass ein kurzer Überblick über die Grundlagen des hier einschlägigen Themenbereichs hilfreich ist. Die dazu im dritten Kapitel diskutierten Grundlagen sind in zwei Bereiche untergliedert: zum einen werden die technischen Aspekte, Protokolle und Formate erörtert, die für den Aufbau und Einsatz eines Online-Redaktionssystems relevant sind, zum anderen werden die für die Diskussion der Konzeption und des Aufbaus einer Web-Präsenz notwendigen Begriffe eingeführt und eingeordnet.

Das in dieser Arbeit beschriebene Online-Redaktionssystem „JDaphne“ hat seine Wurzeln in der Implementation eines frühen Prototypens mit dem Namen „DAPHNE“. Auf dessen Datenmodell und Konzeption basierend wurde die in Java implementierte Variante „JDaphne“, die im Zentrum dieser Arbeit steht, entworfen. Kapitel vier beschreibt in einem ersten Abschnitt die mit DAPHNE realisierte Variante eines Online-Redaktionssystems am Institut für Telematik. An die Diskussion seiner Realisierung anschließend werden in zwei weiteren Abschnitten Add-ons in ihrer Entwicklung beschrieben, die als Komponente für ein Redaktionssystem unerlässlich sind bzw. ein wichtiges Hilfsmittel für dessen Einsatz darstellen. Dort wird zum einen die separat entwickelte, in Grundzügen in JDaphne aufgenommene Hyperlink-Management-Komponente und zum anderen das insbesondere für Web-Autoren interessante Konzept der Assistenten- und Gateway-Komponenten – ergänzt durch deren prototypische Implementationen – dargestellt.

Das zentrale Kapitel fünf der vorliegenden Arbeit beschreibt die momentane Ausprägung von JDaphne mit den zugrundeliegenden Überlegungen und erläutert in diesem Rahmen seine Konzeption sowie sein verteilte Architektur. Großer Wert in diesem Kapitel wird auf die entworfenen Verwaltungsstrukturen einerseits und die realisierten Funktionalitäten andererseits gelegt.

In Kapitel sechs wird exemplarisch die Konzeption und der Aufbau einer Web-Präsenz mit einem System wie JDaphne/DAPHNE basierend auf den praktischen Erfahrungen vorgestellt.

Die in der Konzeptionsphase zu beachtenden Punkte werden dabei unter Berücksichtigung des Redaktionssystems beleuchtet und diskutiert. Ferner wird die mit dem System realisierbare Integration der Web-Präsenz ins Unternehmen, ein wichtiger Punkt bei der Anschaffung eines solchen Systems, besprochen. Im Rahmen der Realisierung der exemplarischen Web-Präsenz steht das „Arbeiten“ mit JDaphne im Vordergrund; die Benutzbarkeit des Systems wird erläutert. Abschließend werden das „Life-Going“ und der Betrieb der Web-Präsenz als weitere Phase in deren Lebenszyklus betrachtet, bevor das Kapitel mit dem Ausblick auf einen Relaunch endet und sich der Kreis damit schließt.

Abgeschlossen wird diese Arbeit durch die Einordnung von JDaphne in das Umfeld existierender Systeme und Konzepte des Web-Präsenz-Managements. Historisch der Entwicklung von Systemen zum „Web-Präsenz-Management“ folgend wird an dieser Stelle ein Überblick über die verschiedenen Kategorien von Werkzeugen, die über „Produkt-Charakter“ verfügen, gegeben. Im Anschluss an eine Übersicht über Entwicklungen aus dem Umfeld der Multi-Autoren-Systeme und einem Ausblick in die Entwicklung „Modellbasierter Web-Präsenz-Generatoren“ erfolgt eine kritische Würdigung von JDaphne.

Das letzte Kapitel liefert eine Zusammenfassung und einen kurzer Ausblick in künftige Entwicklungen.

1.3 Resümee

Die Praxis hat gezeigt, dass ein hoher Bedarf an maßgeschneiderten WCMS für individuelle Web-Präsenzen vorhanden ist. Ein solches System aufzubauen, stellt aufgrund der vielfältigen involvierten Techniken und der im praktischen Umfeld vorgegebenen Randbedingungen ein interessantes Projekt dar.

Eine maßzuschneidernde Export-/Publizierungskomponente und eine allgemein einsetzbare, konfigurierbare Verwaltungskomponente sowie eine plattformunabhängige, interaktive Benutzerschnittstelle bilden das in dieser Arbeit beschriebene Online-Redaktionssystem „JDaphne“. Über die Einbindung einer beliebigen Eingabekomponente sowie die jeweilige Ausbildung der Export-Komponente lässt sich dieses System in der erforderlichen Weise speziell auf konkrete Belange der Unternehmen anpassen.

Die im Laufe der Entwicklung des in dieser Arbeit beschriebenen Prototypen entstandenen Einzelanpassungen konnten mit verschiedenen Projektpartnern erfolgreich evaluiert werden und kommen sowohl für die Verwaltung von Web-Präsenzen als auch für den Aufbau eines Intranets, einem internen Informationssystem, in der Praxis zum Einsatz. Auch mit größeren Dokument-Beständen, d.h. im konkreten Fall deutlich mehr als zweitausend Dokumenten, bleibt durch die Ressort-Einteilung die Übersicht über die Dokumente erhalten. Die von den Systemen verwalteten Dokumentenformate (siehe Seite 42) variieren stark; während für das Internet HTML, JPG und PDF als Format dominieren, werden im Intranet viele Dokumente in den nativen Formaten interner Anwendungen, beispielsweise als PowerPoint-Präsentationen oder Word-Dokumente, eingespielt.

Aufgrund der konzeptionellen Entscheidung, die Benutzer des Systems mit den von ihnen gewohnten Editoren arbeiten zu lassen, konnte auf die eigene Entwicklung einer auf dem Markt in professioneller Ausführung bereits mehrfach verfügbaren Eingabekomponente verzichtet und dem Benutzer stattdessen mittels dieser seine gewohnte Arbeitsumgebung erhalten werden. Da alle in Frage kommenden Eingabekomponenten dateibasiert arbeiten und sich zum Zeitpunkt der Entwicklung noch keine produktreifen XML-Editoren auf dem Markt befanden, geschweige denn im Unternehmen verfügbar waren, wurde damit der Grundstein für die HTML-basierte Dokumentstrukturierung innerhalb des Systems gelegt.

Diese Entscheidung hat sich insofern für den Einsatz des Systems im Unternehmen

bewährt, als dass sie aufwendige und zum Teil unmögliche Konvertierungen eingespielter Dateien vermied. Für den flexiblen Einsatz der Dokumente, insbesondere den Datenaustausch mit anderen Systemen haben sich dadurch jedoch auf Dauer Einschränkungen ergeben, die eindeutig die Verwendung von XML als Basisformat nahe legen. Auch wenn das hier entwickelte System XML-Dokumente verwalten kann, sind aufgrund der auf HTML abgestimmten Hyperlinkmanagement- und Exportmechanismen Modifikationen der internen Methoden notwendig, um einen echten Mehrwert aus dem XML-Format zu erzielen; nach der kürzlich durch das W3C erfolgten Verabschiedung des Standards für XML-Links als Recommendation [DMO01] steht dem zum jetzigen Zeitpunkt nichts mehr entgegen.

1.3.1 Integration ins Unternehmen

Das entwickelte Online-Redaktionssystem ließ sich aufgrund seiner Architektur ohne Probleme in verschiedene Umgebungen einpassen. Die konsequente Auslegung als webbasiertes System, vollständig aufgebaut mit Internet-Technologie, und die Nutzung offener Standards haben dazu beigetragen, dass die Integration relativ reibungslos ablaufen konnte. Die große Unabhängigkeit der einzelnen, für den Betrieb notwendigen Komponenten hat speziell die Anpassung des Systems für den Betrieb mit verschiedenen Datenbanken und Web-Servern erleichtert.

Insbesondere durch die Auslagerung der Eingabekomponente bzw. des Editors, kann jedem Benutzer die von ihm bevorzugte Schnittstelle zur Erstellung und Modifikation von Dokumenten geboten werden. Die Entscheidung, auf diese Weise vorzugehen, ist jedoch nicht allein auf Grund deutlicher Komforteinbußen ambivalent gewesen. Für die mit dieser Entscheidung gewonnene Flexibilität muss bei der Einführung des Systems ein nicht zu vernachlässigender Preis bezahlt werden: Zum einen ist auf der Client-Seite bei der Erstbenutzung des Systems ein an sich unerwünschter Konfigurationsschritt entstanden, der sich bei größeren Benutzerzahlen nur durch zentrale Konfigurationen abfangen lässt, zum anderen hat die Entscheidung, die Editor-Komponente des WCMS frei konfigurierbar zu gestalten und die Wahl damit dem Anwender zu überlassen, auf Grund der nicht immer standardkonformen Dokumentexporte der Office-Produkte, unter Windows von den Sachbearbeitern als Editor bevorzugt, zu einem unerwartet hohen Einarbeitungsaufwand geführt.

1.3.1.1 Schulungen/Support

Der Schulungs- und Support-Aufwand erklärt sich neben diversen Unzulänglichkeiten der ausgewählten Editoren unter anderem auch mit deren „Fehlbedienung“ durch die Anwender, denn nur bei korrekter Bedienung der Editoren (Formatvorlagen) entstehen strukturierte Dokumente, die von dem System sinnvoll weiterverarbeitet werden können.

In der Praxis haben sich als notwendige Maßnahmen für einen reibungslosen Betrieb des Online-Redaktionssystems Schulungen erwiesen: Bei der Inbetriebnahme des Systems im Unternehmen steht an erster Stelle eine Einführung in die im Internet gebräuchlichen Datenformate – in Form kurzer Präsentationen werden die Benutzer über die besonderen Anforderungen der Dokumentenformate des WWW informiert. In einem zweiten Schritt erlernen sie den Umgang mit dem Online-Redaktionssystem, werden in ihre Rollen eingeführt und dabei mit den im System verfügbaren Workflows bekannt gemacht.

1.3.1.2 Workflow

Die Workflows steuern zusammen mit den Dokumentübersichten den Dokumentlebenszyklus und führen die verschiedenen Benutzer zu einer gemeinsam arbeitenden Einheit zusammen.

Im Rahmen der Integration des Systems steht die sinnvolle Einbettung der neuen in die bereits im Unternehmen etablierten Arbeitsabläufe an, gegebenenfalls werden speziell zu entwickelnde Anbindungen an bestehende Workflow-Systeme notwendig.

Mit dem Betrieb des Systems werden den Benutzern entsprechend ihren Rollen die anfallenden Aufgaben in Aufgaben-Listen zusammengestellt; die Aufgabenverteilung erfolgt anhand der unternehmensspezifischen Konfiguration der implementierten Workflows, die, obwohl vergleichsweise einfach, den Anforderungen genügen.

1.3.1.3 Zugriffskontrolle

Die in der Konzeption vorgestellte rollenbasierte Zugriffskontrolle hat sich auch bei größeren Nutzerzahlen, konkret mehr als einhundert Nutzer, bewährt. Gleiches gilt für das vom System bereitgestellte Gruppenkonzept, das eine angemessene Zuordnung zu Sachgebieten innerhalb des Unternehmens realisiert. Die konkrete Integration mit einer unternehmensweiten Benutzerverwaltung erfolgte jedoch weniger über die Gruppen als vielmehr, wie dies exemplarisch durchgeführt wurde, durch die Abbildung von Windows-NT-Kennungen auf die Nutzerkennungen des Redaktionssystems und in der Folge durch die Nutzung der dort spezifizierten Gruppenzugehörigkeiten.

1.3.1.4 Schnittstellen

Substantieller Bestandteil der Integration des Redaktionssystems in ein Unternehmen ist im weiteren die Anbindung an bereits vorhandene Informationsquellen. Deren Einbindung in die Web-Präsenz stellt eine Aufgabe dar, die sich, prinzipiell nur bedingt mit dem im Rahmen dieser Arbeit entwickelten System, lösen lässt: Je stärker sie durch das System ermöglicht wird, desto näher muss es sich in seiner Konzeption bei einem Application-Server (siehe Seite 180) positionieren, dessen Konzepte direkt auf die Ausführung von Kleinst-Applikationen abzielen, die in effizienter Weise die Darstellung von Informationsressourcen für die Web-Präsenz übernehmen können. Da die Umsetzung solcher Konzepte nicht Bestandteil dieser Arbeit ist, muss bei Bedarf an dieser Stelle auf externe Lösungen ausgewichen werden. Die im Rahmen des hier beschriebenen Prototypen entstandene Lösung ermöglicht die Integration von fremden Datenquellen über ein Mediatorenkonzept auf einer vergleichsweise simplen Art und Weise, die sich im laufenden Betrieb jedoch als tragfähig erwiesen hat.

Da die Verwaltung der externen Datenbestände sich bei dem hier umgesetzten Konzept lediglich auf die im Redaktionssystem für diese angelegten „Repräsentanten“ beschränkt, die dynamischen Datenbestände an sich jedoch ausspart, ist im Rahmen eines übergreifenden Content-Managements hier die Entwicklung ergänzender Konzepte notwendig.

1.3.2 Hyperlink-Management

Zu Beginn unterschätzt wurde bei der Entwicklung des Online-Redaktionssystems das Management der die Dokumente verknüpfenden Hyperlinks. Während ein erster Prototyp noch ohne eine solche Funktionalität auskommen musste, konnten mit einer separat entwickelten, exemplarisch am Institut für Telematik eingesetzten Komponente positive Erfahrungen gesammelt werden, die zu ihrer Integration in das aktuelle System geführt haben. Als komplexes Unterfangen erwies sich das Hyperlink-Management mit JDaphne insbesondere deshalb, weil eine dynamische, dokumentenstatusübergreifende Konsistenz erzeugt und erhalten werden muss. Diese gewährleistet das im Rahmen dieser Arbeit entwickelte System durch das dokumentenstatusabhängige Aktivieren und Deaktivieren von Hyperlinks; instantan werden Änderungen im Hyperlink-Netz wirksam und Broken-Links jederzeit effizient verhindert.

Während von der technischen Seite Hyperlinks mit dem System intern effektiv gehandhabt werden, ist auf seiner Benutzerseite hier noch ein deutlicher Mangel zu konstatieren: Da die Eingabekomponente des Benutzers kein direkter Bestandteil des Systems und deshalb jegliche Einflussnahme durch JDaphne unmöglich ist, fehlt dem Benutzer bei dem Einbau von Hyperlinks in seine Dokumente umfassende Unterstützung durch das System. Auch die deutliche Vereinfachung von Referenzen auf andere Dokumente durch die Nutzung eindeutiger Dokumenten-IDs konnte dem Benutzer nur bedingt die Arbeit erleichtern, da Hyperlinks nach wie vor manuell in die Dokumente eingebracht werden müssen. In Anbetracht dieser Problematik bietet der Versuch, zusätzlich über separate Hyperlink-Vorschlaglisten, generiert anhand des verfügbaren Dokumentenbestandes, Abhilfe zu schaffen – trotz seines prinzipiellen Wertes – wenig Nutzen bei der täglichen Arbeit mit dem System, solange keine Integration mit der Editorkomponente erfolgt.

1.3.3 Metadaten

Neben den Hyperlinks nehmen die Metadaten eine exponierte Stellung bei der Verwaltung von Inhalten ein: sowohl für den internen Gebrauch als auch zum Auffinden der Inhalte auf der Web-Präsenz sind sie unerlässlich. Die Erstellung und Pflege von Metadaten stellt einen erheblichen Aufwand dar, der nur von ca. einem Drittel der Informationsanbieter des WWW auf sich genommen wird [LG99]. Mit JDaphne werden die Benutzer bei der Erstellung von Metadaten zu den eingespielten Inhalten durch automatisierte Prozesse unterstützt. Das Redaktionssystem bietet zwei verschiedene Wege, die publizierten Dokumente mit Metadaten zu versehen: zum einen werden, wie allgemein üblich, die Dokumente selbst mit Metadaten ausgestattet; ein Teil der Metadaten, z.B. Erstellungs- und Modifikationsdatum, sind vom System heraus bekannt, weiterhin verfügt jedes Dokument zusätzlich über vom Anwender definierte Metadaten. Zum anderen gibt es ressortabhängige Metadaten, die beim Publizieren von Dokumenten auf der Web-Präsenz in alle dem Ressort zugeordneten Dokumente eingefügt werden. Diese Lösung hat sich bewährt, da sie gewährleistet, dass alle Dokumente zumindest über einen (ressortspezifischen) Basissatz von Metadaten verfügen.

1.3.4 Die fertige Web-Präsenz

Der Maßstab, mit dem ein für das Web-Präsenz-Management eingesetztes Online-Redaktionssystem gemessen wird, sind letzten Endes die mit ihm verwalteten Web-Präsenzen. Diese bestehen im Fall des im Rahmen dieser Arbeit aufgebauten Systems jeweils aus einem inhaltlichen Gerüst, der Ressortstruktur, und den eigentlichen Inhalten, dem Dokumentenbestand; JDaphne gewährleistet die Konsistenz von beidem.

Wie sich eine mit JDaphne generierte Web-Präsenz dem Benutzer im WWW präsentiert, hängt bezüglich des Designs von den benutzten Templates, bezüglich der Präsentation von der gewählten Publikationsmethode ab. Während die tabellenbasierte Variante (siehe Seite 60) im Web-Browser mehr Zeit für die Darstellung benötigt, da die ineinander verschachtelten Tabellen von diesem erst ausgewertet werden müssen, erfordert die Frame-Set-Variante (siehe Seite 59) eine größere Anzahl von Verbindungen zum Web-Server. Die Nutzung textueller Navigationselemente hat sich aufgrund ihrer automatischen Generierung als flexibler als die grafischer Elemente erwiesen, obwohl auch solche erfolgreich eingesetzt wurden.

Ein Zeichen der Flexibilität JDaphne's sind die beiden unterstützten Publizierungsverfahren „statischer Export“ und „dynamischer Export“. Einerseits ist das „Staging“ möglich, bei dem die Web-Präsenz in geregelten Abständen oder bei Bedarf durch manuellen Anstoß vollständig neu generiert und das Ergebnis im Anschluss im WWW sichtbar gemacht wird.

Im Rahmen dieser Neugenerierung werden die zwischenzeitlich im System erfolgten Modifikationen an den Inhalten, der Navigationsstruktur und dem Layout im WWW in allen publizierten Dokumenten sichtbar. Alternativ sind in JDaphne zusätzlich Mechanismen für die teilweise Neugenerierung von einzelnen, modifizierten Dokumentgruppen verfügbar, die automatisiert Hyperlink-Konsistenz auf der externen Web-Präsenz wahren – auch wenn sich beispielsweise Teile der Navigationsstruktur geändert haben.

Andererseits offeriert JDaphne die bei variablen Inhalten interessante Variante des dynamischen Publizierens. Bei ihr werden sämtliche Änderungen im „intern“ publizierten Dokumentenbestand parallel auf der extern zugreifbaren Datenbasis nachvollzogen. Über die oben diskutierte Einbindung von externen Datenquellen über die Mediatorkomponente von JDaphne können dem Benutzer beim dynamischen Export u.a. Schnittstellen zu anderen Informationssystemen bereitgestellt und auf diese Weise bereits vorhandene Datenressourcen in die Web-Präsenz integriert werden.

Für diejenigen Unternehmen, die ihre Web-Präsenz mit dem am Institut für Telematik entwickelten Online-Redaktionssystem erstellt haben, war die Entwicklung der eigenen Web-Präsenz jeweils der erste Schritt in die Online-Welt. Wie von der Konzeption her vorgesehen, hat bei ihnen das Online-Redaktionssystem als Werkzeugkasten gedient, mit dem intern innovativen Technologien der Weg geebnet wurde. Als Resultat der jeweiligen Maßschneiderungsarbeit am Redaktionssystem – an der die unternehmenseigene Anwendungsentwicklung häufig maßgeblich beteiligt war – konnte so im Unternehmen über die Web-Präsenz der Transfer von im Bereich der Internet-Technologie bis dato nicht vorhandenem Basiswissen eingeleitet werden. Über die Möglichkeit, auch nach Abschluss der jeweiligen Projekte den installierten Anwendungscode weiter pflegen und an neue Herausforderungen anpassen zu können, ist in den Unternehmen eine offene Einstellung zu der neuen Technik entstanden, die die Weiterentwicklung der Unternehmen in diesem Feld positiv beeinflusst hat.

Kapitel 2

Die Problemstellung

Während sich durch die jahrelange Erfahrung mit dem Printmedium in den Unternehmen ein umfangreicher Erfahrungsschatz angesammelt hat, betreten sie bei dem Aufbau und der Verwaltung einer Web-Präsenz Neuland. Dieses existiert sowohl bezüglich technischen Anforderungen, beispielsweise der Dokumentenformate und der zugrundeliegenden Informationsarchitektur, als auch in Bezug auf organisatorische, den Aufbauvorgang und den Betrieb einer Web-Präsenz betreffende Arbeitsschritte an sich. Dieses Kapitel erläutert die sich aus einem solchen Schritt für das Unternehmen ergebende Problemstellung beginnend mit einem Vergleich der Publikation für ein Printmedium im Unterschied zu der für ein Onlinemedium. Nach einem exemplarischen Überblick über ein mögliches organisatorisches Vorgehen beim Aufbau einer Web-Präsenz in einem Unternehmen endet dieses Kapitel mit der Diskussion und Beantwortung der Frage, welche Anforderungen an ein Werkzeug zu stellen sind, mit dem diese Aufgabe erleichtert werden kann.

Die Motivation für ein Unternehmen, eine Web-Präsenz zu betreiben, ist, wie in der Einleitung bereits erwähnt, vielfältig. Hauptgrund ist, dass sich bedingt durch die einfache Handhabung des Mediums „Internet“ eine große Zahl Anwender [Ber99] gefunden hat. Damit ist eine breite Basis für die kommerzielle Nutzung des Netzes vorhanden. Bei dieser sind zwei verschiedene Geschäftsmodelle zu unterscheiden. Auf der einen Seite sind die konsumentenorientierten Modelle (Business to Consumer, B2C), bei denen mittels des Internets Leistungen und Produkte an den Endverbraucher gebracht werden. Dem gegenüber stehen die Geschäftsmodelle, die von der Interaktion der Unternehmen untereinander profitieren (Business to Business, B2B). In beiden Fällen wird durch das WWW eine übergreifende Plattform bereitgestellt, die zur einfachen Umsetzung der Geschäftsprozesse beiträgt. Unabhängig von den implementierten Prozessen und dem Geschäftsmodell ist die Web-Präsenz die strategische Einstiegskomponente für diesen Markt.

Bei einem überzeugenden Internet-Auftritt kann die Web-Präsenz für ein Unternehmen eine wichtige Stütze im Wettbewerb mit den Mitkonkurrenten werden. Es gilt, möglichst große Besucherzahlen auf der eigenen Web-Präsenz zu erzeugen, die Besucher auf die Produkte aufmerksam zu machen und sie als Kunden zu gewinnen. Mit dem wachsenden Wettbewerb im Internet ist jeder Kunde jedoch nur einen Maus-Klick weit von Konkurrenzangeboten der entfernt, so dass eine positive Beurteilung entscheidend von der Qualität der bereitgestellten Informationen und deren Aufbereitung abhängt. Anders als in der realen Welt ist es dem Benutzer im WWW möglich, sich seine eigene Sammlung von Informationsressourcen zusammenzustellen. Durch eine Liste von Lesezeichen beispielsweise kann er problemlos verschiedene Angebote aus Online-Shops oder Online-Zeitungen nebeneinander stellen. Dies erlaubt ihm, insbesondere bei Informationsressourcen eine gute Beurteilung der Qualität. Mit den Mitteln des Internets kann er sich so individuell ein hochwertiges personalisiertes

Informationsangebot zusammenstellen.

Eine herausragende Rolle bei dem Auffinden solcher Ressourcen durch den Benutzer spielen die Suchmaschinen [BR01] und Internet-Portale [NBO+99]. Sie versuchen Informationen zu verwalten und bieten für ein breites Publikum die Möglichkeit über Suchanfragen gezielt Dokumente aufzufinden. Mit dem enormen Wachstum des WWW ist dies für die Nutzer oft die einzige Möglichkeit, gezielt bestimmte Informationen zu finden. Für die Betreiber einer Web-Präsenz ist es demzufolge außerordentlich wichtig, ihre Dokumente in einer Art und Weise aufzubereiten, die es solchen Diensten erlaubt, die auf der Web-Präsenz offerierten Inhalte in ihrem Datenbestand aufzunehmen. Neben hinreichenden Metadaten (siehe S. 116) hilft unter anderem ein strukturierter Dokumenten- und Web-Präsenzaufbau die Dokumente zu indexieren.

Eine weitere Auswirkung des WWW ist die verstärkte Internationalisierung. Die einfache Verfügbarkeit und die hohe Vernetzung der Informations-Ressourcen hebt die nationalen Grenzen auf. Für den Informationsabrufenden stellen nationale Grenzen keine realen Grenzen mehr dar. Allein Sprachbarrieren und geographische Entfernungen beim Versand online bestellter Produkte können noch als Problem betrachtet werden. Da die Verwaltung multilingualer Web-Präsenzen den Aufwand, welcher für eine einsprachige Web-Präsenz anfällt, vervielfacht sind solche Web-Präsenzen noch vergleichsweise selten zu finden. Auf internationaler Ebene wird vorrangig Englisch als gemeinsame Kommunikationsbasis eingesetzt.

Dies funktionierte, solange der Großteil der Benutzer des WWW aufgrund seiner Ausbildung mit der englischen Sprache vertraut war. Mit der Öffnung des WWW für eine breitere Benutzergruppe kann davon ausgegangen werden, dass mehr und mehr Benutzer nicht mehr hinreichend mit der englischen Sprache vertraut sind und somit als potenzielle Nutzer für auf die englische Sprache ausgerichteter Web-Präsenzen ausfallen. Je nach Zielgruppe der Web-Präsenz kann dieser Mangel an Benutzbarkeit die Zahl der Nutzer und damit auch die Rentabilität der Web-Präsenz stark einschränken.

Insbesondere größere internationale Unternehmen gehen eher den Weg, für ihre jeweilige nationale Niederlassung eine landesprachliche Web-Präsenz einzusetzen. Diese landesprachlichen Web-Präsenzen haben den Vorteil, auf die jeweiligen kulturellen und sozialen Gegebenheiten effektiver eingehen zu können. Für kleinere Unternehmen, für die insbesondere Vorteile aus der Nutzung des Internets als Plattform erwachsen, ist dieser Weg schlecht gangbar, gerade weil sie keine internationalen Niederlassungen haben. Für sie bietet sich als Lösung eher die Bereitstellung ihrer Inhalte in den benötigten Sprachversionen an. D.h. für sie ist es günstiger, eine einzige Web-Präsenz zu betreiben, deren Dokumentenbestand das anvisierte Sprachenspektrum abdeckt.

2.1 Publizieren für das WWW im Unterschied zum Printmedium

Während in den meisten Unternehmen die Kompetenz im Umgang mit den Printmedien über Jahre hinweg gewachsen ist, entstehen durch den Aufbau und Betrieb einer Web-Präsenz neue, für eine erfolgreiche Web-Präsenz optimalerweise im ersten Versuch zu erfüllende Herausforderungen. Der Umgang mit den Printmedien, d.h. mit Katalogen, Broschüren, Produktbeschreibungen, Anleitungen etc., gestaltet sich in vielen Punkten analog zu den Entsprechungen auf der Web-Präsenz. Es gibt jedoch auch eine Reihe von relevanten Unterschieden zu beachten: in vielen Fällen wird die Web-Präsenz die Printmedien nicht ersetzen, sondern ergänzen. Für ein Unternehmen ist es dann wichtig, den Informationsbestand so medienneutral aufzubereiten, dass er für beide Zwecke eingesetzt werden kann. In wie weit diese Mehrfachnutzung der Informationen möglich ist, hängt in hohem Maße von dem Typ der Informationen ab.

Gemeinsam ist dem Publizieren für Print- und Onlinemedien die Art und Weise, wie die Dokumente entstehen. Distributiv und kooperativ, d.h. in verteilter Zusammenarbeit der Beteiligten, werden die zusammengetragenen Informationen aufbereitet und in Dokumenten zusammengestellt. Ebenso sind beiden Medien verschiedene Bearbeitungs-Hierarchien, z.B. Redakteur und Chef-Redakteur, bzw. Sachbearbeiter und Marketing-Leiter, bei der Erstellung der zu publizierenden Ausgabe gemeinsam. Die Dokumente durchlaufen einen bestimmten Workflow, bevor sie nach außen gegeben werden.

Unterschiede zwischen Print- und Onlinemedien liegen vor allen Dingen in der Art der Dokument-Aufbereitung. So zeichnen sich Online-Dokumente oft durch eine größere Stückelung in kleine bildschirmseitengerechte Dokumente aus. Weiterhin bietet die bislang vorrangig genutzte Dokumentbeschreibungssprache (HTML, siehe Seite 42) nicht die aus dem Print-Bereich (SGML) [Gol90] bekannten Formatierungs- und Layout-Eigenschaften in vollständiger Art und Weise an. Schließlich stellen die Hyperlinks, die im WWW-Umfeld einzelne Dokumentfragmente zu einem großen Ganzen verbinden, eine aus dem Print-Bereich nur in Form von Referenzen gekannte Besonderheit dar. Insbesondere ihrem Management kann eine komplette Anwendungspalette gewidmet werden.

Die Art und Weise wie eine Web-Präsenz gelesen wird, unterscheidet sich im Allgemeinen auch stark von der im Normalfall sequentiellen Leseweise eines Printmediums. Gemäß seiner Interessen folgt der Nutzer den die Dokumente verknüpfenden Hyperlinks und bewegt sich dementsprechend ungeordnet durch den Dokumentenbestand. Für die Erstellung der Dokumente einer Web-Präsenz bedeutet dies eine im Vergleich zu den Print-Dokumenten tendenziell größere Eigenständigkeit. Jedes Dokument steht bereits für sich und ist nur bedingt auf den Kontext, d.h. beispielsweise den Abschnitt davor oder danach, angewiesen. Sehr deutlich wird dies beim gezielten Zugriff auf Informationen über Suchmaschinen. Hier werden basierend auf Schlagworten einzelne Dokumente aufgelistet, die von dem Benutzer zumeist ohne große Einblicknahme in den Kontext gelesen werden. Wie stark ausgeprägt die Autonomie der Dokumente im Print- bzw. Onlinemedium ist, hängt im Einzelfall von der Art der Dokumente ab.

Ein weiterer Unterschied mit gravierenden Folgen für die Vorgehensweise beim Publizieren stellt die hohe Dynamik der Onlinemedien dar. Dokumente können direkt publiziert werden, ohne den Umweg über eine Druckerei. Während sich bei den Printmedien ein „Redaktionsschluss“ etabliert hat, ist eine Web-Präsenz nie fertig, braucht es aber auch nicht zu sein, da sie jederzeit modifiziert werden kann und muss. Dies gilt insbesondere auch für die publizierten Dokumente, denn nur wenn sie ständig aktualisiert werden, sind zufriedenstellende Zugriffszahlen zu erwarten. Die Workflows für die Erstellung und Publikation dieser Dokumente müssen dementsprechend auf eine zyklische Überarbeitung ausgelegt sein. Sie kann dabei entweder bei Bedarf manuell angestoßen oder nach definierten Zeiträumen automatisch in die Wege geleitet werden.

2.1.1 Arten von Web-Präsenzen

Ähnlich wie der Begriff Printmedium gibt auch der Begriff Onlinemedium, in diesem Fall Web-Präsenz, noch keine genaue Auskunft um was für eine Publikationsform es sich letztendlich handelt. Wie beim Printmedium zwischen Buch, Zeitung, Katalog, Broschüre etc. unterschieden wird, so kann prinzipiell auch bei Web-Präsenzen zwischen verschiedenen Arten unterschieden werden.

Mit dem Begriff Web-Präsenz (vgl. die „physikalische“ und die informationsbasierte Definition vom „Online Computer Library Center“(OCLC) [O'N99]) wird im Allgemeinen eine WWW-basierte Veröffentlichung von Informationen bezeichnet. Dabei werden neben

Personen-, Adress- und Produktinformationen mehr und mehr transaktionsorientierte Daten (E-Commerce) vermittelt. Eine Klassifizierung einer Web-Präsenz analog zu den oben genannten Printmedien ist nur schwer möglich. In Anlehnung an das Printmedium kann zwar von einer „Online-Zeitung“ gesprochen werden, dieser Begriff ist jedoch für die Charakterisierung der Web-Präsenz aus technischer Sicht heraus noch vergleichsweise unzureichend, da auch eine Unternehmens-Web-Präsenz nach der gleichen Vorlage aufgebaut werden kann. Differenzierungen hängen somit eher an dem verfügbaren Dokumentenbestand, der Art und Aufbereitung der Dokumente etc. Im MultiMedia-Journal der IEEE [GM01] wird in der Einleitung zu einer Sonderausgabe zum Web-Engineering [Gro01a, Gro01b] eine aktuelle Einschätzung zur Kategorisierung von Web-Präsenzen, in dem Kontext des Web-Engineerings als Webapplikationen betrachtet, gegeben. Sie liefert, wie in Tabelle 2.1 dargestellt, einen Überblick über die Vielfalt der Nutzungsmöglichkeiten der WWW-Technik, macht aber auch deutlich, wie schwer es ist, eine konkrete Web-Präsenz exakt einzuordnen, da in den meisten Fällen eine Kombination verschiedener Kategorien gegeben ist. Von daher favorisiert diese Arbeit die

Kategorie	Beispiel(e)
Informierend	Online-Zeitung, Produkt-Katalog, Nachrichten, Bedienungsanleitungen
Interaktiv	(Benutzer-bereitgestellte Informationen oder auf den Benutzer angepaßter Zugang) Anmeldeformulare, personalisierte Informationen (-Darstellung), Online-Spiele
Transaktionell	E-Commerce, Warenbestelldienste, Online-Banking
Workflow	Online-Planung und Terminverwaltung, Inventar- und Statusüberwachung
Kooperative Arbeitsumgebungen	Verteilte Autoren-Systeme, Kooperative Design-Werkzeuge
Online-Gemeinden, Marktplätze	(Diskussions-)Foren, Online-Marktplätze und Auktionen
Portale	Elektronische Einkaufscenter, Online Vermittler

Tabelle 2.1: Kategorien von Web-Präsenzen mit Beispielen

gröbere, nicht auf die spezifische Ausprägung ausgerichtete, aber mehr auf den Begriff Web-Präsenz abzielende Einteilung in die „Homepages“ und die „Portale“. Der Übergang zwischen diesen beiden Formen ist auch hier fließend und auch umfassende Begriffsklärungen wie die von Bestgen et al. in [BMS01] macht hier keine eindeutigen Aussagen. Bei entsprechender Ponderierung der Faktoren erscheint eine klare Zuordnung einer Web-Präsenz zu der einen oder anderen Art fast immer möglich.

2.1.1.1 “Homepages“

Als „homepage-artige“ Web-Präsenzen können alle diejenigen Web-Präsenzen zusammengefasst werden, die als Internet-Repräsentation, beispielsweise einer Person, eines Unternehmens oder einer Behörde, dazu gedacht sind, deren Interessen zu vertreten. Mit der „Homepage“ möchte der Betreiber seinen Platz im Internet sichern und damit Informationen zu bestimmten Interessensgebieten offerieren. Mit der geeigneten technischen Ausstattung sind

auch geschäftliche Transaktionen möglich. Diese bringen einen entsprechend hohen Anteil von dynamischen Elementen auf die Web-Präsenz, stellen aber in den meisten Fällen einen mehr oder weniger eigenständigen Abschnitt der Web-Präsenz dar, oft in einem eigenen Fenster visualisiert.

2.1.1.2 Portale

Selbstverständlich haben auch die Betreiber von Portalen und Suchmaschinen Interessen, die durch ihre Web-Präsenz vertreten werden sollen. Der Unterschied zu den oben als „homepage-artig“ bezeichneten Web-Präsenzen lässt sich demnach in der Theorie nur an Kleinigkeiten festmachen. So liegt es beispielsweise nicht in ihrem ureigenen Interesse, selbstständige Inhalte zu präsentieren. Vielmehr liegt ihr Hauptbetätigungsfeld in der Vermittlung der Information, wo bestimmte Information im Internet zu finden ist. Ihr Metier sind somit eher sogenannte Meta-Informationen. Durch die Kombination aus in Verzeichnissen sortierten Zugängen zu anderen Web-Präsenzen mit der Möglichkeit, eine gezielte Suche danach zu starten, stellen sie in vielen Fällen den initialen Anlaufpunkt bei dem Start einer Internet-Recherche dar. Benutzer verwenden sie, um Einstiegspunkte in ihr Interessensgebiet zu finden. Die eigentlichen Informationen werden ihnen dann von den „homepage-artigen“ Web-Präsenzen bereitgestellt.

2.1.2 Ziele

Wie bereits aus dem Vorangegangenen ersichtlich geworden ist, werden beim Publizieren im Internet zum großen Teil äquivalente Ziele wie im Printbereich verfolgt. Im Unterschied zum Printmedium ist das Onlinemedium WWW dabei um ein vielfaches flexibler und dynamischer, so dass daraus Möglichkeiten resultieren, die für Printmedien nicht realisierbar sind; nicht zu vernachlässigen ist jedoch die Benutzbarkeit der entstehenden Web-Präsenz die sich in den folgenden Zielen wiederspiegelt und durch entsprechende Tests belegt werden sollte [HS00].

2.1.2.1 Qualität

Wie im Printbereich stellt die Qualität eines der vorrangigsten Ziele dar. Im WWW-Umfeld ist die Qualität eines Angebotes so wichtig, weil der Besucher einer Web-Präsenz ohne großen Aufwand auf eine andere Web-Präsenz (z.B. die der Konkurrenz) wechseln kann. Anders als im Print-Bereich, wo es dem Leser nicht möglich ist, ohne weiteres auf ein Konkurrenzprodukt auszuweichen, reicht wie oben erwähnt im WWW ein Klick mit der Maus, um auf eine Konkurrenz-Web-Präsenz zu gelangen. Neben der Qualität der Inhalte entscheidet aber auch in großem Maße die Qualität des Designs darüber, ob eine Web-Präsenz vom Nutzer angenommen wird. Insbesondere die Funktionalität des Designs ist von ausschlaggebender Bedeutung. Neben einer hohen Ladegeschwindigkeit der einzelnen Dokumente ist auch auf die Navigationsstruktur zu achten. Sind die von dem Benutzer benötigten Informationen auf der Web-Präsenz nicht einfach zugänglich, so ist es unwahrscheinlich, dass sie von ihm in absehbarer Zeit erneut besucht wird. Insbesondere bei Web-Präsenzen, die als Benutzerschnittstelle für ein Online-Business verwendet werden, kann dies fatale Folgen auf den Umsatz haben.

2.1.2.2 Aktualität

Da WWW-Angebote zu jeder Zeit von den Benutzern abgerufen werden können, ist es wichtig, aktuelle Informationen so zeitnah wie möglich, d.h. im Prinzip rund um die Uhr, zu publizieren. Wenn der Benutzer der Web-Präsenz den Eindruck bekommt, die Dokumente, die ihm angeboten werden, sind nicht auf einem aktuellen Stand, wird er sich zum einen nach alternativen Angeboten umsehen; dies ist gerade im WWW-Umfeld bei dem Überangebot von

Suchmaschinen und Portalen vergleichsweise einfach. Zum anderen, selbst wenn der Besucher nicht nach alternativen Angeboten sucht, wird er die Frequenz mit der er die Web-Präsenz besucht, verkleinern und den negativen Eindruck auf das Image des Unternehmens transferieren. Beide Fälle wirken sich negativ auf die Besucherstatistiken, welche ein wertvolles Maß für die Akzeptanz sind, und damit im kommerziellen Umfeld auch den finanziellen Erfolg (z.B. Werbeeinnahmen), aus.

2.1.2.3 Quantität

Da eine Web-Präsenz bezüglich des verfügbaren Datenspeicherplatzes im Vergleich zum Printmedium (z.B. Zeilenbeschränkung bei Zeitungsartikeln) nahezu unbeschränkt ist, sollte eines der Ziele sein, die Zahl und den Umfang der publizierten Dokumente an das Medium Internet anzupassen. Dazu gehört z.B. die Aufspaltung von einzelnen Dokumenten in kleinere Unterdokumente, die einfacher, d.h. ohne zu scrollen, im Web-Browser dargestellt werden können. Weiterhin können Dokumente in Abhängigkeit der Signifikanz ihres Inhalts auch in Form eines Archivs interessierten Besuchern auf der Web-Präsenz zur Verfügung gestellt werden.

2.1.2.4 Struktur

Die Dokumente einer Web-Präsenz können nur abgerufen werden, wenn sie von Benutzern „gefunden“ werden. Mit einer einfachen und klaren inhaltlichen Struktur einer Web-Präsenz, die dem Besucher auch visuell zugänglich ist, wird es ihm ermöglicht, gezielt innerhalb der Web-Präsenz zu bestimmten Angeboten zu navigieren. Unabhängig von einer solchen Navigationsstruktur besteht für die Dokumente die Forderung nach möglichst sinnvoller Verknüpfung untereinander. Im Vergleich zum Printmedium ist dieser Punkt besonders entscheidend, da auf den Web-Präsenzen der Einsatz von Inhaltsverzeichnissen nicht in der Form zum Erfolg führt, wie er bei Printmedien gewohnt ist. Während der Leser eines Buches oder einer Broschüre selbstverständlich bei der Suche nach einer bestimmten Information das Inhaltsverzeichnis konsultiert, ist der erste Versuch des Internet-Nutzers die Bemühung, die gesuchte Information aufgrund ihres Platzes in der logischen Struktur der Web-Präsenz zu lokalisieren. Erst in einem zweiten Schritt wird von der Suchmaschine der Web-Präsenz gebrauch gemacht.

2.1.2.5 Grad der „Verlinkung“

Wie bereits besprochen, baut sich das World Wide Web durch Verknüpfung aller veröffentlichten Dokumente untereinander auf. Dem Benutzer wird dadurch die Chance gegeben, sich neben den „vorgefertigten Pfaden“, die aus der oben genannten Struktur einer Web-Präsenz erwachsen, seine eigenen Wege durch die Informations-Ressourcen zu bahnen, die sich aus seiner eigenen Interessenlage ergeben. Beim Publizieren im Internet ist deshalb eines der Ziele, einen hohen Grad an Dokument-Verknüpfungen zu erreichen. Je nach Interessenlage kann es sinnvoll sein, vorrangig innerhalb der eigenen Web-Präsenz zu verknüpfen, um den Besucher nicht auf fremde Web-Präsenzen abgleiten zu lassen (Werbeeinnahmen). Andererseits sollten alle relevanten Referenzen, die dem Leser/Benutzer das Verständnis des Dokumentes erleichtern, vorhanden sein. Selbstverständlich kann ein Übermaß an Links den Benutzer aber auch irritieren und das Verständnis des Dokumentes behindern [Fre97]. Folgt der Benutzer allen ihm angebotenen Verknüpfungen, so ist die Gefahr des „Verzettelns“ groß. Er entfernt sich schnell von den ihn ursprünglich interessierenden Dokumenten und kann sich dann nur an der von der Web-Präsenz gebotenen Struktur orientieren. Ist diese nicht hinreichend, so erliegt der Benutzer dem „Lost in Space Syndrom“ [Con87], er ist orientierungslos,

weiß nicht mehr, von wo er kam und was eigentlich sein Ziel war. Folglich sind manchmal weniger, dafür aber hochwertige Verknüpfungen mehr als eine Unmenge nur bedingt passender.

2.1.2.6 Interaktivität

Eng verbunden mit diesem Punkt ist die Interaktivität, die eine Web-Präsenz bieten sollte – eine Option, die bei dem Printmedium nicht vorhanden ist. Dem Benutzer sollte der Eindruck vermittelt werden, dass er nicht nur vorgefertigte, allgemeine Inhalte abruft, sondern speziell auf ihn angepasste. Damit dies möglich ist, sind neben den interaktiven Elementen, die clientseitig durch Skripte oder Programme ermöglicht werden, i.a. Verbindungen in das interne Netzwerk des Unternehmens notwendig. Aus internen Datenbanken können die Benutzerdaten als Schlüssel für die Auslieferung spezieller Informationen betrachtet werden. Beispielsweise lassen sich Preise anwenderbezogen gestalten. Dementsprechend bieten die Web-Präsenzen, die transaktionell ausgerichtet sind, d.h. Waren verkaufen, einen hohen Grad von Interaktivität. Interaktivität kann aber auch heißen, dem Benutzer die Möglichkeit zu offerieren, zu bestimmten Inhalten Stellung zu nehmen. Auf spezielle Themen des Unternehmens ausgerichtete Diskussionsforen sind beispielsweise eine gute Gelegenheit für die Kunden, sich untereinander auszutauschen. Das Unternehmen erfährt auf diese Weise eine Menge über die Probleme und Vorlieben der Kunden. Damit Diskussionsforen eine langfristige Überlebenschance haben, sind Qualitätssicherungsmaßnahmen wie von Zeit zu Zeit neu eingebrachte Diskussionsimpulse durch einen Moderator sehr hilfreich.

Mit Interaktivität verbunden sind auch dynamische Elemente auf den publizierten Dokumenten. Navigationsleisten mit einer dynamischen Menüstruktur, die bei Selektion ausklappt, hilft zum einen die Navigation zu erleichtern, zum anderen bekommt der Benutzer den Eindruck, dass die Web-Präsenz einen „Programmcharakter“ besitzt. In Verbindung mit der Möglichkeit, individuelle Informationen abzurufen, z.B. Prämienberechnungen, Einkaufslisten, Annoncen etc., steigt der Mehrwert, der für den Benutzer durch den Besuch der Web-Präsenz entsteht.

2.1.2.7 Sicherheit

Direkt mit der Dynamik einer Web-Präsenz verbunden, aber natürlich nicht allein dadurch initiiert, ist die für das Unternehmen entstehende Gefahr von „Angriffen“. Die Web-Präsenz eines Unternehmens stellt über das Internet das Tor zur großen weiten Welt dar. Durch dieses sollen Kontakte zustande kommen und Informationen verbreitet und eingesammelt werden. Der über diesen Kanal mögliche Datenaustausch zwischen dem Unternehmen und den Kunden stellt allerdings eine große Gefahr dar, da eben dieser Kanal auch zu Angriffen missbraucht werden kann. Diese Angriffe, die weiter unten noch ausführlicher diskutiert werden, können mit verschiedenen Zielsetzungen verbunden sein, beispielsweise Datendiebstahl oder -zerstörung. In jedem Fall ist neben dem materiellen Schaden durch die Datenverluste der immaterielle Verlust an Image und Vertrauen in der Öffentlichkeit bei Bekanntwerden eines solchen Vorfalles groß.

Aus diesem Grund sind bei dem Betrieb einer Web-Präsenz Sicherheitskonzepte im technischen Umfeld unumgänglich. Je näher die Web-Präsenz an das Unternehmen heranwächst, bzw. je stärker sie integriert wird, um so besser müssen die Sicherheitsmaßnahmen greifen, die die Datenströme überwachen.

Im Vergleich zum Printmedium ist das WWW somit nicht nur mit einem großen Potential verbunden. Probleme, die beim Printmedium durch die Abkopplung des Unternehmens vom direkten Kontakt mit dem Kunden nicht vorhanden sind, können dem Mehrwert der großen

Kundennähe einer Web-Präsenz abträglich sein, so dass in diesem Interessenkonflikt für den Betreiber einer Web-Präsenz nur der Einsatz umfassender Sicherheitskonzepte ratsam ist.

2.1.2.8 Einsatz offener Standards

Ziel beim elektronischen Publizieren im WWW ist es, eine breite Benutzerschicht, zumindest aber große Teile der Zielgruppe, zu erreichen. Diese definiert sich zum einen über die angebotenen Inhalte und ihre Aufbereitung, zum anderen aber auch über die technische Möglichkeit des Zugangs zu den Informationen. Selbst wenn beim Benutzer die Grundvoraussetzung des Zugangs zum WWW gegeben ist, ist unklar, ob er auf die Dokumente, die auf der Web-Präsenz publiziert sind, mit seinem Web-Browser zugreifen kann, denn nicht jeder Web-Browser kann alle Dokumentenformate darstellen, da es keinen uniform genutzten Dokumentenstandard für das WWW gibt. Während das Kommunikationsprotokoll als übergreifend betrachtet werden kann, sind die Dokumentbeschreibungssprachen mittlerweile mit so vielen herstellerspezifischen Erweiterungen versehen, dass sich dem Betreiber einer Web-Präsenz ernsthaft die Frage stellt, welche Variante er unterstützen möchte. Sobald hier die offenen Standards, in den meisten Fällen eine Schnittmenge der von allen Web-Browsern unterstützten Elemente, das Ziel sind, sind neben Komforteinbußen im Umgang mit der Web-Präsenz auch durch die mangelhafte Implementierung der Standards Probleme zu erwarten. Nichtsdestotrotz sind offene Standards eines der großen Ziele, da nur sie eine lebendige Weiterentwicklung des WWW ermöglichen. Mit den aktuell aufkommenden Datenbeschreibungssprachen auf XML-Basis wird wieder ein Schritt in die Richtung der Standardisierung unternommen. In wie weit die Unterstützung dieser Techniken in absehbarer Zeit gegeben ist, muss sich noch zeigen. Der Standard HTML als Dokumentbeschreibungssprache im WWW wird allein aus Kompatibilitätsgründen aufgrund der breiten existierenden Dokumentenbasis, deren umfassende Konvertierung nicht zu erwarten ist, auch zukünftig noch unterstützt werden.

2.1.2.9 Medienneutrale Dokumentenverwaltung

Nicht für alle Medien, die das WWW nutzen, gelten die gleichen Anforderungen. Während die aktuellen Web-Präsenzen vorrangig für die Darstellung an PC-Arbeitsplätzen optimiert sind, welche eine vergleichsweise hohe Bildschirmauflösung und Rechenleistung bieten, bleiben andere Medien noch weitgehend unberücksichtigt. Beispielsweise stellt der Fernseher kombiniert mit einer „Set-Top-Box“, welche die Aufgaben des PCs übernimmt, ein großes Potential dar. Die Auflösung und Rechenleistung ist aber bislang der eines PCs deutlich unterlegen. Noch deutlicher wird dieser Unterschied bei den aktuellen Mobilgeräten. Die winzigen Displays verhindern das Browsen auf PCs ausgerichteter Web-Präsenzen, selbst wenn diese eins zu eins in das von den Mobilgeräten darstellbare Format „Wireless Markup Language“ (WML) [FBM+00] konvertiert werden. Benötigt wird somit eine medienneutrale Verwaltung der Dokumente, genauer ihrer Komponenten, mit einem anschließenden Export für das entsprechende Zielmedium. Beispielsweise spricht wenig dagegen, einzelne Verzeichnisse einer Web-Präsenz als Printausgabe zusammenzufassen, oder umgekehrt, die für die Zeitung präparierten Artikel internetgerecht aufbereitet parallel auch in der Online-Zeitung zu veröffentlichen. Wie oben beschrieben, steht aber im Normalfall die unterschiedliche Philosophie des Onlinemediums einer vollautomatisierten Aufbereitung aus dem Printmedium-Dokumentenbestand im Wege.

2.1.3 Benötigte Infrastruktur

Mit dem Einzug der Digitaltechnik und der verstärkten Nutzung der modernen Technologien unterscheiden sich die Infrastrukturen für das Online-Publizieren und das Printmedium nur noch in wenigen Punkten. Davon absehend, dass sowohl die Eingabekomponenten als auch die Verwaltungskomponenten auf die jeweilige Zielrichtung optimiert sind, unterscheiden sich die beiden Publikationsmedien hauptsächlich in der Ausgabekomponente. Im Publikationsprozess kommen fundamental voneinander abweichende Mechanismen zum Tragen. Für das Printmedium ist die Druckmaschine mit allen dazugehörigen Prozessen die zentrale Ausgabekomponente. Nach dem Druck wird die Verpackung und Distribution des Druckerzeugnisses in die Wege geleitet, ein Vorgang, der von der Vertriebsabteilung des Unternehmens mit großem logistischen Aufwand durchgeführt werden muss.

Analog zu der fertigen Ausgabe aus der Druckmaschine wird beim Online-Medium die Web-Präsenz durch eine Export-Komponente (Analogon zu der Druckmaschine) auf den Rechner gespielt, der mit dem Internet verbunden ist. Lokalisiert im Unternehmen oder bei einem Internet-Service-Provider (ISP) übernimmt dieser Rechner mit dem auf ihm arbeitenden Web-Server-Programm die Verteilung der Dokumente im WWW. Abgesehen vom Medium selbst ist einer der großen Unterschiede bei diesen Publikationsverfahren die Geschwindigkeit, mit der die Resultate veröffentlicht werden können. Da bei dem Onlinemedium die Verpackung und Distribution entfallen, sind die so publizierten Inhalte sofort mit dem Aufspielen auf den Web-Server im WWW verfügbar.

2.2 Vorgehen beim Aufbau einer Unternehmens-Web-Präsenz

Wie ein Unternehmen beim Aufbau einer Web-Präsenz vorgeht, wird sich aufgrund der verschiedenartigen Voraussetzungen von Fall zu Fall unterschiedlich gestalten. Im Rahmen dieser Arbeit wird exemplarisch eine mögliche Vorgehensweise beleuchtet, deren Ziel der produktive Betrieb einer Web-Präsenz in einem Unternehmen ist. Einem chronologischen Ablauf folgend wird dabei, ausgehend von einer Entscheidung der Unternehmensleitung eine eigene Web-Präsenz aufzubauen, in einem ersten Schritt mit der Zusammensetzung einer interdisziplinären Arbeitsgruppe begonnen, deren Aufgabe die Konzeption, der Aufbau und die Integration der Web-Präsenz ist.

2.2.1 Zusammenstellung der Arbeitsgruppe

Wie bei jedem Projekt ist die Zusammenstellung des Teams entscheidend für das Projekt-Ergebnis. Obwohl im konkreten Fall stark abhängig von der Zielsetzung, sollte sich das Team, das die Web-Präsenz konzeptioniert, durch einen hohen Grad von Interdisziplinarität auszeichnen, d.h. die Fachrichtungen Marketing, Grafik/Design, Informationsverarbeitung (IV) und Mitarbeiter aus den inhaltlich betroffenen Fachabteilungen abdecken.

2.2.1.1 Marketing

Da die Web-Präsenz in erster Linie als Marketing-Instrument eingesetzt wird, stellt die Marketing-Abteilung die Mitarbeiter, die über die Zielsetzung und die Ausrichtung der Web-Präsenz wachen. Mit dem technischen Grundverständnis ausgestattet, werden von ihnen die Marketingaspekte der Web-Präsenz in die Konzeption eingebracht. Schließlich ist die Marketing-Abteilung auch für die Definition der Zielgruppe zuständig und infolgedessen auch an der Überprüfung der Umsetzung der Konzepte durch Marktforschung beteiligt. Insbeson-

dere die Interpretation der Zugriffstatistiken (siehe S. 170) und die damit einhergehende Neuausrichtung der Web-Präsenz im laufenden Betrieb stellt ein wichtiges Aufgabenfeld dar.

2.2.1.2 Fachabteilungen

Inhaltliche Fragen werden zwar von der Marketing-Abteilung auf die Relevanz für die mit der Web-Präsenz verknüpften Ziele überprüft, die eigentliche Zusammenstellung und die Strukturierung der Informationen wird jedoch von den Mitarbeitern der Fachabteilungen geleistet. Ihnen sind die fachspezifischen inhaltlichen Zusammenhänge klar, und sie verfügen über die Informationsquellen, die die Web-Präsenz mit Leben füllen sollen. Innerhalb ihrer Kompetenzbereiche können sie entscheiden, welche Priorisierung für die einzelnen Produkte und Daten angemessen ist. Ihre Arbeit liefert somit die inhaltlichen Werte der Web-Präsenz.

2.2.1.3 Grafik/Design

Die von der Marketing-Abteilung und den Mitgliedern der Fachabteilungen angedachte inhaltliche Struktur wird von den Mitarbeitern der Grafik/Design-Agentur gestalterisch umgesetzt. Sie erarbeiten die Design-Studien, anhand derer entschieden wird, wie die Inhalte verpackt werden sollen. Gemeinsam mit dem Marketing und den Fachabteilungen wird versucht, ein gestalterisches Konzept zu finden. Dieses muss sowohl den speziellen gestalterischen als auch den inhaltlichen Anforderungen gerecht werden. Neben der Corporate Identity (CI) muss die Art der zu publizierenden Informationen in die Konzeption eingehen.

2.2.1.4 IV-Abteilung

Optimalerweise von Anfang an in das Team integriert überwachen die Mitarbeiter der IV-Abteilung die Umsetzbarkeit der diskutierten Ideen. Umsetzbarkeit ist hierbei vor allem aus technischer Sicht zu verstehen. Fragen der Anbindung von internen und externen Datenquellen müssen von ihnen berücksichtigt werden. Ihre Kenntnis der internen Infrastruktur hilft, viele Probleme vor deren Entstehen schon abzufangen.

2.2.2 Analyse der Anforderungen

Steht das Projektteam, so beginnt die konkrete Konzeption der Web-Präsenz mit der Analyse der Anforderungen. Konkrete Anforderungen an die Web-Präsenz und das einzusetzende Instrumentarium beruhen i.a. auf unternehmensstrategischen Entscheidungen. Solche Entscheidungen betreffen u.a. die Corporate Identity (CI) bzw. das Corporate Design (CD) des Unternehmens, die Ausrichtung der Web-Präsenz, die einzusetzenden Basis-Software-Komponenten (z.B. Datenbank) und die Entscheidungshierarchien bzw. Rollen, die für den Betrieb abgebildet werden müssen. Die einzelnen Anforderungen müssen in dieser Phase identifiziert und in ihrer Relevanz priorisiert werden.

Während es für ein Unternehmen, das sich aus Wettbewerbsgründen im Internet präsentieren möchte, hinreichend ist, wenn aktuelle Informationen im Tagesrhythmus veröffentlicht werden, ist es für eine Online-Zeitung unter Umständen schon schädlich, wenn Aktualisierungen nur im Stundenrhythmus durchgeführt werden. In Abhängigkeit von solchen konkreten Anforderungen ist die Verwaltung zu konzeptionieren und dementsprechend die Web-Präsenz aufzubauen. Die im Folgenden aufgeführten Szenarien geben einen kleinen Eindruck, welche Anforderungen miteinander konkurrieren können.

2.2.2.1 Online-Zeitung

Eine Online-Zeitung lebt zu einem gewissen Teil von den veröffentlichten Werbe-Anzeigen [KM99, Len00, Bal97]. Abgerechnet wird in den meisten Fällen über die Zahl der Seitenabrufe. Damit die aktuell eingesetzte Zählsoftware der „Informationsgemeinschaft zur Feststellung der Verbreitung von Werbeträgern e.V." geeignete Zahlen ermitteln kann, müssen die Dokumente mit speziellen Zusatzinformationen versehen werden, die die Zählsoftware zur Auswertung verwendet.

Bei den gängigen Online-Zeitungen wird in den meisten Fällen zwischen den ausführlicher recherchierten Artikeln und den eher „ad hoc-Status" besitzenden Tickermeldungen unterschieden. Während die Tickermeldungen möglichst sofort publiziert werden, ist es bei den Artikeln hinreichend, einen dem Redaktionsschluss entsprechenden Zeitpunkt zu setzen, nach dem der Publizierungsmechanismus (z.B. täglich) in Gang gesetzt wird.

Es bietet sich bei einer Online-Zeitung an, den in dem Printbereich etablierten Workflow auf das Onlinemedium zu übertragen. Die Journalisten und Redakteure schreiben die Artikel in den ihnen zugeordneten Ressorts. Der Chefredakteur autorisiert die Veröffentlichung bzw. ordnet die Überarbeitung einzelner Artikel an und realisiert auf diese Weise ein Vier-Augen-Prinzip.

Da bei dem Verlag die Artikel für das Printmedium vorhanden sind, sollen Teile davon auch für das Onlinemedium genutzt werden. Aus diesem Grund werden im Zeitungsumfeld spezielle Datenfilter benötigt, welche die für das Printmedium formatierten Datensätze für das Onlinemedium aufbereiten.

2.2.2.2 Bank/Versicherung

Bei Banken und Versicherungen ist häufig die Web-Präsenz in zwei Bereiche unterteilt. Zum einen gibt es den eher informativen Bereich, in dem sich das Unternehmen mit seinen Produkten vorstellt, zum anderen gibt es den transaktionellen Bereich, in dem mit den Produkten gehandelt werden kann. Während letzterer in der Web-Präsenz dynamische Arbeitsabläufe bereitstellt und deshalb von seiner Konsistenz her als Programm-Code betrachtet werden kann, handelt es sich bei ersterem um eine von der Anlage her statische Web-Präsenz. Bei diesen statischen Seiten können dynamische Elemente, z.B. Grafiken mit Börsenkursen, realisiert durch die Einbindung externer Datenquellen, in Form aktueller Informationen in Echtzeit integriert werden. Die rahmengebenden Dokumente hingegen benötigen im Normalfall nur eine seltenere Aktualisierung.

Damit keine unternehmenskritischen Daten im Internet veröffentlicht werden, verlangt das Banken- und Versicherungsumfeld nach mehrstufigen Abzeichnungsverfahren. So kann durchaus ein Sechs- oder Mehr-Augen-Prinzip eingeführt werden. Weiterhin sind bei Banken und Versicherungen in großem Umfang Datenbanken im Einsatz. Um eine unwartbar heterogene Datenbanklandschaft im Unternehmen zu vermeiden, werden strategische Plattformen definiert, welche dann auch für den Betrieb der Web-Präsenz genutzt werden müssen. Insbesondere bei Banken im grenznahen, steuerlich begünstigten Luxemburg ist die Zielgruppe, die mit der Web-Präsenz angesprochen werden soll, internationalen Charakters. Demzufolge sind viele Dokumente in den wichtigsten Sprachen vorzuhalten (Multilingualität) und miteinander zu verknüpfen.

2.2.2.3 Instituts-Präsenz

Andere anforderungen stellen die Internetdarstellungen kleinerer Unternehmen u.a. aus dem Hochschulumfeld, z.B. die des Instituts für Telematik, dar. Hier wirken sich flache Hierarchien

in der Unternehmensstruktur auch auf die inhaltliche Arbeit und Pflege der Web-Präsenz aus. Während bestimmte Teile der Präsenz über ein zweistufiges Abzeichnungsverfahren gepflegt werden, ist der große Rest dem direkten Zugriff der Mitarbeiter unterstellt.

Aktualität ist in diesem Umfeld allerdings differenzierter aufzufassen als bei den vorangehenden Beispielen. Aktuelle Ereignisse sind hier Veranstaltungen, z.B. Symposien oder Kolloquien, News betreffen z.B. Konferenzteilnahmen oder neue prototypische Entwicklungen – alles Punkte, bei denen eine regelmäßige, aber nicht zwingend tägliche Aktualisierungsrate hinreichend ist. Ist ein Unternehmen in seiner Arbeit inhaltlich mit den Internet-Technologien verbunden, wird die eigene Web-Präsenz in besonderem Maße zum Aushängeschild. In diesem Fall sollte die Web-Präsenz zum einen durch ein solides Auftreten gefallen, zum anderen aber zeigen, dass das Unternehmen hoch innovativ ist. Der Mut zur Innovation kann sich in der Nutzung neuester, noch in der Erprobung befindlicher Techniken niederschlagen. Entsprechende Beispiele auf der Web-Präsenz, bzw. ihr Einsatz beim Aufbau der Web-Präsenz demonstrieren diese Kenntnisse und Fähigkeiten nach außen.

2.2.2.4 Freiheitsgrade: Die Web-Präsenz als Individuallösung?

Bereits nach diesen wenigen Beispielen drängt sich die Frage auf, ob für jede Web-Präsenz individuelle Anforderungen gestellt werden. In der Tat ist es so, dass je nach Branche, in der ein Unternehmen tätig ist, sich die Anforderungen unterscheiden. Weiterhin spielt die Größe des Unternehmens und damit auch der Umfang der Web-Präsenz eine Rolle. Je größer ein Unternehmen ist, desto eher ist aufgrund der finanziellen Möglichkeiten die Umsetzung individuellen Auftretens möglich. Strebt ein Unternehmen mit der Web-Präsenz nach neuen Geschäftsmodellen und möchte in dem neu entstehenden Markt zusätzliche Umsätze generieren, steigen die Individualisierungsbestrebungen nochmals, da dann außerdem noch eine Anpassung an die internen Geschäftsprozesse erfolgen muss.

2.3 Die Frage nach dem Werkzeug – Motivation für die Eigenentwicklung des Online-Redaktionssystems

Wie in der Einleitung beschrieben, gibt es für das Unternehmen zwei verschiedene Wege, seine Web-Präsenz zu betreiben: entweder sie wird an eine Agentur ausgelagert oder – dies ist das Szenario, das in dieser Arbeit gewählt wurde – sie wird in das Unternehmen integriert. In diesem letzteren Fall stellt sich nach der inhaltlichen und strukturellen Konzeption der Web-Präsenz in einem nächsten Schritt die Frage nach der technischen Realisierung. Die Diskussion von Web-Präsenzen z.B. im Unterschied zum Printmedium haben verdeutlicht, dass es sich bei einer Web-Präsenz um ein „lebendiges“ Objekt handelt bzw. handeln sollte: regelmäßige Aktualisierungen bei gleichzeitig hoher Qualität sind zur Bindung eines Kundenstammes unerlässlich.

Dementsprechend reicht es nicht, ein Werkzeug zum Aufbau der Web-Präsenz zu verwenden, das für ihre spätere Pflege und Aktualisierung ungeeignet ist. Erfahrungen mit den vielen im WWW verfügbaren Web-Präsenzen zeigen in diesem Zusammenhang, dass der initiale Aufbau der Web-Präsenz zwar aufwendig, prinzipiell aber unkritisch ist und erst ihr regelmäßige Aktualisierung bzw. Konsistenzhaltung die eigentlichen Probleme mit sich bringt. Arbeiten viele Personen parallel an dem Dokumentenbestand der Web-Präsenz, so existiert neben eventuellen Zugriffskonflikten inhärent die Gefahr von Inkonsistenzen. Abgesehen von inhaltlichen Stilbrüchen, die durch die unterschiedlichen Schreibstile der Autoren zustandekommen, sind insbesondere die „technischen Inkonsistenzen“ aus der Sicht des Web-Präsenz-Managements problematisch. Unmittelbar ins Auge fallen die formatierungsbedingten Inkonsistenzen be-

ruhend auf den „gestalterischen Freiheiten“ der Autoren. Im Ergebnis entsteht unversehens eine Web-Präsenz, bei der auf den ersten Blick an den Dokumenten die persönlichen Noten der jeweiligen Autoren zu erkennen sind – ein Zustand der im Normalfall nicht gewünscht wird, da im Allgemeinen ein kohärentes Aussehen dem Gesamteindruck der Web-Präsenz förderlich ist. Das für die Verwaltung der Web-Präsenz eingesetzte System muss diese Freiheit der Autoren auf ein Mindestmaß reduzieren, ohne diese in ihrer Kreativität zu stark einzuschränken.

Eine zweite Inkonsistenzproblematik ergibt sich aus der Verknüpfung von Dokumenten durch Hyperlinks (siehe Seite 44). Arbeiten mehrere Personen simultan an der Web-Präsenz und entfernen Dokumente, benennen bestehende um oder verschieben diese, sind Inkonsistenzen eher die Regel als die Ausnahme. Anders als beim Printmedium die Referenzen und Fußnoten, bauen die Verknüpfungen im WWW ein hochgradig dynamisches Netz auf. Während die Referenzen im Printbereich von Bestand sind, stellen die Verknüpfungen von Online-Dokumenten aufgrund deren eingeschränkter Lebensdauer immer nur temporäre Zuordnungen dar.

Besonders kritisch, weil sehr augenscheinlich, sind Hyperlink-Inkonsistenzen in der Navigationsstruktur. Werden keine Frame-Sets (siehe Seite 59) verwendet, so ist die Navigation jeweils direkt in die Dokumente codiert. Ändert sich ihre Struktur, müssen alle Dokumente aus dem von der Änderung betroffenen Zweig der Struktur modifiziert werden – eine Aufgabe, die sich bei den in einer lebendigen Web-Präsenz häufig anfallenden Modifikationen auf Dauer aufwendig gestaltet.

Um eine Web-Präsenz dauerhaft attraktiv zu gestalten, werden hochwertige Inhalte benötigt. Üblicherweise ist ein gewisser Teil dieser Inhalte fix und muss demnach nur initial erstellt werden. In Abhängigkeit von der Zielrichtung der Web-Präsenz wird jedoch auch ein mehr oder weniger großer Anteil an aktuellen Inhalten erforderlich. Diese müssen aus den verschiedenen Quellen des Unternehmens zusammengeführt und für die Web-Präsenz aufbereitet werden – eine Aufgabe, die sich bei normalen Web-Präsenzen, oft auch wegen der Komplexität der zu publizierenden Inhalte und der damit für das Unternehmen verbundenen Konsequenzen kaum noch von einer Person allein oder einem kleinen Team (Online-Redaktion) bewältigen lässt. Eine Lösung dieser Problematik stellt die Verteilung der Erstellungs- und Pflegearbeit auf eine größere Gruppe von Mitarbeitern, ausgewählt nach fachlicher Kompetenz in dem jeweiligen Bereich, dar. Als negative Konsequenz dieser zeitgleichen Teamarbeit an ein und demselben Dokumentenbestand kommt es zu Reibungsverlusten, es sei denn die der Verwaltung der Web-Präsenz zugrundeliegende Technik regelt übergreifend die Zuständigkeiten der einzelnen Mitarbeiter und verhindert so Zugriffskonflikte. Anders als bei Hypermedia (siehe Seite 42) allgemein sind bei einer Web-Präsenz die Anforderungen aus kooperativer Sicht jedoch geringer, da die Web-Präsenz als Gesamt-Dokument zwar gemeinsam, jedes einzelne Teildokument aber im Normalfall von einem Autor allein erstellt wird.

Ein weiteres Problemfeld ergibt sich weiterhin aus der hohen Dynamik des Dokumentenbestandes; um eine aktuelle Web-Präsenz zu betreiben, ist es geradezu notwendig, dass sich alle Dokumente „im Fluss“ befinden. Für das Management dieses Dokumentenbestandes bringt das erheblichen Aufwand mit sich: Um die Konsistenz des publizierten Dokumentenbestandes nicht zu gefährden, werden Überarbeitungsprozesse nicht direkt an der publizierten Version, sondern an einer intern vorgehaltenen Variante ausgeführt. Der Versionsabgleich über entsprechende Workflows stellt eine zentrale Anforderung an die zur Verwaltung der Web-Präsenz einzusetzende Software dar.

2.3.1 Content-Management als Orientierungshilfe

Um dem skizzierten Umfeld gerecht zu werden, wird technische Unterstützung beim kooperativen Arbeiten an der Web-Präsenz benötigt – ein Web-Präsenz-Verwaltungs-System, als Web-Präsenz-Content-Management-System (WCMS) bezeichnet, das Workflows vom Sammeln der Informationen in den Fachabteilungen über die WWW-gerechte Aufbereitung in der Online-Redaktion bis hin zur Publikation auf der Web-Präsenz unterstützt. An dessen Fähigkeiten wird eine Reihe von Anforderungen gestellt, die denen eines Content-Management-Systems entsprechen. Darüber hinaus gibt es zusätzliche Anforderungen, die aus der speziellen Nutzung für das WWW erwachsen.

2.3.1.1 Content-Management

Der Einsatz eines Content-Management-Systems (CMS) in einem Unternehmen soll helfen, durch geeignete Organisation die Übersicht über vorhandene und neu zu erstellende Inhalte zu bewahren; Mitarbeiter finden damit schnell und sicher die benötigten Informationen; Verzögerungen durch nicht auffindbare Dokumente entfallen. Zu einem erfolgreichen Content-Management gehören medienneutrale Dokumente ebenso wie geeignete Indexierungs- und Archivierungsmechanismen.

2.3.1.1.1 Medienneutrale Dokumente

Die Verwaltung von Dokumenten in medienneutraler Form setzt voraus, dass es ein internes Format gibt, aus dem die für die entsprechenden Medien benötigten Formate generiert werden können. Damit die Dokumente in dem internen Format verwaltet werden können, ist es notwendig, dass entweder die Autoren der Dokumente diese bereits in dem intern benötigten Format bereitstellen (z.B. über Eingabemasken erzwungen) oder ein Importmechanismus existiert, der die Dokumente von denjenigen Formaten, die die Autoren verwenden, in das interne Format überführt.

2.3.1.1.2 Indexierung

Durch eine geeignete Indexierung von Dokumenten (vgl. z.B. [Rap01]) wird deren Auffindbarkeit gewährleistet, Suchalgorithmen erlauben den direkten Zugriff auf einzelne Dokumente oder deren Komponenten. Über die Indexierung ist weiterhin die Gruppierung der Dokumente nach inhaltlichen Gesichtspunkten möglich [LH01].

2.3.1.1.3 Zugriffsberechtigungen

Zum Content-Management gehört grundlegend die Verwaltung von Zugriffsberechtigungen: Informationen sind nur für diejenigen Personen zugänglich, für die sie bestimmt sind, Berechtigungen werden sowohl für die Erstellung und Modifikation von Dokumenten als auch den lesenden Zugriff überprüft. Ein anderer in diesem Kontext wichtiger Aspekt ist die Nachvollziehbarkeit der an einem Dokument vollzogenen Arbeitsschritte: das System muss in der Lage sein, Auskunft darüber zu geben, wer wann was für eine Modifikation an einem Dokument vorgenommen hat (Revisionssicherheit). Solche Mechanismen schützen die Konsistenz und die Qualität des Datenbestandes, ferner verhindern sie Missbrauch der Daten.

2.3.1.2 Komponenten eines CMS

Vereinfacht betrachtet, besteht ein CMS aus drei Basiskomponenten, die für die Hauptaufgaben des Content-Managements ausgelegt sind. Dabei handelt es sich um die Eingabe-, die

Verwaltungs- und die Exportkomponente.

2.3.1.2.1 Eingabekomponente

Die Eingabekomponente stellt die Benutzerschnittstelle dar. Sie erlaubt die Erstellung und Modifikation von Dokumenten durch die Autoren, stellt Beziehungen zwischen den Dokumenten her und ermöglicht das Editieren von Metadaten für die Dokumente.

2.3.1.2.2 Verwaltungskomponente

Eine eigenständige Verwaltungskomponente übernimmt die von der Eingabekomponente erstellten Dokumente mitsamt deren Metadaten, legt sie im internen Speicher (z.B. Datenbank) ab und wahrt ihre Konsistenz. Über die Vergabe von Zugriffsrechten und Verantwortlichkeiten wird der Zugriff auf die verwalteten Informationen geregelt.

2.3.1.2.3 Exportkomponente

Mit der Exportkomponente werden aus den in der Verwaltungskomponente zur Verfügung stehenden Inhalten die für die Publikation relevanten Teile herausgegriffen und entsprechend den Vorgaben des gewünschten Ausgabemediums aufbereitet. Die Selektion der zu publizierenden Informationen kann automatisiert, manuell oder kombiniert manuell mit automatischer Unterstützung erfolgen.

2.3.1.2.4 Kooperatives Arbeiten – Rollenbasierte Workflow-Komponente

Üblicherweise werden die drei Basiskomponenten eines CMS durch eine Workflow-Komponente ergänzt, da an der Verwaltung eine wachsende Zahl von Personen beteiligt ist und die Dokumente für den Weg von der Erstellung bis zur Publikation einem Workflow unterworfen sind. Sie koordiniert zwischen den Beteiligten – häufig basierend auf rollenbasierter Autorisierung – die kooperative Bearbeitung, die zeitgerechte Erstellung, Überarbeitung und Freigabe von Dokumenten während deren Lebenszyklus'.

2.3.1.3 Web-Anforderungen

Liegen die erklärten Ziele des Content-Managements vor allem in der Verwaltung der Inhalte einer Web-Präsenz, treten spezielle Anforderungen aus dem WWW-Umfeld vorwiegend in Form der Nachfrage von Algorithmen und Methoden, die sich der gerade beim Online-Publizieren auftretenden Inkonsistenzen annehmen, hinzu.

Ziel eines WCMS ist es, ein aus vielen kleinen Teildokumenten dynamisch aufgebautes Gesamtdokument, die Web-Präsenz, dauerhaft konsistent zu halten.

Inkonsistenzen von Web-Präsenzen entstehen bei größeren Präsenzen hauptsächlich durch die parallele Arbeit an einer Vielzahl von häufig untereinander nicht unabhängigen Dokumenten. Durchlaufen solche Dokumente zudem noch einen Workflow, so ziehen beispielsweise die Dokumente verknüpfenden Hyperlinks ein Inkonsistenz-Potential nach sich.

2.3.1.3.1 Dokumentenbestandaktualisierungen

Für die Web-Präsenz werden Workflows benötigt, die die ständige Fluktuation von einzelnen Dokumenten, Dokumentenmengen und Verzeichnissen abdecken, wobei viele Mitarbeiter des Unternehmens, in verschiedenen Rollen, vom Autor, über die Qualitätssicherung bis hin zum Webmaster gleichzeitig an dem dynamischen Gesamtdokument „Web-Präsenz“ mitwirken. Da nicht alle Arbeiten gleichzeitig abgeschlossen werden, die Web-Präsenz aber jederzeit im Internet verfügbar sein soll, muss das WCMS dafür Sorge tragen, dass trotz des ständigen

Dokumentenflusses immer eine konsistente Version der Web-Präsenz im Internet abrufbar ist; gegebenenfalls müssen bestimmte Arbeitsschritte zurückgehalten werden, bis andere Dokumente nachgeführt wurden oder andere Voraussetzungen erfüllt sind.

Für formale Konsistenz der Dokumente trägt bereits der Content-Management-Anteil eines WCMS Sorge: die Publikation der Dokumente in ein gemeinsames Layout sorgt für einen äußerlich konsistenten Auftritt. Für die inhaltliche Konsistenz der Dokumente kann ebenfalls der Content-Management-Anteil des Systems genutzt werden, er muss jedoch auf den speziellen mehrstufigen, auf schnelle Publikation von Dokumenten optimierten Workflow einer Web-Präsenz angepasst werden; dies immer unter Beachtung der hohen Dynamik des im WWW sichtbaren Dokumentenbestandes auf der einen und der in der Vorbereitung der Publikation befindlichen Dokumente der internen Datenbasis auf der anderen Seite.

2.3.1.3.2 Hyperlink-Management

Eine weitere spezifische Web-Anforderung ist das aus dem im Fluss befindlichen Dokumentenbestand resultierende Hyperlink-Management:

- Dokumente werden referenziert, die in der referenzierten Form nicht mehr veröffentlicht sind.
- Dokumente sollen nicht mehr länger im WWW sichtbar bleiben, obwohl noch eine Reihe anderer Dokumente auf sie verweist.
- Einstiegsseiten für bestimmte Bereiche sollen ersetzt werden, die alten Dokumente aber weiterhin verfügbar bleiben (Alle Referenzen bleiben erhalten, Referenzen auf die Einstiegsseite verweisen auf die neue Einstiegsseite, Referenzen auf das ehemalig als Einstiegsseite verwendete Dokument verweisen weiterhin auf dieses Dokument.).

Bereits anhand dieses Auszugs aus der Liste von Aufgaben, die zum Hyperlink-Management einer Web-Präsenz zählen, lässt sich ein Eindruck der speziellen Web-Anforderungen an ein WCMS ableiten.

2.3.1.3.3 Import/Konvertierung

Abschließend sei zu den Web-Anforderungen noch angemerkt, dass beispielsweise die Übernahme eines bestehenden Dokumentenbestandes aus einer bislang mit anderen Werkzeugen erstellten und gepflegten Web-Präsenz eine häufig im Bereich des Web-Präsenz-Managements anzutreffende Anforderung darstellt, aber unabhängig davon ein entsprechend hoher Bedarf an Importfunktionen bzw. Konvertierungsmechanismen existiert, die Dokumente auf die speziellen Formate (vgl. Seite 42) des WWW abbilden.

2.3.2 Zeit-Schiene – Marktanforderung und Marktanalyse im Vorfeld

Für ein Unternehmen, das sich im WWW-Umfeld platzieren möchte, spielt die zeitliche Komponente eine große Rolle. Auf der einen Seite ist es aus Marketing-Sicht interessant, einer der Ersten zu sein, der ein bestimmtes Produkt, in diesem Fall eine eigene Web-Präsenz, anbietet. Auf der anderen Seite ist es in einem derart gelagerten Fall jedoch nicht möglich, auf die Erfahrungen anderer zurückzugreifen. In den meisten Fällen wird sich daher ein Mittelweg bewähren. In der vorliegenden Arbeit wurde ein solcher Mittelweg beschritten. Einerseits hatten sich die Vorteile einer eigenen Web-Präsenz herauskristallisiert, andererseits war klar geworden, dass der Betrieb einer Web-Präsenz durch eine externe Agentur sich nur bedingt bewährt.

Aus dieser Sicht heraus ist die eigene Web-Präsenz, gepflegt vom Unternehmen selbst, die angestrebte Variante. Sie hat die Kreation eines technischen Umfelds im Unternehmen zur Folge, dessen wichtigster Bestandteil ein System, mit dem die angestrebte Web-Präsenz erstellt, gepflegt und publiziert werden kann, ist.

Über dessen gegebenenfalls sukzessive Integration, wie dies beispielsweise im konkreten Projekt mit der Online-Zeitung realisiert wurde, wird ein Technologie-Transfer in das Unternehmen in die Wege geleitet.

Gespräche mit Kunden zeigen, dass dort Bedarf insbesondere an Systemen besteht, die für ihre speziellen Belange angepasst und ggf. ihren Anforderungen entsprechend modifiziert werden können. Neben speziellen Funktionen ist vorwiegend die Integrationsfähigkeit eines neuen Systems in die bestehende IV-Landschaft, i.e. dies betrifft primär seine Plattformunabhängigkeit, ein wichtiges Argument. Um die bestehende Infrastruktur, die gemäß der Unternehmensstrategie betriebenen IV-Komponenten, nicht zusätzlich durch neue Systeme, die neues Wissen für ihre Bedienung erfordern, zu erweitern, werden große Anstrengungen unternommen, die bestehenden Komponenten in neu angeschaffte und weiterentwickelte Systeme zu integrieren und somit sicherzustellen, dass zumindest auf das bereits vorhandene Basiswissen für diese Komponenten zurückgegriffen werden kann. Gerade die Integration von für das Web-Präsenz-Management so zentralen Komponenten wie Datenbank-Management-Systemen oder Web-Servern ist dementsprechend eine relevante Forderung bei projektgebundener Entwicklung, die zu Restriktionen in der Auswahl und bei der Entwicklung eines entsprechenden Systems führt.

Die größte Flexibilität und den höchsten Anpassungsgrad an spezielle Bedürfnisse liefern Eigenentwicklungen, da sie von Grund auf bezüglich der Vorgaben optimiert werden können und allgemein „schlanker“ als fertige Systeme sind. Bevor jedoch die Eigenentwicklung eines WCMS, wie dies im Rahmen dieser Arbeit geschehen ist, in die Wege geleitet wird, ist im Vorfeld eine Analyse der auf dem Markt in dem angestrebten Umfeld bereits vorhandenen Systeme empfehlenswert.

Im Fall des Projektumfeldes des Instituts für Telematik wurden im Rahmen einer solchen Analyse zum Zeitpunkt der Entscheidungsfindung keine Systeme sichtbar, die den konkreten Anforderungen genügt haben bzw. sich durch Anpassungsarbeiten hätten entsprechend modifizieren lassen. Bei den verfügbaren Produkten handelte es sich entweder um Editor-Programme mit einer Down- und Upload-Funktionalität für einzelne Dokumente oder um skript-basierte Lösungen, die die Verzeichnisse des Web-Servers visualisierten und den Up- und Download von Dokumenten bequemer machten.

Systeme, deren Workflows speziell auf das Management einer Web-Präsenz durch eine Mehrbenutzerzahl ausgelegt waren, die verteiltes Arbeiten auf verschiedenen Plattformen ermöglichten und zudem die Basis für spezielle Individualisierungen und eigene Weiterentwicklungen hätten darstellen können, sind im Rahmen der im Vorfeld betriebenen Analyse nicht sichtbar geworden.

Zu vernachlässigen ist die zeitliche Entwicklung jedoch nicht. Von der Idee, ein Redaktionssystem für das Publizieren im WWW exemplarisch für eine Online-Zeitung zu entwickeln bis zu dem heutigen Entwicklungsstand des Systems für das Content-Management einer Web-Präsenz ist in den Zeiteinheiten der IV-Technologie viel Zeit vergangen. Neben den gestiegenen Anforderungen an das System, welche sich aus der fortgeschrittenen Technik ergeben haben, sind vor allem die direkten Anforderungen der Projektpartner in Form von Anpassungen und Erweiterungen in das System eingegangen.

Während zu Beginn der Entwicklung nur wenige, noch nicht dem Bedarf der Anwender genügende Systeme alternativ zur Verfügung standen und deshalb eine Eigenentwicklung sinnvoll war, ist heute eine breite Palette von Systemen verfügbar, die durch Konfiguration

und kleine Erweiterungen den Unternehmensanforderungen entsprechend gestaltet werden können (vgl. dazu Seite 187).

2.3.3 Strategische Überlegungen

Ein weiteres Argument, das die Entwicklung eines eigenen Systems am Institut für Telematik motiviert hat, ist strategischer Art: Da die eigene Web-Präsenz für ein Unternehmen in den meisten Fällen gleichzeitig auch den Einstieg in die Internet-Technologie bedeutet, sprechen strategische Überlegungen dafür, ein eigenes WCMS zu entwickeln und als Einstiegskomponente für einen Internetauftritt anzubieten.

Nur wenn der Source-Code eines solchen Systems verfügbar ist, kann seine strategische Bedeutung voll ausgeschöpft werden, denn nur dann sind ganz spezielle Anpassungen an die Bedürfnisse der Kunden möglich. Bereits vorhandene Datenquellen können über den Programm-Code einfacher integriert werden als dies durch die Konfiguration eines Fremdsystems machbar wäre.

Die benötigten Arbeitsabläufe können speziell abgestimmt und über den Programmcode optimiert werden. Schließlich können die mit einem Internet-Auftritt verbundenen Dienstleistungen (z.B. Provider, Sicherheitsmanagement) als Erweiterungen des Einstiegspakets betrachtet werden und eine dauerhafte Beziehung zu einem Unternehmen begründen, die sich wiederum in anforderungsgetriebenen Verbesserungen des WCMS niederschlägt.

2.3.4 Evaluation der Technologie

Vielversprechend für eine hohe Integrationsfähigkeit ist der Einsatz von Internet-Technik als Integrationsmedium und die Verwendung offener Standards als Basis für unternehmensweite Interoperabilität. Beides steigert die Flexibilität und Erweiterbarkeit eines Systems und hilft, maximale Zukunftssicherheit zu erzielen. So in etwa kann die theoretische Leistungsfähigkeit der offenen Internet-Technologie beschrieben werden. In der Praxis stellt sich jedoch ihrem Einsatz eine Reihe von Randbedingungen entgegen, die es im Rahmen erfolgreicher Projektarbeit zu überwinden gilt. Abgesehen von technischen Einschränkungen, die den Einsatz bestimmter einzelner Programmkomponenten verbieten oder ganz im Gegenteil erzwingen, stehen hier auch konzeptionelle Überlegungen auf dem Prüfstand. Zu diesen zählen das „Modell der Web-Präsenz-Verwaltung“, die angedachten „Web-Workflows“ und nicht zuletzt als integraler Bestandteil des Gesamtsystems das Hyperlink-Management.

2.3.5 Entwicklungsplattform

Weiterhin stellt ein eigenes WCMS eine Entwicklungsplattform für weitergehende Ziele dar. Auf der Basis des eigenen Systems, für das die Struktur und das Datenmodell bekannt und wichtiger noch modifizierbar sind, können problemlos neue Techniken ausprobiert werden. Sofern ihre Evaluierung erfolgreich verläuft, lassen sie sich direkt in das System integrieren. Schließlich und endlich stellt ein WCMS durch seine vielfältigen Schnittstellen zum Menschen und den Datenquellen auf diversen Servern ein komplexes System dar. Insbesondere die Erkenntnisse aus diesem Bereich liefern einen wichtigen Beitrag dieser Arbeit.

Kapitel 3

Technische Grundlagen im Internet-Umfeld

Die Entwicklung eines Online-Redaktionssystems für den Aufbau und Betrieb einer Web-Präsenz setzt auf einer Reihe von Basistechnologien auf. Die Beschreibung der dahinterliegenden Konzepte und Technologien steht im Mittelpunkt dieses Kapitels. Aufgehängt an dem Begriff „Plattformunabhängigkeit“, der sich auf das für den Einsatz der Systeme notwendige Umfeld bezieht, werden in einem ersten Abschnitt neben Dokumentenformaten Protokolle und Systemarchitekturen erläutert. Die durch den Gebrauch der offenen Internet-Technologie entstehenden Risiken werden mit entsprechenden Lösungsansätzen unter dem Punkt „Sicherheitsmechanismen“ in Abschnitt 3.1.6 abgehandelt. Der zweite Abschnitt dieses Kapitels beschäftigt sich mit der technischen Seite von Web-Präsenzen. Den Grundlagen zugeordnet ist eine Betrachtung der Bausteine, aus denen sich eine Web-Präsenz zusammensetzt. Nach einem Überblick über den Lebenszyklus einer Web-Präsenz werden einige wichtige Kriterien, die eine „gute“ Web-Präsenz ausmachen, diskutiert. Sie liefern einen Eindruck, welche vielfältigen, sich zum Teil auch widersprechenden Anforderungen eine moderne Web-Präsenz erfüllen muss. Von technischer Seite betrachtet, kann sie über zwei verschiedene Mechanismen publiziert werden: entweder sie wird statisch generiert oder einzelne Dokumente werden sozusagen „just in time“ auf Anfrage eines Benutzers generiert. An die Diskussion der Vor- und Nachteile beider Vorgehensweisen schließt sich die Betrachtung von Performance-Fragen und Sicherheitsaspekten an, die beim laufenden Betrieb einer Web-Präsenz relevant werden.

3.1 Plattformunabhängigkeit/offene Standards

In den nachfolgenden Betrachtungen wird unter einer Plattform die Kombination aus einer Rechnerarchitektur und einem Betriebssystem verstanden. Eine Anwendung wird als plattformabhängig bezeichnet, wenn sie nur auf einer bestimmten Hardware unter einer Betriebssystemgattung arbeitsfähig ist. Die Entwicklung einer Anwendung für eine konkrete Zielplattform hat Vor- und Nachteile: Findet eine Entwicklung für eine Zielplattform statt, so können bereits bei der Konzeption plattformspezifische Besonderheiten berücksichtigt werden. Weiterhin lassen sich auf dieser Plattform bereits nativ vorhandene Funktionalitäten einsetzen und so unter Umständen ein Teil des Entwicklungsaufwandes reduzieren. Implementieren Entwickler viele Anwendungen für eine spezielle Plattform und bauen sie des weiteren einen großen Erfahrungsschatz im Umgang mit dieser Plattform auf, sind sie in der Folge in der Lage, durch Anwendung von Detailkenntnissen und Ausnutzung von Systemspezifika u.U. deutliche Performance-Vorteile zu realisieren. Dementsprechend wird auch heute noch

in denjenigen Bereichen sehr plattformnah entwickelt, die auf eine starke Optimierung der Anwendungen ausgerichtet sind. Solange Dokumente nur zwischen Systemen einer Plattform ausgetauscht werden, sind keine Konvertierungen der Daten notwendig.

Die Vorteile einer plattformspezifischen Implementation liegen bei einer homogenen IV-Landschaft auf der Hand. Anders stellt sich die Lage in einer heterogenen IV-Landschaft dar. Hier sind Anwendungen gefordert, die möglichst portabel sind, d.h. unabhängig von einer konkreten Hardware und einem bestimmten Betriebssystem eingesetzt werden können. Die Vorteile plattformunabhängiger Anwendungen liegen in ihrer Einsatzmöglichkeit auf einer Vielzahl verschiedener Plattformen, ohne dass grundlegende Modifikationen notwendig werden. Aus der Sicht der Software-Entwicklung lohnt sich der Verzicht auf Plattformabhängigkeit, da sich auf diese Weise ein viel größeres Anwendungspotential ergibt. Für ein Unternehmen ist plattformunabhängige Software eine sinnvolle Lösung, da sie bei einem Wechsel der Plattform weiterhin einsetzbar bleibt. Sie trägt somit zur Investitionssicherheit bei der Anschaffung einer solchen Anwendung bei. Die Nachteile plattformunabhängiger Anwendungen liegen in der Einigung auf den kleinsten gemeinsamen Nenner aller angestrebten Plattformen. Sobald eine Anwendung Eigenschaften einer Plattform ausnutzt, die auf einer anderen Plattform nicht zur Verfügung stehen, wird die Plattformunabhängigkeit eingeschränkt. Durch den Verzicht auf nicht überall zur Verfügung stehende Funktionalitäten verschwindet die Möglichkeit zur Optimierung von Software für eine Plattform. Im Normalfall wird die Plattformunabhängigkeit durch der Performance wenig zuträgliche Zwischenschichten wie z.B. der virtuellen Laufzeitumgebung (Virtual Mashine) in der Java Byte-Code ausgeführt wird (siehe Seite 50), erkauft.

In der Realität ist Plattformunabhängigkeit nur stark eingeschränkt realisierbar. Von Plattformunabhängigkeit wird oft bereits gesprochen, wenn die Software auf mehr als einer Plattform eingesetzt werden kann. Im Internet-Umfeld wird ein System als plattformunabhängig betrachtet, wenn auf der Client-Seite lediglich ein Web-Browser (siehe Seite 38) benötigt wird. Diese Betrachtung ist insofern wichtig, als dass bei einer mit Internet-Technik arbeitenden Anwendung im Umfeld des WWW mit einer Vielzahl von verschiedenen Konfigurationen von Hardware und Betriebssystem auf der Client-Seite gerechnet werden muss. Da der Betreiber einer Webapplikation (siehe Seite 174), die von Benutzern aus unterschiedlichem Umfeld genutzt werden soll, nicht für jede denkbare Plattform eine eigenständige Anwendung entwickeln kann, ist hier die Beschränkung auf das Vorhandensein und die Einsetzbarkeit eines Web-Browsers als gemeinsame Grundlage bereits ein großer Schritt. Sie erlaubt es, eine Anwendung für den Ablauf mit bzw. in einem Web-Browser auszulegen. Voraussetzung für diesen Schritt ist das Vorhandensein eines eigenen Web-Browsers für jede Plattform.

3.1.1 Web-Browser & Web-Server

Der Begriff Web-Browser wird im Zusammenhang mit dem WWW als Synonym für HyperText-Client verwendet. Das Wort (Web-) Browser kommt vom englischen Wort (to) browse = blättern, durchstöbern, schmökern. Web-Browser stellen mit ihrer Fähigkeit zur Darstellung von Webseiten und zur Navigation im WWW auf der Client-Seite die Schnittstelle zum Internet und Intranet dar. Moderne Web-Browser sind multimediafähig – sie können Text-, Bild-, Ton- und sogar Video-Dokumente am Bildschirm visualisieren.

1992 war am CERN das erste zeilenorientierte Programm im Internet verfügbar. Auch heute werden im Internet noch zeilenorientierte Browser verwendet. Das Programm *Lynx* kommt beispielsweise noch bei Telnet-Sitzungen, d.h. wenn „remote“ auf einem Rechner gearbeitet wird, zum Einsatz. Die zeilenorientierten WWW-Browser wurden mittlerweile von den grafikfähigen Browsern praktisch vollständig vom Markt verdrängt. Neue und komfortablere WWW-Browser mit einer grafischen Benutzeroberfläche wurden ab 1993 entwickelt. Damit

begann auch der weltweite Boom des neuen Internetdienstes. Als erste grafische WWW-Browser waren *NCSA-Mosaic* und *Cello* im Internet vertreten. Der später aus *Mosaic* erwachsene *Netscape Navigator* hatte bis 1995 einen Marktanteil von rund 90%. Seit 1996 von *Microsoft* die auch auf *Mosaic* beruhende Implementation des *Internet Explorers* in der ersten Version auf den Markt kam, hat sich der Marktanteil von *Netscape* deutlich reduziert. Mit jeder neuen Version konnte der *Internet Explorer* gegenüber der entsprechenden Antwort von *Netscape* mit dem *Navigator* aufholen. Mittlerweile hat sich das Verhältnis sogar deutlich zugunsten von Microsoft gewendet. Ein großer Vorteil von *Netscape's Navigator* stellt die nach wie vor größere Plattformunabhängigkeit dieses Produktes dar. Im Wettstreit mit *Microsoft* wurde diese zugunsten von Microsoft- Windows-lastigen Implementationen vernachlässigt. Trotz dieser Probleme steht heute auf jeder gängigen Plattform ein hochfunktionaler Web-Browser zur Verfügung. Die beiden meistgenutzten Browser unterscheiden sich in der Umsetzung diverser Standards zum Teil noch recht deutlich voneinander – ein Punkt, der gerade bei der Darstellung komplexer Dokumente deutlich wird. Ergänzt werden die beiden marktbeherrschenden Web-Browser durch eine Vielzahl anderer Web-Browser, die eine Existenz als Nischenprodukte führen. Größte Popularität unter ihnen genießt der norwegische Browser *Opera*.

Ein Konzept, mit dem Web-Browsern Funktionalitäten zugeführt werden können, über die diese nicht nativ bzw. in unterschiedlicher Implementierung verfügen, sind die Plug-ins [Ang96]. Als Plug-ins werden Anwendungen bezeichnet, die in den Web-Browser eingeklinkt werden. Der Web-Browser kümmert sich um die Kommunikation mit dem Web-Server und lädt die ausführbare Anwendung oder das darzustellende Dokument. Die eingeklinkte Anwendung, das Plug-in, gewährleistet die Ausführung bzw. die Darstellung. Über für viele Plattformen nativ oder plattformunabhängig verfügbare Plug-ins wird die Nutzung der einheitlichen Datentransferschnittstelle des Web-Browsers mit der speziellen Funktionalität des Plug-ins kombiniert.

Das Gegenstück zum Web-Browser stellt der Web-Server dar. Er ist der technische Kern der Web-Präsenz. Seine Aufgabe ist die Kommunikation mit den Web-Browsern. Auf Anfragen der Web-Browser liefert er unter Berücksichtigung von Zugriffsschutzmechanismen die angeforderten Daten zurück. Bei den zurückgelieferten Daten kann es sich sowohl um aus dem Dateisystem des Web-Server-Rechners, der Einfachheit halber synonym auch als Web-Server bezeichnet, gelesene Dateien als auch um speziell auf diese Anfrage hin generierte Dokumente handeln. Mit dem Wandel von Web-Präsenzen im herkömmlichen Sinne zu Applikationen mit Web-Schnittstelle, zu sogenannten Webapplikationen, steigt die von den Web-Servern implementierte Funktionalität zur Einbindung und Ausführung von Programmelementen.

Moderne Web-Server haben sich von reinen Dateiauslieferungs-Maschinen zu multifunktionalen Servern weiterentwickelt, die einen großen Anteil an der Integration der verschiedenen Applikationen in einem Unternehmen besitzen.

Das noch aus den Anfängen des WWW stammende Common Gateway Interface (CGI) [Gun00] findet dabei auch heutzutage noch starke Anwendung. Im Trend liegen nach wie vor skriptbasierte Programmabläufe, entwickelt in Skriptsprachen wie PERL [WS92] oder PHP [HZ00]. Sie werden aber zunehmend von vollständigen Applikationen verdrängt. In letzterem Fall kommen statt eines einfachen Web-Servers sogenannte Application-Server (siehe Seite 180) zum Einsatz. Das sind Programmpakete, bei denen eine Laufzeitumgebung für komponentenorientierte Programmbausteine bereitgestellt wird, die mit einer Web-Schnittstelle versehen sind. Zwischenstufen zwischen Web-Server und Application-Server werden durch eine Erweiterung der Web-Server-Funktionalität durch Einbindung von Zusatzfunktionalitäten in Form von Modulen analog zu den Web-Browser Plug-ins realisiert.

Um dynamische Dokumente, d.h. Dokumente, die erst bei der Anfrage durch den Web-

Browser erzeugt werden, zu generieren, gibt es zwei verschiedene Mechanismen: entweder die Dokumente werden vollständig von Skripten bzw. Programmkomponenten erzeugt oder es wird beim Ausliefern eines Dokumentes darin enthaltener Programmcode ausgeführt. Der erste Fall entspricht den SKripten, die über die CGI angesprochen werden. Im zweiten Fall, bei dem in den Dokumenten Programmcode abgelegt bzw. eingebunden wird, sind gängige Mechanismen die Server-Side Includes [LL99] bzw. Active Server Pages [BUD+00] oder Java Server Pages [Hal00]. Hier wird der in den Dokumenten enthaltene bzw. referenzierte ausführbare Programmcode vor der Auslieferung an den Web-Browser vom Web-Server ausgeführt. Das Resultat des Programmdurchlaufs wird in das Dokument anstelle des Programmcodes eingebunden und das so vollständige Dokument an den Web-Browser zur Darstellung zurückgeliefert.

Bekannte aktuelle Web-Server sind der unter der GNU Public Licence erhältliche *Apache* (1.3.x)[LL99], der kommerzielle *Netscape Enterprise-Server* (*I-Planet*) sowie der *Internet Information Server* von *Microsoft*.

Auf der Server-Seite ist Plattformunabhängigkeit zwar ebenfalls wünschenswert, aber nicht so zwingend, da die Zahl der hier betroffenen Rechner im Verhältnis zu der Zahl der Clients relativ klein ist. Unterschieden werden muss auf der Server-Seite zwischen dem Server an sich und der von dem Server genutzten Skripte sowie Programmkomponenten. Der Web-Server wird als Applikation gegebenenfalls zu Lasten der Plattformunabhängigkeit auf höchste Performance optimiert, um eine möglichst große Zahl von Anfragen simultan bewältigen zu können. Für die Skripte und Programmkomponenten ist die Performance ebenfalls ein bedeutendes Ziel, aus Gründen der Investitionssicherheit kann es sich trotzdem rentieren, auf plattformunabhängige Lösungen zu setzen. Ist die Performance eines Systems nicht mehr gegeben und zieht deshalb die Webapplikation auf eine neue Plattform mit größerer Skalierbarkeit um, zahlt es sich aus, wenn sie nicht komplett neu entwickelt werden muss, sondern direkt auf der neuen Plattform produktiv werden kann.

3.1.2 Protokolle

Zum Erreichen von plattformunabhängiger Kommunikation ist der Einsatz allgemein akzeptierter Protokolle, Standards und Techniken notwendig. Der große Erfolg des Internets fußt auf allgemein akzeptierten Protokollen zur plattformübergreifenden Kommunikation miteinander vernetzter Rechner. Basierend auf vergleichsweise einfachen Protokollen – dem Internet-Protokoll (IP) bzw. dem darauf aufsetzenden Transmission-Control-Protocol (TCP/IP) – werden komplexe verteilte Anwendungen möglich.

Das IP ist Teil eines anerkannten Industriestandards für die Kommunikation zwischen offenen Systemen. Paket- statt leitungsvermittelnd arbeitend, besteht die Hauptaufgabe des IP in der netzübergreifenden Adressierung. Definierte Datenfragmente, als Datagramme bezeichnet, werden über die Infrastruktur des Internets zu den gewünschten Zieladressen dynamisch, d.h. ohne dass ein konkreter Weg von vornherein festgelegt wird, geleitet. Entsprechend hohes Gewicht kommt bei der Kommunikation im Internet der IP-Adresse zu. Eine klassische IP-Adresse besteht aus vier Bytes (IPv4), die durch Punkte getrennt sind, zum Beispiel 153.96.230.24 . Über die IP-Nummern werden programmintern Netzwerk-Ressourcen, wie z.B. Computer, Web-Server, Drucker oder Web-Cams identifiziert. Da IP-Nummern für Menschen nicht sprechend sind, werden sie über einen speziellen Dienst, den Domain-Name-Service (DNS), in eindeutige, klartextliche Namen übersetzt. Dadurch kann beispielsweise in Web-Browsern die IP-Adresse 153.96.230.24 parallel zu dem klartextlichen Namen „www.ti.fhg.de“ verwendet werden. Der DNS stellt einen wichtigen Eckpfeiler des heute bekannten WWW dar. Ohne DNS wären bei weitem nicht so breite Benutzerschichten in der Lage, gezielt im WWW

Ressourcen aufzusuchen.

Auf dem IP setzt das TCP/IP (Transmission Control Protocol/Internet Protocol) [Hun98] als Basisprotokoll des Internets auf. Das IP dient wie beschrieben der Adressierung von Datenpaketen, sorgt jedoch nicht für eine gesicherte Übertragung. Zum einen kann es nicht verhindern, dass Datenpakete verloren gehen, zum anderen ist es aufgrund unterschiedlicher Routen, die Datenpakete über das Internet nehmen können, möglich, dass Laufzeitunterschiede zu einer geänderten Reihenfolge der Datenpakete führen. Das TCP sorgt beim Empfänger für die Einsortierung der Pakete in der richtigen Reihenfolge. Durch Bestätigung des Paket-Empfangs und der Korrektur von Übertragungsfehlern stellt es die erfolgreiche Kommunikation sicher.

Auf dem gemeinsamen Basisprotokoll TCP/IP bauen die weiteren im Internet bekannten Protokolle [Wil99] für die spezifischen Aufgaben auf: z.B. HTTP im WWW oder FTP für den Dateientransfer (Filetransfer) im Internet, oder POP3 [MR94], IMAP [Cri96] und SMTP [Pos82] für den Austausch von E-Mails.

Erst durch die Einführung von Hyperlinks wurde die dem WWW seinen Namen gebende Netzstruktur möglich. Dies beruht auf der Eigenschaft von Hyperlinks, auf Dienste und Dokumente verweisen zu können, die „irgendwo“ im Internet über einen Web-Server angeboten werden. Für diese internetweite Verknüpfung von Dokumenten werden eindeutige Adressen für jedes einzelne Dokument benötigt. Um die Adressierung einzelner Rechner, Dienste und Dokumente im Internet einfach und nachvollziehbar zu gestalten, wurde das Konzept der URIs (Uniform Resource Identifier) [BLFM98, Wil99] entworfen, welches sich als Standard durchgesetzt hat.

Eine URI setzt sich in festgelegter Reihenfolge aus diversen Komponenten zusammen. Diese sind erstens das zu verwendende Dienst-Protokoll, zweitens die Adresse des Rechners (Rechnername und Domäne), drittens der Port und schließlich die Spezifizierung des angeforderten Datensatzes, z.B. der Dokumentname, gegebenenfalls versehen mit einem Pfad im Datei-System des Web-Servers. Die URI eines Dokumentes der Web-Präsenz des Instituts für Telematik lautet beispielsweise: `http://www.ti.fhg.de/Profil und Kontakt/Willkommen.html`. Dabei wurde analog zu dem oben beschriebenen DNS-Konzept die IP-Adresse des Web-Servers durch den klartextlichen Namen ersetzt. Weiterhin verfügen die meisten Dienste über Standard-Ports. SMTP kommuniziert z.B. auf Port 25 und HTTP im Normalfall auf Port 80. Sofern der Standard-Port verwendet wird, kann auf die explizite Nennung des Ports in der URI verzichtet werden. Die vollständige URI des obigen Beispiels wäre somit `http://www.ti.fhg.de:80/Profil und Kontakt/Willkommen.html`. Die in der URI verwendeten Trennzeichen, die Kombination „://“ und der Doppelpunkt vor dem Port sind zwingend vorgeschrieben. Zentrales Protokoll im Rahmen dieser Arbeit ist das HTTP, eine Abkürzung für „Hypertext Transfer Protocol“. Es stellt das alles beherrschende Kommunikationsprotokoll zwischen Web-Server und Web-Browser zur Übertragung von Daten dar. Momentan sind zwei Versionen von HTTP im Internet zu finden. Zum einen die mittlerweile fast abgelöste Version HTTP 1.0 definiert in RFC 1945 [BLFF96] und die aktuelle Nachfolgeversion HTTP 1.1 beschrieben in RFC 2068 [GMM+99].

Beim HTTP werden, basierend auf dem Client-Server-Prinzip (siehe Seite 48) Daten abgerufen und bereitgestellt. Die Aufgabe der Datenlieferanten übernehmen dabei die Web-Server. Bei denjenigen Client-Programmen, die Daten abfragen, handelt es sich im Allgemeinen um Web-Browser, aber auch beliebige andere Anwendungen (z.B. Robots, Spider etc.) können mittels HTTP mit einem Web-Server kommunizieren. Die Kommunikation mittels HTTP erfolgt klartextlich. Es ist ohne weiteres möglich, in einer Telnet-Sitzung mit einem Web-Server über die Eingabe von HTTP-Befehlen zu kommunizieren. Die klartextliche Formulierung von

Protokollbefehlen ist eine Eigenschaft der auf dem TCP/IP implementierten Standardprotokolle, die stark zu ihrer Verbreitung beigetragen haben und die die Kommunikation zwischen verschiedenen Plattformen erleichtern.

Ein weiteres Merkmal von HTTP sind die zustandslosen Verbindungen, HTTP baut keine Sitzungen auf. Jede Anfrage wird als ein separater Vorgang behandelt, unabhängig davon, ob es bereits zuvor Anfragen von dem gleichen Client gegeben hat. Damit wird die Verwaltung der Sitzungsinformation auf die Webapplikation selbst übertragen. Konzepte, die hier eingesetzt werden, sind das Plaztieren von Cookies [Wil99] (temporär oder persistent) bei dem Web-Browser und das Umschreiben der URI.

3.1.3 Dokumentformate im WWW – Hypertext

Das WWW kann als ein großes Informationssystem (siehe Seite 173) betrachtet werden, es besteht aus einer Vielzahl von Dokumenten. Diese sind entsprechend ihrem Hypertext-Charakter miteinander durch Hyperlinks verbunden [Fre94]. Der Brockhaus [Ges00] definiert den Begriff Hypertext kurz als „nicht lineare Strukturierungs- und Präsentationsform von Textinformationen, die sich dem Nutzer mittels gezielter Verweise (Hyperlinks) auf andere Dokumente erschließen. Sind auch multimediale Daten (z. B. Ton, Bild) eingebunden, spricht man von Hypermedia.“

Die offenen Datenformate, die im WWW zum Einsatz kamen und kommen, sind zu einem nicht unerheblichen Teil an dem immensen Erfolg des WWW beteiligt. Die Offenheit der Dokumentenformate (siehe unten) ermöglichte den Inhalten die Überwindung der Plattformabhängigkeit. Der Schritt von nativen, plattformabhängigen zu plattformübergreifenden Formaten erlaubte die WWW-weite Distribution von Dokumenten. Dass Verteilung auch wirklich stattfindet, hängt mit dem Hypertext-Charakter des WWW zusammen. Allein durch das Netz von Hyperlinks zwischen den einzelnen Dokumenten wird der Vorgang des „Browsens“ möglich.

Ohne die Einbindung von multimedialen Daten wäre das WWW nie zu dem bunten, lebendigen und TV-ähnlichen Medium geworden, als das wir es heute wahrnehmen. Neben Grafiken werden Audio-Dateien und Filme in einer Web-Präsenz platziert. In der Handhabung sind multimediale Elemente deutlich anspruchsvoller als textuelle Bausteine und beanspruchen deutlich mehr Speicherplatz, weshalb gerade bei ihnen Komprimierung essentiell ist.

Abgesehen vom Umfang der betreffenden Dateien sind multimediale Daten auch für die inhaltlich orientierte, automatisierte Verarbeitung, z.B. durch ein (W)CMS, denkbar ungeeignet. Die in den multimedialen Dokumenten enthaltenen Informationen sind so vielfältig, dass ihre maschinelle Erkennung entweder nur in sehr speziellen Umfeldern oder entsprechend unzureichend funktioniert. Im Rahmen des WCM wird versucht, multimedialen Dateien mit Metadaten beizukommen. Mit der Hilfe von textuellen Informationen, die den Dateien zugeordnet werden, wird eine Möglichkeit geschaffen, beispielsweise eine Suche über einen Bilderbestand durchführen zu können.

3.1.3.1 HTML

Neben dem vergleichsweise simplen Datentranferprotokoll (HTTP) hat maßgeblich die einfache Dokumentbeschreibungsprache HyperTextMarkupLanguage [CW00], die von Goldfarb entwickelt wurde und in der ISO-Norm 8879 definiert ist, an der explosiven Ausbreitung des WWW Anteil maßgeblichen Anteil. Wie auch die Extended Markup Language (XML) [BPSMM98] beruht HTML auf dem Konzept der Standard General Markup Language (SGML) [Gol90]. Während XML eine echte Teilmenge von SGML darstellt, handelt es sich

bei HTML um einen stark simplifizierten Auszug von SGML-Elementen. Mit HTML, mittlerweile in mehreren Versionen (HTML 2.0, 3.2, 4.0, XHTML etc.) vorliegendend, können einfache Formatierungen und Layouts von Dokumenten vorgegeben werden. Das eigentliche Layout der Dokumente für die Darstellung am Rechner des Benutzers bleibt aber der Layout-Engine des Web-Browsers überlassen, welcher die Formatierungsinformationen interpretiert und die Dokumente in Abhängigkeit von seiner Implementierung darstellt.

3.1.3.1.1 Tags

Im HTML-Text markieren Steuerungsmarkierung oder Steuerungsanweisung (engl. Tags) Abschnitte oder Wörter und einzelne Buchstaben mit besonderer Formatierung. Damit der Browser die Steuerzeichen erkennt, beginnen und enden sie mit einer spitzen Klammer (`< >`). Die Tags sind sozusagen einzelne HTML-Befehle, die den Web-Browser anweisen, den so markierten Text auf eine bestimmte Art darzustellen. Fast alle Befehle von HTML bestehen aus einem einleitenden und einem abschließenden Tag. Der Text dazwischen ist der „Gültigkeitsbereich“ für die betreffenden Tags. Die Tags `<b>` und `</b>` (engl. bold) umschließen beispielsweise **fett** darzustellenden Text. Jedes HTML-Dokument beginnt grundsätzlich mit dem Tag `<html>` und endet mit dem Tag `</html>`. Der manuelle Umgang mit Tags entfällt, wenn ein HTML-Editor verwendet wird, der die HTML-Eigenarten beherrscht.

Ein HTML-Dokument ist in zwei Bereiche aufgeteilt: den Kopf (Head) und den Textkörper (Body). Im Head werden neben dem Titel des Dokuments, der von dem Web-Browser angezeigt wird, Metadaten (siehe Abschnitt 5.2.4) abgelegt. Metadaten sind Informationen über Daten und Objekte, die deren Beschreibung dienen. Neben der Einbettung in den Header der HTML-Datei, wie hier beschrieben, können sie in HTML-Dateien, die nur aus Metadaten bestehen und die auf Nicht-HTML-Dateien (Bild-, Tondateien etc.) verweisen sowie in Datenbankstrukturen, abgelegt werden [Dem97].

Ein auszugsweises Beispiel für die Tags, die im Head eines HTML-Dokumentes stehen können, wäre:

```
<HEAD>
 <TITLE>Erklärung HTML-Tags</TITLE>
  <META NAME="package" TYPE="begin" CONTENT="Dublin Core">
  <META NAME="DC.title" CONTENT="Erklärung HTML-Tags">
  <META NAME="DC.creator" TYPE="name" CONTENT="Andreas Heuer">
  <META NAME="DC.creator" TYPE="email" CONTENT="heuer@ti.fhg.de">
<META NAME="DC.subject" TYPE="keyword"
CONTENT="Tag, HTML-Befehl, Steuerungsanweisung, Erklärung">
<META NAME="DC.description" CONTENT="Ein kurzer Überblick
 über die Steueranweisungen in HTML-Dokumenten">
...
 </HEAD>
```

An den Kopf eines HTML-Dokumentes schließt sich der Textkörper mit seinem eigentlichen Inhalt an. Er enthält neben Text, auch eingebettet durch spezielle Tags, Grafiken, Audio- oder Video-Elemente sowie Animationen und Java-Applets. Der Body eines Dokumentes hat in etwa die Form:

```
<BODY>
<H1>Hier eine Überschrift</H1>
<P>Ein Paragraph mit Text zu der Überschrift</P>
Eine Grafik zum Text <IMG SRC="Name der Datei">.
</BODY>
```

3.1.3.1.2 Hyperlinks

Eines der wichtigsten Elemente von HTML ist die Verknüpfung, der sogenannte Hyperlink, wie er im Aufsatz von Schwartz und Halasz [HS94] beschrieben wird. Ein Hyperlink ist ein Verweis auf ein anderes Dokument. Dies kann sowohl ein multimediales Element sein, welches in das HTML-Dokument zur Darstellung eingebunden wird, als auch einfach ein Navigations- oder Referenzverweis auf ein weiterführendes Dokument. Wichtig bei der Adressierung ist die Unterscheidung zwischen absoluten und relativen Hyperlinks: ein absoluter Hyperlink verweist unabhängig vom Kontext auf ein bestimmtes Dokument im WWW, bei einem relativen Link ist der Verweis nur in Abhängigkeit von dem verweisenden Dokument möglich. In HTML hat ein Hyperlinks, auch als Anker, bezeichnet das Format:

```
<A HREF="URI der Referenz">Text oder Grafik, die die Referenz beschreibt</A>
```

Auch Grafiken oder andere multimediale Elemente lassen sich über einen analogen Tag, den Source-Tag, einbinden:

```
<IMG SRC="URI der Grafik">
```

Einer der großen Nachteile der z.Zt. im Internet gebräuchlichen Hyperlinks ist ihre **Unidirektionalität**. Ein Hyperlink geht üblicherweise von einem HTML-Dokument Q (Quelle) aus und verweist auf ein beliebiges Dokument Z (Ziel) im Internet. Im Normalfall „weiß“ der für Dokument Z Verantwortliche nicht, dass es ein Dokument Q gibt, welches auf das Dokument Z verweist. Die Unidirektionalität hat zur Folge, dass es im WWW sehr viele Hyperlinks gibt, die auf Dokumente verweisen, welche nicht mehr verfügbar sind. Solche Hyperlinks werden als „Broken-Links“ bezeichnet. Mit dem Altern des WWW ist zu erwarten, dass die Zahl dieser Broken-Links zunimmt, da immer mehr veraltete Dokument aus dem WWW entfernt werden.

Während es prinzipiell möglich ist, die Hyperlinks zwischen Dokumenten innerhalb der Web-Präsenz konsistent zu halten, ist es nur durch regelmäßige Überprüfung der Hyperlinks, die auf externe Dokumente zeigen, machbar auch bei diesen Konsistenz zu erzielen. In allen Fällen ist bei der Erhaltung von Verknüpfungskonsistenz zu beachten, dass zwischen formaler und inhaltlicher Konsistenz unterschieden werden muss.

Hyperlinks sind formal konsistent, wenn das Dokument, auf das sie verweisen, verfügbar ist, d.h. der Hyperlink nicht außer Funktion ist. Aber selbst wenn der Hyperlink formal in Ordnung ist, kann es vorkommen, dass sich der Inhalt des Dokumentes, auf das der Hyperlink zeigt, geändert hat und nicht mehr in den Kontext des verweisenden Dokumentes passt. In diesem Fall ist der Hyperlink inhaltlich inkonsistent.

Um Web-Präsenzen zumindest formal konsistent zu halten, gibt es neben der manuellen oder skriptbasierten Lösung des „Testens“ aller Hyperlinks noch weitere, elegantere Möglichkeiten. Beispielsweise können durch das Speichern aller Hyperlink-Informationen in einer Datenbank viele Inkonsistenzen schon vor deren Auftreten vermieden werden. Ist bekannt, welche Hyperlinks auf ein Dokument zeigen, so kann vor dem Editieren oder Löschen durch eine Sicherheitsabfrage in der Form: „Achtung, auf dieses Dokument verweisen die Dokumente X, Y etc.“, auf ein Inkonsistenz-Potential hingewiesen werden. Im Prinzip sind auf diese Weise innerhalb der Web-Präsenz bidirektionale Hyperlinks realisiert. Jedes Dokument „weiß“, welche Dokumente auf es verweisen. Diese Kenntnis zu verwalten und und bei Bedarf zu intervenieren, ist Aufgabe für die Hyperlink-Management-Komponente in einem WCMS.

3.1.3.1.3 Cascading Style Sheets (CSS)

Bei CSS [LB99] handelt es sich um eine HTML-Erweiterung. Diese wird eingesetzt, um HTML-Seiten abstrakter layouten zu können und häufig benutzte Formatvorgaben (wie

bold, kursiv, italic etc.) und Schrifttypen nur noch ein einzige Mal definieren zu müssen. Da die Style-Informationen in separate Dateien ausgelagert werden können, die in HTML-Dokumente eingebunden werden, sind so dokumentübergreifende Designvorgaben möglich. Bei der Visualisierung der Dokumente verwendet der Web-Browser die Einstellungen des Style-Sheets anstelle der Standardeinstellungen. Somit lassen sich global für alle Dokumente oder für spezielle Dokumentmengen Textelemente formatieren. Allein durch die Modifikation des Style-Sheets werden alle Dokumente der Web-Präsenz, die auf das Style-Sheet zurückgreifen, gleichzeitig in ihrem Design modifiziert.

Leider interpretieren Web-Browser verschiedener Hersteller diese Formatvorlagen jeweils unterschiedlich. Deshalb ist bei ihrer Anwendung mit Vorsicht vorzugehen und im Vorfeld eine Reihe von Tests durchzuführen. Nicht vergessen werden darf, dass der Benutzer des Web-Browsers aus Sicherheitsgründen die Anwendung von Style-Sheets verboten haben kann. Ist dies der Fall, so sollte das bei ihm angezeigte Dokument immer noch über eine Formatierung verfügen, die es nicht unbrauchbar macht. Ein Style-Sheet hat in etwa das in dem folgenden kurzen Beispiel vorgestellte Aussehen:

```
<STYLE type="text/css">
<!--\#afett {font-family: Arial; font-style: bold;} --> </STYLE>
```

Die Anwendung dieses Styles in einem HTML-Dokument ist dann beispielsweise über die Zeile :

```
"<P id="afett">fetter Arial-Text.</P>"
```

möglich.

3.1.3.2 XML

Mit dem Ziel, die Extreme zwischen strukturierten Daten (im Backend) und unstrukturierten HTML-Dokumenten (im Web) aufzuheben, wurde XML [BPSMM98] als eine Meta-Sprache im W3-Consortium zur Beschreibung von Dokumenten und Daten aus SGML entwickelt. Sie unterscheidet nicht zwischen Dokumenten und Daten. XML ist gerade in Bezug auf den Datenaustausch im E-Business (B2B) absolut unerlässlich. Die große Chance, die XML für die Informationsverarbeitung eröffnet, liegt in der Universalität seiner Einsatz-Möglichkeiten und dem damit gebotenen Integrationspotential. XML trennt zwischen Content (Inhalt), Struktur und Präsentation und beseitigt damit zahlreiche Probleme von HTML und seinen proprietären Erweiterungen.

Die beiden Sprachen HMTL und XML verfolgen einerseits unterschiedliche Ziele und ergänzen sich andererseits. Im direkten Vergleich zu HMTL ist XML wesentlich flexibler und potenter und grenzt sich von HTML in vier wesentlichen Punkten ab: erstens ist XML eine Enabling-Technologie, die den Basis-Rahmen für Entwicklungen legt, zweitens gibt es anders als bei HTML bei XML keinen vorgegebenen Satz an Formatierungsbefehlen (Tags); neue Tags können jederzeit vom Benutzer definiert werden, drittens können Daten- und Dokumentstrukturen beliebig geschachtelt werden und viertens kann jedes Dokument (bzw. jeder Datensatz) eine eigene Bauvorschrift (Grammatik) enthalten, die beispielsweise für eine Syntaxprüfung genutzt werden kann. XML ist mithin eine Sprache, mit der eine Grammatik für eine Dokumenten-Familie beschrieben werden kann.

Analog zu HTML werden bei XML Text und Daten über Tags markiert. Dabei beschreiben sogenannte Document-Type-Definitions (DTDs) [BPSMM98], welche Tags ein XML-Dokument haben darf und wie sie strukturiert/verschachtelt werden dürfen. Weiterhin liefern die DTDs eine inhaltliche Semantik, die die automatisierte Weiterverarbeitung der in den Dokumenten abgelegten Daten ermöglicht. Im Allgemeinen folgt aus den XML-Tags aber noch

keinerlei Information für den Web-Browser, wie der entsprechende Tag darzustellen ist. Das Layout wird analog zu HTML über Style-Sheets, XSL (Extended Style Language) [ABC+00] geregelt, deren Standardisierung sich derzeit dem finalen Stadium nähert, deren Umsetzung in Web-Browsern jedoch noch weit von der Einführung entfernt ist.

Während bei HTML die Akzeptanz der Browser von beliebigem, auch absolut nicht wohlgeformten HTML-Code, dazu geführt hat, dass HTML-Dokumente eine Mischung aus verschiedenen HTML-Standards sind und häufig auf das Schließen von HTML-Tags verzichtet wird, fordert XML gerade wohlgeformte Dokumente. Dadurch wird die automatische Verarbeitung von XML-Dokumenten stark erleichtert. Die aufwendigen, beliebige HTML-Dokumente bearbeitenden Parser-Mechanismen werden bei XML deutlich einfacher.

Aufgrund der weiten Verbreitung von HTML und der noch geringen Verfügbarkeit von XML-fähigen Web-Browsern steht zu erwarten, dass bei Web-Präsenzen der Wechsel im Dokumentenbestand von HTML nach XML noch einige Zeit auf sich warten lassen wird. Basierend auf XML wird jedoch mit XHTML [otWHWG00] eine neue Generation von HTML geschaffen, die den Übergang zu beliebigen, in XML beschriebenen Dokumenten erleichtert. Da die Konvertierung von XML-Dokumenten in HTML-Dokumente viel einfacher ist als der umgekehrte Weg, werden in zunehmendem Maße Daten strukturiert in XML verpackt und bei Bedarf dann nach HTML konvertiert. Dies ist das übliche Verfahren, das Content-Management-Systeme aufgrund der klaren Trennung von Inhalt und Layout anwenden.

3.1.3.3 Portable Document Format (PDF)

Mit dem Portable Document Format (PDF) [Inc00] von *Adobe* entwickelt sich für die Dokumente einer Web-Präsenz ein Konkurrenzformat zu HTML und XML. Im Gegensatz zu vorgenannten Dokumentformaten ist PDF ein universelles Dateiformat, das alle Schriften, Formatierungen, Farben und Grafiken jedes Ausgangsdokuments beibehält, unabhängig von der Anwendung und der Plattform, die im Erstellungsprozess verwendet wurden. Dem Konzept der Trennung von Inhalten und Layout widerspricht das Konzept von PDF damit vollständig. Aufgrund ihres proprietären Formats sind PDF-Dokumente, im Gegensatz zu den textbasierten Formaten HTML und XML, nur schwerlich für die automatisierte Dokumentverwaltung geeignet. Für PDF-Dokumente werden spezielle Import- und Export-Filter benötigt. Für den Publikationsprozess werden die PDF-Dateien nicht durch Einbindung in Layout und Navigationsstruktur modifiziert. Trotz dieser Nachteile für das Content-Management haben PDF-Dateien im WWW ihre Daseinsberechtigung. Für bestimmte Dokumente ist ein fest definiertes, plattformunabhängiges Darstellungsformat von großer Bedeutung. Solche Dokumente können beispielsweise Formulare und Broschüren sein, die für den Druck beim Benutzer bestimmt sind. Nach Aussagen von *Adobe* stellt PDF den Workflow-Standard der Zukunft im Verlagswesen, das vor der Aufgabe steht, die bislang in Printform vertriebenen Publikationen wie Bücher und Kataloge nun auch online zu vertreiben, dar.

3.1.3.4 Multimedia-Formate

Im multimedialen Umfeld des WWW gebräuchliche Formate sind für Grafiken GIF, JPEG und PNG, für Video MPEG und AVI und für Audio MP3. Eine Vielzahl von anderen Formaten ist bekannt und wird auch genutzt, erreicht aber nicht den Verbreitungsgrad der oben genannten Formate. Im folgenden werden die oben genannten Typen kurz beleuchtet.

- Das GIF (Graphics Interchange Format) wurde 1987 von *Compuserve* entwickelt, um Grafiken anzeigen zu können. Das Format erlaubt die Verwendung von 256 Farben, Kompression, Überblendung und Animationen. GIF wird hauptsächlich für Grafiken mit einer klaren Trennung von Farben genutzt.

- JPEG (Joint Photographic Experts Group) ist ein effizientes, komprimierbares Echtfarben-Grafikformat. Es setzt verlustbehaftete Kompression ein, um hohe Kompressionsfaktoren realisieren zu können. Im Gegensatz zu GIF unterstützt JPEG Millionen von Farben. JPEG wird am sinnvollsten eingesetzt, wenn Grafiken mit Gradienten, Übergängen und inkonsistenten Farbvariationen vorliegen, wie dies bei Fotografien oder Bildern der Fall ist. Grafiken mit klar separierten Farben sollten im GIF oder besser im PNG-Format abgelegt werden.

- PNG (Portable Network Graphics) ist ein noch vergleichsweise neues Format, das die Vorherrschaft von GIF und JPEG bedroht. PNG unterstützt wie GIF die verlustfreie Kompression, kann jedoch mit Farbtiefen von mehr als 24 Bit agieren. Anders als bei GIF ist der Kompressionsalgorithmus nicht über ein Patent geschützt und kann deshalb frei eingesetzt werden. Weiterhin können PNG-Grafiken teilweise transparent sein, ein Vorteil bei der Platzierung auf Web-Seiten.

- MPEG(Motion Pictures Expert Group) steht für Dateiformate und Verfahren zum platzsparenden Komprimieren und Speichern von Video- bzw. Multimediadaten (Video, Bild- und Tondaten) in hoher Qualität. Der MPEG-Standard unterteilt sich inzwischen in MPEG-1, MPEG-2, MPEG-3 und MPEG-4, wobei der MPEG-3-Standard mittlerweile in MPEG-2 integriert wurde.

- AVI (Audio Video Interleave) ist ein ursprünglich in der Windows-Welt beheimatetes Audio-/Videoformat, bei dem Audio- und Videodaten ineinander verzahnt, also „interleaved“, abgespeichert werden. Die erste Definition von AVI ist so alt wie die Multimedia-PCs. Das Format wurde von *Microsoft* als einheitliche Lösung für die Wiedergabe von kurzen Videoclips kreiert.

- MP3 ist eine Multimedia-Entwicklung zur effektiven Komprimierung von Sound/Musik, das von der Fraunhofer-Gesellschaft (FhG) entwickelt wurde. MP3 ist nicht mit MPEG-3 zu verwechseln, sondern steht für „MPEG 2.5 Audio Layer III“. MP3 ist ein Audio-Format, das eine hohe Kompression von Audio-Daten bei sehr geringem Qualitätsverlust ermöglicht.

Abgesehen von den hier in Kürze beschriebenen Formaten bietet das WWW noch zahlreiche weitere, die für das Web-Präsenz-Management relevant sind. Allen ist gemeinsam, dass sowohl für ihre interne Verarbeitung und Verwaltung im System als auch ihre externe Repräsentation Spezialanwendungen benötigt werden. Auf der Client-Seite erfolgt die Darstellung solcher Formate über die Einbindung entsprechender Anwendungen als Plug-ins in den Web-Browser. Auf der Seite des Content-Managements gibt es zwei verschiedene Ansätze, mit multimedialen Dateien umzugehen: der einfache Ansatz, den auch JDaphne (siehe Seite 116) verfolgt, betrachtet solche multimedialen Objekte als binäre Dateien, die als Ganzes vom System verwaltet werden. Analog zu den oben erwähnten PDF-Dateien werden Metadaten zu den Dateien verwaltet; diese werden allerdings nicht automatisch aus dem Objekt extrahiert. Abgesehen von den zum Teil gewaltigen Datenvolumina, die multimediale Dateien mit sich bringen, werden an das Content-Management in diesem Fall keine Anforderungen gestellt, die über eine textuelle Beschreibung der Objekte hinausgeht. Kompliziertere, zum Teil noch in der Entwicklung (z.B. der Prototyp für PADL von Goh und Leggett [GL00] oder der „Multi-View Intelligent Editor“ von Myers et al. [MCS+01]) befindliche, multimediale Content-Management-Systeme für digitale Bibliotheken zielen darauf ab, die multimedialen Dateien (z.B. Videos) in ihre Bausteine zu zergliedern und auf diese Weise den Autoren z.B. gezielt Zugriff auf einzelne Sequenzen des Videos zu bieten (vgl. Multimedia-Autorensysteme, Boles und Schlattmann [BS98]).

3.1.3.5 Einfacher Text

Ein weiterer auch heute noch auftretender Dokumenten-Typ ist der unformatierte unstrukturierte Text. Für die Verwendung in einem Content-Management-System ist unstrukturierter Text nur bedingt geeignet. Aufgrund der fehlenden Strukturierung ist es den Systemen nicht möglich, explizite Struktur-Informationen für eine weiterführende Verwendung des Dokumentes zu nutzen. Lediglich die Ausnutzung von speziellen, impliziten Konventionen für konkrete Dokumente kann zum Aufbau einer Struktur-Information ausgenutzt werden und einem Content-Management-System die Verarbeitung des Dokumentes ermöglichen. Fehlt bei einem Dokument die Strukturierung, so kann sie nicht für den Aufbau eines an die Struktur-Informationen bzw. semantische Bedeutungen geknüpften Layouts genutzt werden. Lediglich eine globale Formatierung des Textes mit Schriftarten und die Einbindung in ein Gesamtlayout sind dann möglich. In der modernen Informationswelt ist die Bedeutung von unstrukturiertem Text aufgrund der Probleme, die daraus für das Content-Management erwachsen, stark im Abnehmen begriffen.

3.1.4 Software-Architekturen

3.1.4.1 Verteilte Anwendungen

Ein „verteiltes System“ ist eine Ansammlung autonomer Rechner, welche durch ein Netzwerk miteinander verbunden sind und über eine Software für verteilte Systeme verfügen, die integrierte Berechnungen ermöglicht (vgl. [CDK00]). Eine verteilte Anwendung ist eine Anwendung, deren Funktionalität in eine Menge von kooperierenden Teilkomponenten (vgl. Abb. 3.1) zerlegt ist. Jede Teilkomponente hat einen internen Zustand. Die Teilkomponenten sind autonome Verarbeitungseinheiten, die auf verschiedene Rechner abgebildet werden können. Jede Teilkomponente läuft als Prozess auf dem ihr zugeordneten Rechner ab. Teilkomponenten tauschen dabei über das Netzwerk untereinander Informationen durch Funktionsaufrufe in darauf ausgelegten Protokollen und Architekturen (CORBA [Pop98], RMI [Far98], DCOM [RBR98] etc.) aus. Die Vorteile verteilter Anwendungen liegen in dem Teilen von Ressourcen, der Flexibilität und Modularität, Skalierbarkeit und Fehlertoleranz. Es ist möglich, verteilte Systeme durch zusätzliche Rechner zu erweitern, ohne das Verhalten der ursprünglichen Systeme zu ändern. Ein verteiltes System bleibt verfügbar, auch wenn Fehler in der Hard- oder Software auftreten. Nachteile verteilter Anwendungen gegenüber zentraler Anwendung sind die geringere Sicherheit durch zusätzliche Angriffsstellen, aufwendigere Entwicklungsarbeit sowie ein Verwaltungsüberhang, der durch die Kommunikation der Komponenten untereinander anfällt. Beispiele für komplexe verteilte Systeme sind der Domain Name Service oder das E-Mail im Internet.

3.1.4.2 Client-/Server-Architektur

Eine Client-/Server-Anwendung (Abb. 3.2) ist ein Beispiel für eine einfache verteilte Anwendung. Bei ihr stehen im vernetzten Rechnerverbund eine geringere Anzahl dezidierter, dienstanbietender Rechner (Server) einer größeren Zahl dienstanfordernder Arbeitsplatzrechner (Clients) gegenüber. Die in der Regel leistungsstärkeren Server erfüllen im Netzverbund meist spezialisierte und teilweise auf sie zentralisierte Aufgaben, die sie ihrer Systemumgebung als Block von Diensten anbieten; insbesondere Web-Server sind solche Dienstanbieter. Die Clients, meist PCs oder Workstations, führen ihnen angemessene Aufgaben, z.B. im Fall des WWW das Darstellen der vom Server geladenen Dokumente, unter Nutzung ihrer lokalen Rechen- und Speicherkapazitäten aus und fordern bei darüber hinausgehenden

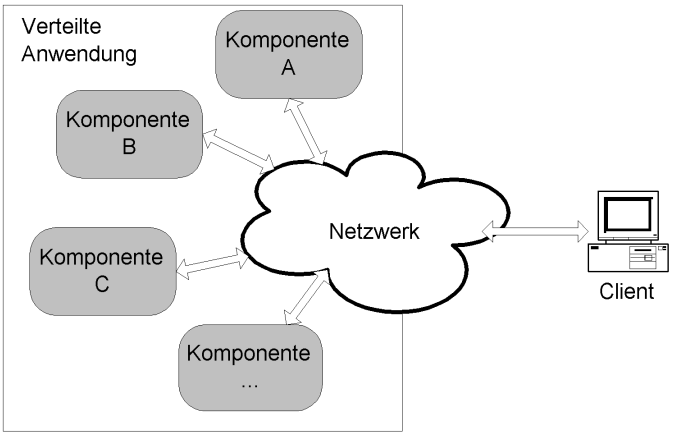

Abbildung 3.1: Aufbau einer verteilten Anwendung.

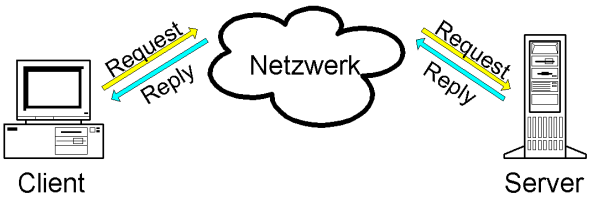

Abbildung 3.2: Eine Client-/Server-Architektur. Der Client sendet Anfragen (Requests) an den Server. Dieser bearbeitet die Anfrage und sendet ein Ergebnis (Reply) zurück.

Verarbeitungsschritten die Dienste der Serverstationen an, mit denen sie über das Netz Informationen austauschen. Der Aufruf von Server-Diensten erfolgt über beim Client erstellte Anfragen (Requests), die über geeignete Protokolle zu den Servern übermittelt werden. Die entsprechenden Server bearbeiten die Anfragen und schicken die Ergebnisse (Replies) an die Client-Rechner zurück.

3.1.4.3 Multi-Tier-Architekturen

Client-/Server-Architekturen liegt eine Gliederung in einzelne Software-Schichten zugrunde. Dabei werden Software-Komponenten auf mehreren Rechnern verteilt. In Multi-Tier-Architekturen (siehe Abb. 3.3) werden die Anwendungskomponenten auf mehrere Schichten verteilt. Dabei ist die Präsentationsschicht auf dem Client-Rechner installiert während die Daten auf einem Back-End-Server liegen. Auf den dazwischen liegenden mittleren Schichten sind weitere Server implementiert, die sich zum Beispiel gegenüber einem Back-End-Server wie ein Client verhalten und bestimmte Services anfordern, während sie sich gegenüber dem Client-Rechner wie ein Server verhalten, der ihnen Dienste zur Verfügung stellt.

3.1.4.3.1 One-Tier-Architekturen

Als One-Tier-Architekturen werden monolithische Modelle, bei denen lediglich die Präsentation auf einem „dummen“ Terminal dargestellt wird, die komplette Anwendungsverarbeitung einschließlich aller Datenbank-Zugriffe aber auf einem zentralen (Groß-)Rechner abläuft, bezeichnet.

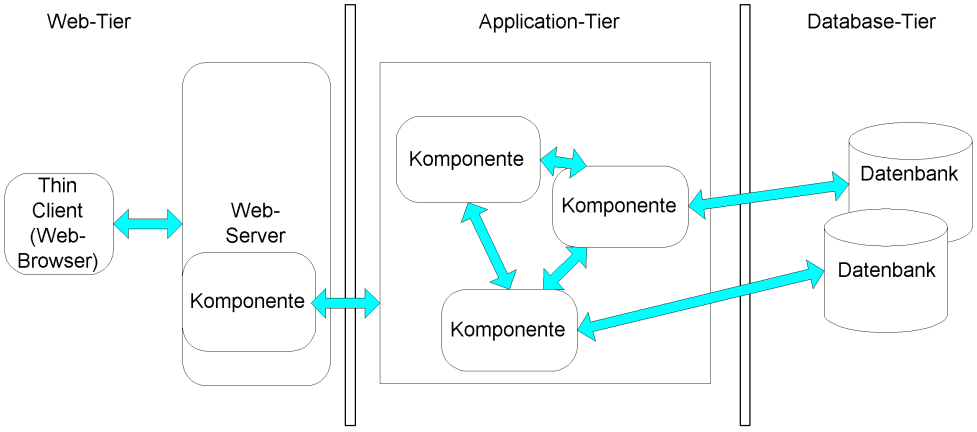

Abbildung 3.3: Beispiel für eine Multi-Tier-Architektur.

3.1.4.3.2 Two-Tier-Architekturen

Tow-Tier-Architekturen teilen die Anwendung in zwei Schichten auf: sie arbeiten mit einem Server im Back-End, auf dem die Datenbank gespeichert ist; die Benutzerschnittstelle liegt auf dem Client-Rechner. Die Applikationslogik kann hier entweder auf dem Server oder auf dem Client-Rechner liegen.

3.1.4.3.3 Three-Tier-Architekturen

In einer Three-Tier-Architektur wird die Anwendung in drei Schichten aufgeteilt. Auf dem Front-End, dem Client-Rechner, liegt die Benutzerschnittstelle, im Back-End sind die Daten gespeichert. Dazwischen wird eine sogenannte „Middle-Tier“ implementiert (häufig auch als Applikations-Server bezeichnet), die für Steuerung und Verarbeitung verantwortlich ist. Einzelne Teile der Anwendungsverarbeitung können auch auf dem Client-Rechner laufen.

3.1.5 Die Programmiersprache Java

In ihren Objekten und ihrer Struktur speziell auf den Gebrauch im Internet ausgelegt, ist Java [GM95] zur Zeit die Programmiersprache der Wahl, wenn eine plattformunabhängige Anwendung entwickelt werden soll. Das Konzept von Java in seiner Vollständigkeit zu besprechen steht nicht im Zentrum dieser Abhandlung. Da die im Rahmen dieser Arbeit entwickelten Anwendungen aber zu einem großen Teil in Java entwickelt wurden, werden in den folgenden Paragraphen die wichtigsten Konzepte aus dem Java-Umfeld skizziert.

3.1.5.1 Virtual-Machine-Konzept

Ihre Plattformunabhängigkeit hat diese Programmiersprache dem Konzept der virtuellen Maschine (virtual machine, VM) [TL97a] zu verdanken. Diese wird nativ für jede Plattform in standardisierter Form implementiert und stellt damit eine definierte Umgebung für den Ablauf von Java-Programmen dar. Diese Java-Programme werden dementsprechend nicht in nativen Code übersetzt, sondern in einen Meta-Code – einen Byte-Code, der speziell auf die virtuelle Maschine abgestimmt ist. Dieser Byte-Code wird von der virtuellen Maschine als Laufzeitumgebung interpretiert. Er ist unabhängig von der Plattform, auf der die ausführende VM installiert ist. Da der Byte-Code interpretiert wird, ist die Performance schlechter als bei nativem Code. Durch Konzepte wie die Laufzeitkompilation (just in time

compilation, [Eck00]) wird die Performance von Java spürbar verbessert. Bei der Laufzeitkompilation wird im Rahmen der Ausführung ein nativer Maschinen-Code erzeugt und für weitere Durchläufe der gleichen Programmelemente konserviert. Bei mehrfachem Durchlauf analoger Code-Fragmente wird auf diese Weise ein signifikanter Performance-Gewinn erzielt.

3.1.5.2 Java-Applets

Den Einstieg hat Java als Programmiersprache durch das Applet-Konzept geschafft. Applets sind, den schmalen Bandbreiten im Internet Rechnung tragend, kleine, mobile Applikationen, die über das Internet transportiert werden können. Durch die Implementierung von virtuellen Maschinen in Web-Browsern wurden diese in die Lage versetzt, Programm-Code auszuführen, der unabhängig von der Plattform ist, auf der der Web-Browser installiert ist. Die virtuelle Maschine stellt bereits ein mächtiges API (Application Programming Interface) zur Verfügung. Dadurch können Applets mit wenig eigentlichem Programm-Code realisiert werden. Konzepte wie die Java-Archive [Eck00] bieten durch ihre Möglichkeit, zum einen alle benötigten Klassen in einem einzigen Archiv zusammenzufassen und dieses zum anderen zusätzlich noch zu komprimieren, deutliche Vorteile beim Download der Applets.

So bestechend das Konzept des mobilen Java-Codes auf den ersten Blick auch erscheint, es weist ebenfalls Nachteile auf. Zwei davon werden im Umfeld des in dieser Arbeit beschriebenen Online-Redaktionssystems relevant: Der erste Nachteil von Java-Applets ist ihre Vertrauenswürdigkeit; zum einen ist dem Benutzer, der ein Java-Applet über das Internet lädt, nicht klar, wer das Applet erstellt hat, zum anderen könnte der Code des Applets während des Datentransfers modifiziert worden sein. In beiden Fällen stellt sich dem Benutzer, der den Code ausführt, die Frage, ob er wirklich mit dem Code arbeiten möchte. Um diese Problematik einzugrenzen, wurde von vornherein vorgesehen, dass Java-Applets von dem Web-Browser nur in einer Sicherheitsumgebung, der Sandbox, ausgeführt werden können. Die Sandbox verhindert, dass das Applet Zugriff zu den Ressourcen des Rechners, auf dem es gerade aktiv ist, erhält. Damit ist es Java-Applets u.a. verwehrt, in dem Dateisystem des Gast-Rechners Dateien zu lesen oder zu schreiben. Auf diese Weise wird eine Funktionalität verhindert, die im Rahmen des Redaktionssystems für die Java-basierte Benutzerschnittstelle benötigt wird. Abhilfe für dieses Problem schafft die digitale Signierung des mobilen Codes (siehe Seite 53). Nach dem Verpacken des Applet-Codes in ein Archiv kann dieses von dem Entwickler signiert werden. Vor dem Ausführen des Codes auf dem Arbeitsplatz des Benutzers wird die Signatur überprüft. Sofern die Signatur in Ordnung ist und der Benutzer dem Entwickler, der den Code signiert hat, vertraut, kann er nun dem Applet die Erlaubnis erteilen, auch außerhalb der Sandbox zu agieren.

Der zweite Nachteil besteht in den vielen verschiedenen Java-Versionen, die im Laufe der Zeit herausgegeben wurden. Diese haben zur Folge, dass kein Web-Browser die aktuelle Version von Java in seiner VM verarbeiten kann. Hinzu kommt noch, dass der *Internet Explorer* von *Microsoft* eine nicht dem Standard von *SUN* entsprechende Implementation der VM besitzt und neben einem anderen Sicherheitsmodell auf eine Nutzung von Windows-spezifischen Funktionalitäten abzielt.

Für die Plattformabhängigkeit von Java auf der Client-Seite ist insbesondere der letzte Punkt ein großer Rückschlag. Abhilfe aus dieser Problematik schafft das Java-Plug-in von *SUN*. Dieses bietet in den Web-Browser als Plug-in (siehe Seite 39) eingebunden eine Laufzeitumgebung für Applets in einer frei selektierbaren Version. In jedem Fall wird immer auch die aktuellste Version von Java unterstützt, ein Vorteil insbesondere bei der Entwicklung mit den neuesten Technologien.

3.1.5.3 Java-Applikationen

Während Java zu Beginn aufgrund der vergleichsweise schlechten Performance zunächst hauptsächlich durch das Applet-Konzept von sich reden machte und auf der Server-Seite nur wenig Beachtung fand, hat mittlerweile eine Trendwende stattgefunden. Zum einen ist das API von Java um viele, gerade im Internet-Umfeld und dort im Enterprise-Bereich notwendige, Klassen und Methoden gewachsen. Zum anderen ist mit den oben erwähnten „just in time-Compilern“ (siehe Seite 51) ein deutlicher Performance-Zuwachs erzielt worden. Trotz der auch dann noch geringeren Performance als bei nativen Anwendungen überwiegen oft die Vorteile der Plattformunabhängigkeit und Portabilität des Codes.

Mittlerweile werden zahlreiche Server-Applikationen in Java implementiert. Um Performance zu gewinnen, werden zum großen Teil die eigentlichen Server-Programme noch in performanteren Programmiersprachen implementiert. Diese Server dienen dann als Laufzeitumgebungen, in die Java-Komponenten eingebunden werden können. Während so die Infrastruktur performant gehalten wird, können portable, gegebenenfalls noch durch interne Compiler performance-optimierte Java-Komponenten, die die eigentliche Verarbeitungslogik beherrschen, in diese Server eingeklinkt werden. Dieses Verfahren wird momentan bei den Application-Servern (siehe Seite 180) eingesetzt.

3.1.5.4 Java-Servlets

Servlets [Hal00] stellen das serverseitige Analogon zu den Applets dar. In einer vom Web-Server bereitgestellten Variante der virtuellen Maschine ist die Ausführung von serverseitigem Programm-Code möglich. Die Parametrisierung erfolgt dabei analog zu dem CGI [Gun00]. Servlets sind durch das mächtige API von Java für Server-Anwendungen sehr kompakt programmierbar.

3.1.6 Sicherheitsmechanismen

Bei Systemen, die auf Internet-Technik basieren, spielen Sicherheitsüberlegungen eine besonders große Rolle, da es sich in den meisten Fällen um, im Vergleich zu konventionellen Systemen, sehr offene Konstrukte handelt, die mit großen Nutzerzahlen arbeiten. Gerade diese Offenheit der Systeme erfordert aber spezielle, standardisierte Sicherheitsmechanismen, vgl. dazu die Werke von Raepple [Rae01] und Stalling [Sta01], die sowohl den Zugang (Authentisierung) zum System, die Berechtigung zu Aktionen (Autorisierung) als auch die Datenübermittlung zwischen den Systemkomponenten betreffen.

3.1.6.1 Secret- und Public-Key-Verfahren

Moderne Sicherheitsarchitekturen unterscheiden zwischen symmetrischen und asymmetrischen Verfahren zum Verschlüsseln der Daten. Die momentan im Internet favorisierte Lösung zur Realisierung von Sicherheitsinfrastrukturen beruht vorwiegend auf dem Public-Key-Verfahren, kombiniert es aber für performance-kritische Anwendungen mit dem Secret-Key Verfahren.

3.1.6.1.1 Secret Key Verfahren

Bei den symmetrischen Verfahren ist der Schlüssel zum En- und Decodieren der Daten identisch. Er ist geheim zu halten – daher Secret-Key-Verfahren – und die Schwierigkeit besteht im Transport des Schlüssels von A nach B, wenn eine bei A verschlüsselte Nachricht bei B entschlüsselt werden soll. Der Vorteil von symmetrischen Verfahren besteht in der höheren Per-

formance bei der Verschlüsselung. Bekannte Verfahren zur Erzeugung symmetrischer Schlüssel sind:

- DES, triple-DES (Data Encryption Standard) [Sch96]. Bei DES handelt es sich um einen Algorithmus, der mit der sogenannten Produktverschlüsselung arbeitet, bei der als elementare Verschlüsselung Substitutionen und Transpositionen (Permutationen) verwendet werden.
- IDEA(International Data Encryption Algorithm) [LM90]. Bei IDEA handelt es sich um einen blockorientierten konventionellen Verschlüsselungsalgorithmus.

Eine Reihe weiterer Verfahren ist bekannt, zum Einsatz kommen allerdings in Abhängigkeit von der letztendlichen Anwendung vorwiegend die beiden hier genannten Verfahren.

3.1.6.1.2 Public-Key-Verfahren

Jedem Teilnehmer dieses Verfahrens wird ein digitales Schlüsselpaar, bestehend aus einem privaten und einem öffentlichen Schlüssel, generiert. Wie der Name schon andeutet, stellt der private Schlüssel das Geheimnis des jeweiligen Besitzers dar. Der öffentliche Schlüssel hingegen wird „publiziert“. Das Prinzip des Public-Key-Verfahrens beruht darauf, dass die Schlüssel wechselseitig Nachrichten ver- und entschlüsseln können. Mit dem öffentlichen Schlüssel encodierte Nachrichten können nur mit dem privaten Schlüssel decodiert werden. Umgekehrt können mit dem privaten Schlüssel encodierte Nachrichten nur mit dem öffentlichen Schlüssel decodiert werden. Bekannte Algorithmen, die für die Generierung von solchen Schlüsselpaaren eingesetzt werden, sind:

- RSA (Rivest-Shamir-Adleman) [RSA78]. Das RSA-Verfahren hat sich mittlerweile bewährt und ist eines der am besten analysierten asymmetrischen Verfahren überhaupt. Die Sicherheit basiert auf der Schwierigkeit, eine große natürliche Zahl zu faktorisieren. Bei entsprechend großem Schlüssel ist eine Entschlüsselung (beim derzeitigen Stand der Technik und der Kryptoanalyse) nicht in einem akzeptablen Zeitraum möglich. RSA war in den USA patentiert, außerhalb der Staaten aber frei nutzbar. Das Patent ist am 20. September 2000 abgelaufen.
- ElGamal [Sch96], ein auf dem diskreten Logarithmusproblem beruhendes Kryptosystem.

Asymmetrische Verschlüsselung ist um ein Vielfaches rechenaufwendiger als symmetrische Verschlüsselung. In der Praxis werden aus Gründen der Effizienz beide Verfahren daher kombiniert eingesetzt. Unter Ausnutzung des Public-Key-Verfahrens werden die symmetrischen Schlüssel mit den asymmetrischen encodiert und so ausgetauscht. Für den weiteren Datentransfer wird der über dieses sichere Verfahren ausgetauschte symmetrische Schlüssel eingesetzt.

3.1.6.2 Digitale Signatur

Die heute bekannten digitalen Signaturen nach dem Signatur-Gesetz [I01] beruhen durchgehend auf dem Public-Key-Verfahren. Der Unterzeichner encodiert als Ersatz einer Unterschrift das zu signierende Dokument mit seinem privaten Schlüssel, der sein Geheimnis, seine Identifikation, darstellt. Da privater und öffentlicher Schlüssel eine Einheit sind, ist gewährleistet, dass, wenn das verschlüsselte Dokument mit dem öffentlichen Schlüssel decodiert werden kann, es zwangsläufig von dem Besitzer des privaten Schlüssels encodiert (signiert) worden

sein muss. Da das Verschlüsseln von kompletten Dokumenten zum Zweck der digitalen Unterschrift zum einen sehr aufwendig ist, zum anderen einen deutlichen Datenüberschuss nach sich zieht, wird für die digitale Signatur nur eine eindeutige Kennzeichnung (ein Hash-Wert) des Dokumentes mit dem privaten Schlüssel encodiert. Diese eindeutige Kennzeichnung wird durch einen „Hash-Algorithmus“ [Pre93], im Allgemeinen entweder mit dem Message Digest (MD2, MD4, MD5) oder SHA, ermittelt. Dieser ist eine eindeutige Abbildung des digitalen Dokuments, die sich dementsprechend in ihrem Ergebnis ändert, sobald auch nur ein Bit des Dokumentes modifiziert wurde. Zur Verifizierung der digitalen Signatur berechnet der Verifizierende ebenfalls den Hash-Wert des ihm vorliegenden Dokuments. Danach entschlüsselt er mit dem öffentlichen Schlüssel des Signierenden die ihm vorliegende Signatur. Konnte die Signatur decodiert werden und stimmt der so erhaltene Hash-Wert mit dem selbst berechneten überein, so ist die digitale Signatur wirklich von dem Signierenden geleistet worden und es ist sichergestellt, dass an dem Dokument zwischenzeitlich keine Veränderungen vorgenommen wurden. Neben dem oben beschriebenen RSA-Algorithmus kommt für digitale Signaturen hauptsächlich der Digital Signature Algorithm (DSA) des National Institute of Standards and Technology (NIST) zum Einsatz. DSA basiert auf dem diskreten Logarithmus Problem und benutzt die Kryptosysteme von Schnorr [Sch89] und ElGamal (s.o.). Eine detaillierte Beschreibung von DSA findet sich in der zugehörigen Veröffentlichung vom NIST [Nat92].

3.1.6.2.1 Digitales Zertifikat

Ein digitales Zertifikat ist im Wesentlichen die personalisierte Version des öffentlichen Schlüssels aus dem Schlüsselpaar. Zu den Personalisierungsinformationen gehört im Normalfall der Name des Eigentümers, der Name der ausgebenden Zertifizierungsstelle und der Gültigkeitszeitraum. Über die Personalisierungsinformation, deren Form von dem verwendeten Standard, z.B. X509 [HFPS99], abhängt, wird der öffentliche Schlüssel seinem Besitzer zugeordnet. Digitale Zertifikate werden deshalb auch mit digitalen Ausweisen verglichen. Der Eigentümer des Schlüsselpaares identifiziert sich durch den Einsatz seines privaten Schlüssels. In Verbindung mit dem privaten Schlüssel dienen Zertifikate der Authentisierung.

3.1.6.2.2 Public-Key-Infrastructure (PKI)

Um digitale Zertifikate sinnvoll einsetzen zu können, wird eine funktionierende Infrastruktur (PKI) benötigt. Die PKI betrifft alle mit dem Lebenszyklus eines Zertifikates verbundenen Elemente. Das Zertifikat wird ausgestellt von einer Zertifizierungsstelle [HFPS99](Certificate Authority, CA). Dazu wird ein Schlüsselpaar erzeugt und nach Authentisierung des Beantragenden mit dessen Daten personalisiert. Anschließend wird das Zertifikat an den Eigentümer übergeben. Dies kann auf verschiedenen Wegen geschehen; denkbar sind beispielsweise die Speicherung auf Smart Cards, Disketten oder der direkte Download mit dem Web-Browser. Nach der Ausgabe des Zertifikates wird dieses veröffentlicht, ein Vorgang der in erster Linie über das Publizieren in vertrauenswürdige Verzeichnisdienste geschieht. Alternativ müsste der Eigentümer des Zertifikates dieses persönlich an seine Kontakte übergeben. Anhand der Einträge in den Verzeichnisdiensten kann die Korrektheit, und ebenso wichtig, die Gültigkeit des Zertifikates zu jedem Zeitpunkt überprüft werden. Wenn der private Schlüssel kompromittiert wurde, muss das Zertifikat widerrufen werden. Dazu wird es entweder aus dem Verzeichnisdienst entfernt oder auf eine „Schwarze Liste“, die Certificate Revocation List (CRL), gesetzt. Unter einer PKI ist darüber hinaus zu verstehen, dass sich der Inhaber eines Zertifikates mit diesem gegenüber internetbasierten Anwendungen authentisieren kann. Die Anwendung muss dazu in der Lage sein, die Gültigkeit des übermittelten Zertifikates gegen den Verzeichnisdienst zu überprüfen.

3.1.6.3 Authentisierung/Autorisierung

Authentisierung beschreibt den Vorgang, bei dem sich ein Anwender dem System gegenüber ausweist. Nur wenn das System den Benutzer authentisiert hat [Sta01], darf dieser das System einsetzen. Oft werden anhand der Authentisierung auch die dem Benutzer zugeordneten Rollen und Rechte innerhalb des Systems vergeben (Autorisierung). Während viele Systeme nur einfache Authentisierungsverfahren einsetzen, kommt diesem Punkt bei sicherheitskritischen Systemen große Bedeutung zu. Neben einer Art elektronischen Ausweises, dem Zertifikat, das z.B. auf einer Chip-Karte abgelegt werden kann, werden zunehmend biometrische Verfahren diskutiert (s.u.).

3.1.6.3.1 Benutzerkennung-Passwort-Paar

Die einfachste und bislang noch weitverbreitetste Form der Authentisierung ist die Verwendung von Benutzerkennung-Passwort-Paaren. Eine Person wird über die Angabe eines geheimen Passwortes zu einer gegebenenfalls öffentlichen Benutzerkennung identifiziert. Dabei können verschiedene Verfahren zur sicheren Übermittlung des Passwortes eingesetzt werden, die verhindern sollen, dass Passworte „abgehört“ werden. Wegen seiner Einfachheit ist diese Form der Authentisierung beliebt. Das Verfahren weist jedoch einige Schwächen, die dazu führen, dass sicherheitskritische Anwendungen Abstand davon nehmen.

3.1.6.3.2 Zertifikate

Zertifikate [Koh78] bieten einen deutlich höheren Sicherheitsstandard bei der Authentisierung als das zuvor geschilderte Verfahren. Damit das Zertifikat mit einem elektronischen Ausweis verglichen werden kann, müssen die zu ihm gehörenden privaten Schlüssel gut geschützt verwahrt werden. Um die Authentisierung mit Zertifikaten durchführen zu können, wird zwingend eine funktionierende PKI benötigt. Es muss jederzeit möglich sein, Zertifikate zu widerrufen und ihren Zustand zu überprüfen. Ein Verzeichnisdienst ist deshalb für den Einsatz von Zertifikaten unumgänglich.

3.1.6.3.3 Biometrische Verfahren

Für Internet-Anwendungen kaum in Benutzung, aber trotzdem mit einem großen Potential versehen, sind die momentan noch in der Entwicklung befindlichen biometrischen Authentisierungsverfahren [JHP00]. Basierend auf eindeutigen Körpermerkmalen, wie z.B. dem Abbild der Retina, Fingerabdrücken etc., findet bei diesem Verfahren die Identifizierung und Authentisierung statt [Poh01].

3.1.6.4 Verschlüsselte Datenübermittlung

Da die Übermittlung von Daten bei internetbasierten Systemen über offene Netzwerke stattfindet, werden in sicherheitsrelevanten Fällen Verschlüsselungsmechanismen eingesetzt. Abgesehen von der Verschlüsselung des eigentlichen Datenblocks bei ansonsten offener Kommunikation, z.B. bei verschlüsselter E-Mail, werden häufig Verfahren eingesetzt, bei denen nach dem Austausch eines Schlüssels für eine Sitzung (Session-Key) die gesamte Kommunikation zwischen den Komponenten verschlüsselt abläuft. Solche sicheren Verbindungen zu betreiben, ist aufgrund der höheren erforderlichen Rechenleistung deutlich aufwendiger als bei unverschlüsselten Verbindungen. Bei der Auswahl der Hardware für einen Server muss dies unbedingt berücksichtigt werden. Standardprotokolle für die verschlüsselte Datenübermittlung im Internet sind SSL (Secure Socket Layer) [FKK96] bzw. darauf aufbauend HTTPS [Sta01]. Diese auf dem Public-Key-Prinzip basierenden Protokolle tauschen zur Authentisierung zunächst

die öffentlichen Schlüssel der beiden Partner (Web-Server-Betreiber und Benutzer am Web-Browser) beispielsweise in Form von Zertifikaten aus. Werden die öffentlichen Schlüssel akzeptiert, wird von dem Web-Server ein symmetrischer Schlüssel generiert. Encodiert mit dem öffentlichen Schlüssel des Benutzers wird er an dessen Web-Browser übertragen. Dieser decodiert ihn mit dem privaten Schlüssel und verfügt ab sofort für den symmetrischen Schlüssel, mit dem die im Weiteren zwischen Web-Server und Web-Browser übertragenen Daten verschlüsselt werden.

3.2 Web-Präsenzen

Die bislang in diesem Kapitel dargestellten Techniken und Konzepte sind allgemeiner Natur und werden zur Entwicklung von webbasierten Anwendungen eingesetzt. Mit der speziellen Zielrichtung, ein webbasiertes System zur Verwaltung einer Web-Präsenz aufzubauen, wird in diesem Abschnitt ein Überblick über die dafür benötigten Ansätze und Begrifflichkeiten gegeben. Zu Beginn dieses Abschnittes werden die Grundlagen einer Web-Präsenz besprochen; dabei werden in diesem ersten Schritt ihre Basiskomponenten erläutert. An die Betrachtung des Lebenszyklus' einer Web-Präsenz schließt sich die Frage nach Bewertungskriterien und formalen Zielsetzungen beim Aufbau einer Web-Präsenz an. Abschließend wird die Sicherheitsfrage beim Betrieb einer Web-Präsenz beleuchtet und auf ihre Performance eingegangen.

3.2.1 Grundlagen

Von grundlegender Bedeutung beim Umgang mit einer Web-Präsenz ist die Kenntnis der Basiskomponenten, aus denen sie sich zusammensetzt.

3.2.1.1 Die Basiskomponenten einer Web-Präsenz

Bei der Erstellung einer Web-Präsenz werden drei Basiskomponenten verwendet, diese sind:

- Content
- Navigation
- Layout

Sie werden bei einem effizienten WCM separat verwaltet. In einem als Generierung oder Export bezeichneten Vorgang werden sie für die Web-Präsenz zusammengeführt. Erst die so erstellten Dokumente können von dem Web-Browser (siehe Seite 38) des Benutzers der Web-Präsenz dargestellt werden und erlauben es ihm, durch die Web-Präsenz zu „browsen“. Die folgenden Abschnitte erläutern die Basiskomponenten mit ihrer jeweiligen Funktion.

3.2.1.1.1 Content

Der Begriff „Content“ bezeichnet im Zusammenhang mit Web-Präsenzen den eigentlichen zu vermittelnden Inhalt, die Botschaft der Web-Präsenz. Je nach Zielrichtung kann der Content unterschiedliche Ausprägungen haben. Für eine Online-Zeitung stellt jeder einzelne Artikel, zusammen mit den zugehörigen Grafiken, den Content dar, für eine Auktions-Site sind dies die Angebote und Gebote, in einer Unternehmens-Web-Präsenz sind die Unternehmensinformationen sowie die Produktbeschreibungen der Content. Aus der Sicht des WCM können zwei verschiedene Arten von Content-Objekten technisch unterschieden werden: zum einen die Medienobjekte und zum anderen die Applikationsobjekte. Zur ersten Gruppe zählen die aus klassischen Multimedia-Präsentationen bekannten Objekte; Texte und Grafiken, Zeichnungen

und Bilder werden hier durch Audio- und Videodateien ergänzt. Gegebenenfalls können Animationen und Simulationen eingebunden werden. Neu im Kontext der Web-Präsenz sind Inhalte in Form der Applikationsobjekte. Zu ihnen gehören die im Normalfall benutzerinteraktiven Komponenten wie Formulare, Datenbankanbindungen, Java-Applets und ActiveX-Komponenten. Weitere Applikationsobjekte werden insbesondere bei den dynamisch generierten Web-Präsenzen auf der Server-Seite eingesetzt (siehe Seite 68). Die Verwaltung von Medien- und Applikationsobjekten gehört zu den Kernaufgaben des WCM.

3.2.1.1.2 Navigation

Nach der Eingabe der URI einer Web-Präsenz in einen Web-Browser findet sich der Benutzer bei seinem ersten Kontakt auf der Startseite des Anbieters wieder. Um den Besucher zu den ihn interessierenden Informationen zu führen, müssen Pfade vorbereitet werden, an denen er sich orientieren kann [SG99]. Solche Pfade können beispielsweise inhaltliche Verzeichnisstrukturen sein. Durch das „Öffnen“ eines Verzeichnisses gelangt der Besucher zu immer spezielleren Inhalten und kann sich durch das Selektieren von Unterverzeichnissen bis zu der gewünschten Information navigieren.

Bei dem Aufbau solcher Navigationsstrukturen ist viel Sorgfalt gefordert. Zu beachten ist, dass die Wege zu den Informationen so kurz wie möglich gehalten werden sollten. Daher ist i.d.R. bei Web-Präsenzen eine Tendenz wegführend von tiefen, hin zu flachen, aber breiter angelegten Inhaltsstrukturen zu beobachten. Weiterhin ist bei ihrem Aufbau zu beachten, dass es verschiedene Wege zum gleichen Ziel geben kann und sollte. Nicht jeder Benutzer hat die gleiche inhaltliche Strukturierung einer Thematik vor Augen, wenn er auf der Suche nach einer bestimmten Information ist. Hier hilft es, in Analysen mit verschiedenen Probanden mehrere mögliche Lösungen zu evaluieren, bevor eine ausgezeichnet wird. Für die Bereitstellung von Navigationsmöglichkeiten innerhalb einer Web-Präsenz gibt es eine Reihe verschiedener Navigationselemente, die im Folgenden besprochen werden.

Bezüglich der Navigation gibt es auf einer Web-Präsenz für jedes Dokument eine Reihe markanter Punkte, die dem Besucher bei seiner Navigation von großer Hilfe sind, da sie seine Orientierung erleichtern. Markante Navigationspunkte sind die Home-Page (die Startseite), die Übersichtsseite zu einem inhaltlichen Bereich und bei größeren Dokumenten der jeweilige Seitenanfang. Bei hierarchischen Strukturen kann auch die jeweils übergeordnete Ebene bzw. deren Deckseite einen guten Platz zur Neuorientierung bieten. Um bei dem Benutzer das Gefühl des Verlorenseins zu vermeiden, sollten die erwähnten Punkte auf keinem Dokument der Web-Präsenz fehlen.

Außerdem ist es wichtig, dem Benutzer jederzeit die Möglichkeit zu offerieren, seine aktuelle Position innerhalb der Gesamtstruktur zu ersehen. An dieser Stelle kann das Layout, z.B. durch die Farbgestaltung einzelner inhaltlicher Bereiche, wertvolle Hilfestellungen geben. Bei geeignetem Einsatz erleichtert es dem Benutzer so, seine aktuelle Position innerhalb der Web-Präsenz-Struktur zu determinieren.

Neben der in Navigationsleisten an den Content angefügten Navigations-Struktur ist insbesondere die Verknüpfung der Contents untereinander ein wertvolles Element der Navigation innerhalb einer Web-Präsenz. Prinzipiell ist eine Web-Präsenz, deren einzelne Dokumente untereinander stark verknüpft sind, erstrebenswert. Bezüglich der Navigation innerhalb einer solchen Web-Präsenz entstehen jedoch für den Benutzer oft übermäßig viele Alternativen für das Fortschreiten von einer bestimmten Position in einem Dokument. Dies überfordert viele Benutzer in ihrer Entscheidungsfindung und führt, wenn sie mehreren dieser Links folgen, leicht zu Orientierungslosigkeit [Con87]. Weitere Möglichkeiten, den Benutzer zu den gewünschten Informationen zu führen, sind Schlagwort-Suchmasken und sogenannte „Site-Maps“. Bei einer guten Indexierung der über die Web-Präsenz abrufbaren Informationen kann

dem Benutzer durch die lokale Suchmaschine der direkte Zugang zu den gewünschten Dokumenten ermöglicht werden. Die Site-Map hingegen erlaubt es dem Benutzer, einen schnellen Überblick über die inhaltliche Struktur der Web-Präsenz zu gewinnen. Der heutzutage übliche große Umfang der auf einer Web-Präsenz verfügbaren Dokumente führt allerdings in vielen Fällen zu einer eher unübersichtlichen Site-Map. Aus diesem Grunde sind sie oft nur noch als Beiwerk für einen Gesamtüberblick über die Web-Präsenz im Einsatz und verzichten auf hohe Detailtiefen.

Schließlich gibt es für bestimmte Arten von Dokumenten, welche z.B. einen Workflow definieren, auch die Variante, die Navigation über vorgefertigte Pfade vorzugeben. Dem Benutzer werden in dieser Situation, z.B. wie in einem Tutorial, Start- und Endpunkte angeboten. Zwischen diesen Fixpunkten wird der Inhalt, z.B. die Unterrichtseinheit, vermittelt.

Da sich die Benutzer einer Web-Präsenz in ihren Präferenzen bezüglich der Navigationselemente stark voneinander unterscheiden, ist es für die Betreiber einer Web-Präsenz ratsam, möglichst viele verschiedene Elemente miteinander zu kombinieren. Trotz des Einhaltens einer klaren Navigationsstruktur können so die verschiedenartigste Benutzer zu den von ihnen gewünschten Informationen geleitet werden.

Mit einem WCMS müssen die hier besprochenen Navigationselemente für eine Web-Präsenz konzeptioniert und anschließend automatisiert umgesetzt werden. Die technische Ausgestaltung der Navigationselemente hat großen Einfluss auf die Handhabung der Web-Präsenz – sowohl seitens des Betreibers als auch seitens des Benutzers. Neben der Handhabung hängen aber auch Design-Aspekte von der technischen Ausgestaltung ab. Für sämtliche technischen Aspekte der Navigation lassen sich Vor- und Nachteile nennen, die bei der Konzeption einer Web-Präsenz gegeneinander abgewogen werden müssen. Im Folgenden ist eine Aufzählung der gängigen Ausprägungen von Navigationselementen gegeben.

Textuelle Navigationselemente Navigationselemente sind textuell, wenn der dahinterliegende Hyperlink durch einen Text beschrieben wird und dieser lediglich durch Einbringen der ASCII-Zeichen in die HTML-Seite entstanden ist. Dies ist im Unterschied zu einem grafischen Navigationselement, welches einen Text visualisiert, bei dem der Browser eine binäre Datei, z.B. eine Grafik, lädt, hinter der der Hyperlink liegt, zu sehen.

Textuelle Navigationselemente haben den Vorteil, dass sie leichter automatisiert erstellt werden können. Damit gehen aber auch starke Einschränkungen bezüglich der Gestaltungsfreiheiten der Navigationsleisten einher. Es besteht nur die Möglichkeit, im Rahmen normaler HTML-Formatierungen die Navigationspunkte zu gestalten. Die letztendliche Darstellung der Navigationselemente verbleibt bei dem Web-Browser. In Abhängigkeit von den ausgewählten Schriften und Schriftgraden des Benutzers kann sich bei ihm das Layout drastisch ändern.

Da textuelle Navigationselemente, wie der Name schon impliziert, auf Text beruhen, ist das zu übertragende Datenvolumen im Vergleich zu multimedialen Elementen relativ klein. Dies macht sich bei einer umfangreichen Navigationsstruktur durchaus positiv auf der Seite des Benutzers bemerkbar, insbesondere beim Erstkontakt, da kurze Ladezeiten die Attraktivität der Web-Präsenz steigern.

Multimediale Navigationselemente In ein Dokument eingebettete multimediale Elemente ziehen im Normalfall mehr Aufmerksamkeit auf sich als rein textuelle Bausteine. Weiterhin haben multimediale Elemente bei der Darstellung im Web-Browser den Vorteil, dass der Designer der Elemente die absolute Kontrolle über deren Aussehen hat. Web-Browser visualisieren die multimedialen Elemente, nehmen aber keinen Einfluss auf deren Darstellung. Auch Benutzereinstellungen bezüglich der Art der Darstellung von z.B. grafischen Elementen gibt es nicht. Der einzige Einfluss, der dem Benutzer auf multimediale Elemente verbleibt,

ist die globale Entscheidung, diese nicht vom Web-Browser darstellen zu lassen. Basiert eine Navigationsleiste auf grafischen Elementen, so muss sie deshalb zusätzlich mit einer textuellen Navigationsmöglichkeit versehen werden, da anderenfalls der Benutzer nicht innerhalb der Web-Präsenz navigieren kann. Dies bietet sich im Übrigen auf jeden Fall an, wenn behinderten Benutzern des WWW die Möglichkeit (vgl. Seite 67) verschafft werden soll, mit der Web-Präsenz zu arbeiten. Wie oben schon erwähnt, sind multimediale Elemente in der Navigation im Vergleich zu rein textuellen Elementen deutlich komplexer in der Erstellung und umfangreicher in dem Maße des zu übertragenden Datenvolumens.

Dynamische Navigationselemente Eine Sonderform von multimedialen Navigationselementen stellen die dynamischen Navigationselemente dar. In den meisten Fällen handelt es sich um aktive Elemente, die eine Funktion ausführen, sobald der Benutzer mit sie mit seinem Mauszeiger aktiviert. Je nach Ausprägung lassen sich solche dynamischen Komponenten einfach oder mit manueller Unterstützung automatisiert erstellen.

3.2.1.1.3 Layout

Das Layout oder Design einer Web-Präsenz stellt die Verpackung der zu übermittelnden Inhalte dar. Da für den ersten Eindruck von einer Web-Präsenz vorrangig deren Design entscheidet, ist dieses durch Fachleute zu erstellen. Neben störenden Faktoren wie der Ladezeit, die bei zu großen und zu vielen Grafiken entsteht, ist z.B. auch die Beachtung einer geeigneten Bildschirmauflösung von enormer Bedeutung. Außer diesen eher technischen Faktoren sind die gestalterischen Elemente zu berücksichtigen, welche vom persönlichen Geschmack des Betreibers der Web-Präsenz bestimmt sind. Einflussfaktoren für das Design sind die Inhalte, die in der Web-Präsenz vermittelt werden sollen.

Eine konzeptionelle Entscheidung stellt bei der Layoutfrage der Aufbau der Dokumente dar. Die Zusammenführung von Content, Navigation und grafischem Design kann sowohl in Form eines Frame-Sets geschehen als auch in der einer Tabellenstruktur. Beide Lösungen haben eine Reihe von Vor- und Nachteilen, die in Abhängigkeit vom Einsatzzweck entscheiden, welche Variante zu bevorzugen ist. Im Folgenden werden beide Lösungen besprochen.

Frame-Sets Bei einem Frame-Set werden im Browser innerhalb eines Fensters verschiedene Dokumente gleichzeitig angezeigt (siehe Abb. 3.4). Ein Hauptdokument beinhaltet eine Beschreibung, welche Dokumente dargestellt werden sollen und wie ihre Aufteilung vorgesehen ist. Sowohl vertikale als auch horizontale Aufteilung ist möglich. Weiterhin können Frame-Sets ineinander geschachtelt werden. Sie werden beispielsweise sinnvoll eingesetzt, um Navigationsleisten von den eigentlichen Dokumenten zu trennen. Dies hat den Vorteil, dass bei längeren Dokumenten die Navigationsmöglichkeit immer gegeben ist, da die Navigationsleiste(n) immer sichtbar bleiben. Weiterhin müssen diese nur dann neu vom Browser geladen werden, wenn sich die Navigation ändert; dies verringert die Ladezeiten deutlich.

Ein weiterer Vorteil von Frame-Sets ist die einfache Einbindung von externen Dokumenten. Ein Frame-Set kann jederzeit als innere Frames Dokumente von anderen Web-Präsenzen integrieren. Dies ist sinnvoll, wenn einzelne Dokumente, z.B. Nachrichten-Ticker, Börsenkurse etc., von Agenturen eingekauft werden.

Nachteile von Frame-Sets sind die Adressierung von Dokumenten und die schwierigere Navigation. Üblicherweise zeigt der Browser in der Adressleiste immer die URI des Frame-Set-Hauptdokuments an. Häufig ist dies die Adresse der Einstiegsseite, eher seltener die Adressen der Deckdokumente (siehe Seite 62). Dies macht es zum einen schwer, dem Benutzer URIs auf bestimmte Content-Seiten zu übermitteln, zum anderen kann der Benutzer keine sinnvollen Bookmarks setzen.

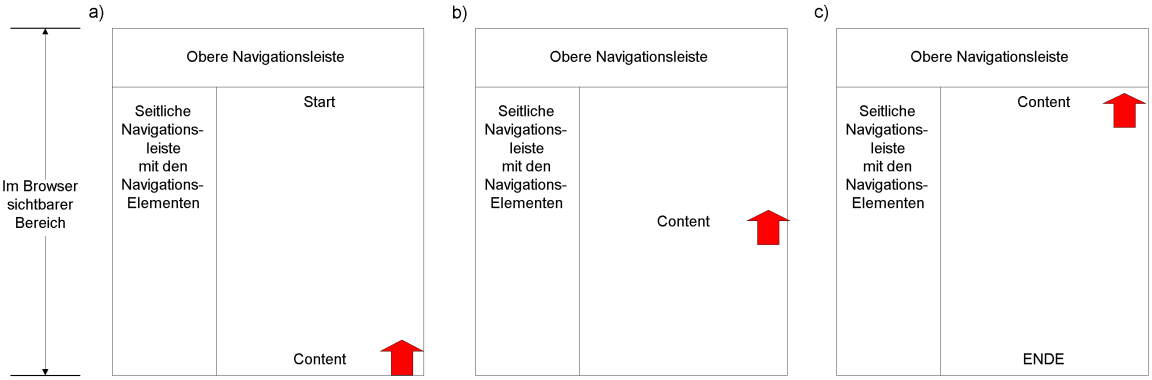

Abbildung 3.4: Layout in Form eines Frame-Sets.

Gleichzeitig erschweren es Frame-Sets dem Betreiber der Web-Präsenz, Links zwischen Content-Seiten zu erstellen, welche unterschiedliche Navigationsleisten erfordern. Das Problem ergibt sich wiederum durch die Adressierung der Frames untereinander, welche nur den Austausch innerer Dokumente ermöglicht.

Abhilfe für diese Probleme kann darin bestehen, für jedes Dokument ein eigenständiges Frame-Set vorzuhalten. Damit kommt der Vorteil der immer sichtbaren Navigation zum Tragen. Trotzdem verfügt jedes Dokument über eine eindeutige URI des Frame-Sets und ermöglicht so die gezielte Adressierung eines Dokumentes. Für jedes darzustellende Dokument müssen aber mindestens das Frame-Set-Hauptdokument und das Content-Dokument geladen werden. Dies ist der Ladezeit abträglich, da das Content-Dokument erst nach dem Hauptdokument geladen werden kann. Das Management solcher Frame-Sets ist bei größeren Web-Präsenzen nur über technische Hilfsmittel, die diese automatisch generieren, möglich.

Tabellenstrukturen Die andere Möglichkeit, Elemente einer Web-Präsenz zusammenzustellen, ist, sie in einer Tabelle zu platzieren (siehe Abb. 3.5). Je nach Umfang des eigentlichen Contents kann dies eine empfehlenswerte oder abzulehnende Lösung sein. Optimal ist eine Tabellenstruktur, wenn der Content genau in die Bildschirmseite passt und der Benutzer nicht den Bildschirminhalt verschieben (scrollen) muss, um das Ende des Dokumentes zu sehen. Ist der Content zu umfangreich, muss der Benutzer scrollen und verschiebt somit die Navigation aus dem sichtbaren Bereich, wodurch das Navigieren auf der Web-Präsenz verkompliziert wird. Abhilfe können bei langen Dokumenten in regelmäßigen Abständen angebrachte Rücksprungmarken schaffen, die es dem Benutzer ermöglichen, zu dem Kopf des Dokumentes mit der Navigationleiste zurückzuspringen. Die Einbindung externer Contents in die Tabellenstruktur kann nicht wie bei Frame-Sets über eine simple Verknüpfung geschehen; stattdessen müssen serverseitig diese Contents in die Tabelle eingebunden werden. Dies erfordert nicht unerheblichen Aufwand, kann aber hilfreich sein, wenn die zurückgelieferten Daten bezüglich ihres Layouts an die Web-Präsenz angepasst werden sollen.

3.2.1.2 Klassen von Dokumenten

Wird eine Web-Präsenz hierarchisch strukturiert, so lassen sich hinsichtlich ihrer Funktion verschiedene Typen von Dokumenten klassifizieren. Bei Betrachtung gängiger Web-Präsenzen können drei verschiedene Arten von Dokumenten ausgemacht werden (siehe Abb. 3.6):

- Startseite (Homepage)

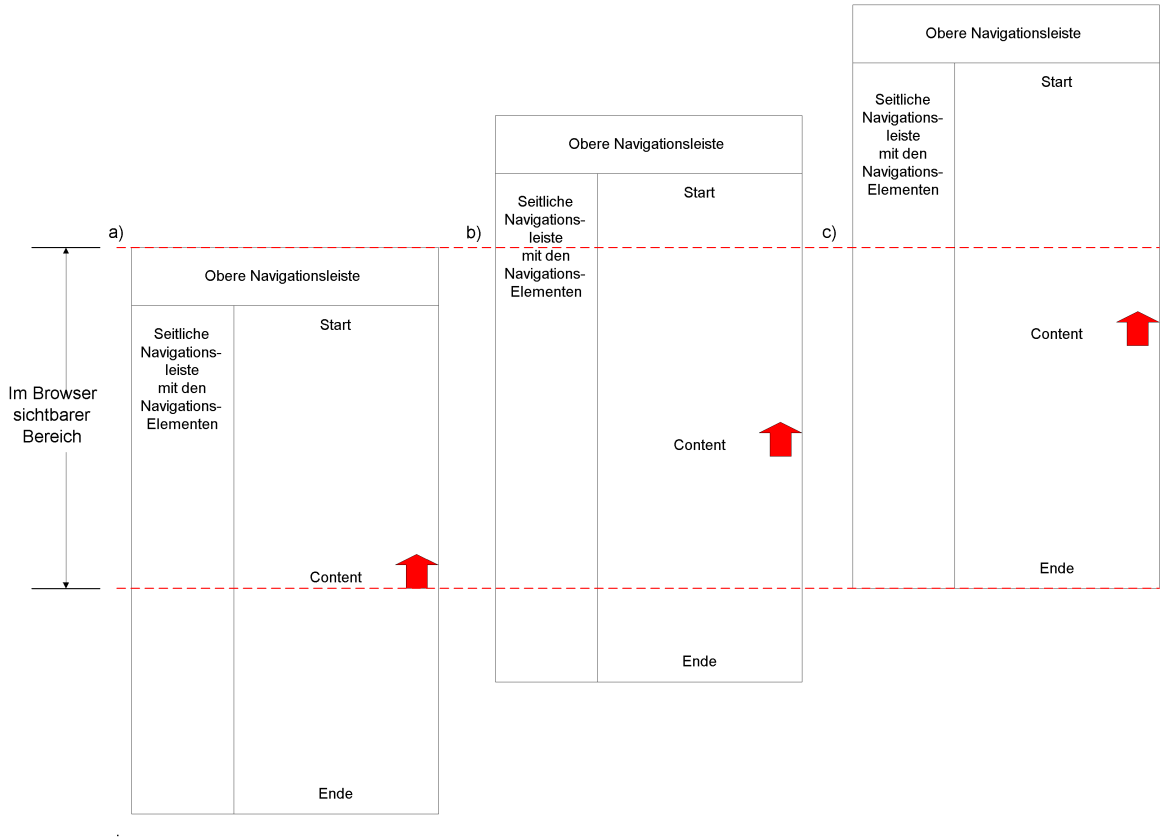

Abbildung 3.5: Layout in Form einer Tabelle.

- Übersichtsseite (Deckseite)
- Content-Seite

Es gibt eine Startseite (Homepage), eine vergleichsweise kleine Zahl von Übersichtsseiten sowie eine große Zahl von Content-Seiten. Entsprechend ihrer Funktion unterscheidet sich oft das Design der verschiedenen Typen. Es hängt aber stark vom individuellen Design der Web-Präsenz ab, ob die Startseiten, die Deckseiten und schließlich darunter eingeordnet die Content-Seiten ihre Funktionalität durch ein eigenständiges Layout unterstützen.

3.2.1.2.1 Startseite

Das „Startseite“ oder auch „Homepage“ genannte Dokument ist das dem Besucher präsentierte Dokument, wenn er als URI die Adresse der Web-Präsenz eingegeben hat. In den meisten Fällen entscheidet dieses Dokument, ob der Besucher eine Web-Präsenz „betritt“ oder direkt wieder verlässt. Dementsprechend hoch sind die Anforderungen an diese Startseite. Während zu Anfang die Startseite einer Web-Präsenz häufig nur als eine Art Einband (um in der Sprache der Printmedien zu sprechen) eingesetzt wurde und dementsprechend das Analogon zu einem Titel mit einem hübschen Hintergrund darstellte, steht bei heutigen Web-Präsenzen ihre Funktionalität im Vordergrund. Um den Besucher in möglichst wenigen Schritten zu seinem Ziel, zu führen wird bereits die Startseite mit einer großen Zahl weiterführender Verweise versehen. Daher unterscheidet sich heute eine Homepage nur noch geringfügig von den im Folgenden erläuterten Übersichtsseiten.

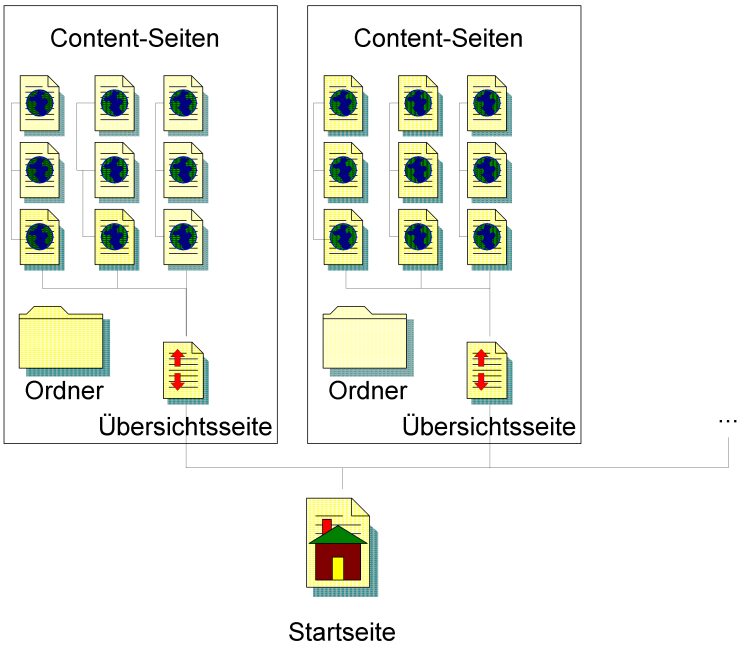

Abbildung 3.6: Eine einfache hierarchische Struktur einer Web-Präsenz mit den Dokumentklassen Startseite, Übersichtsseiten und Content-Seiten.

3.2.1.2.2 Übersichts-/Deckseiten

Ziel bei dem Aufbau von Web-Präsenzen ist es, diese inhaltlich zu strukturieren. Bei Verwendung hierarchischer Strukturen werden die einzelnen Knoten, d.h. Bereiche, mit Einstiegsseiten versehen. Diese auch als Übersichts- oder Deckseiten bezeichneten Dokumente geben eine inhaltliche Zusammenfassung und liefern Übersichten von Verweisen zu tiefer in der Struktur eingeordneten Dokumenten. Übersichtsseiten stellen in erster Linie Strukturelemente dar, mit denen die Besucher der Web-Präsenz über die weiteren Navigationsmöglichkeiten informiert werden. Anhand der Übersichtsseiten können sich die Benutzer zu den inhaltlich tiefer gelagerten Informationen vortasten.

3.2.1.2.3 Content-Seiten

Die Dokumente, die die eigentlichen Inhalte, Informationen, enthalten, die über die Web-Präsenz dem Publikum zugänglich gemacht werden sollen, werden als Content-Seiten bezeichnet. In diesen werden die oben (siehe Seite 56) beschriebenen Medien- und Applikationsobjekte dem Benutzer der Web-Präsenz präsentiert. Content-Seiten werden über die Übersichtsseiten in die Struktur der Web-Präsenz eingebettet. Dabei können sie sowohl als einzelne Dokumente für sich stehen oder untereinander inhaltlich verknüpft sein. Werden die Content-Seiten, z.B. im Fall der serverseitigen Applikationsobjekte über dynamische Prozesse generiert, können sie auch Workflows, beispielsweise für die Entgegennahme von Benutzereingaben, aufbauen.

3.2.1.3 Multilingualität

Das Internet ist ein globales Medium. Um die sich daraus ergebenden Chancen für einen internationalen Markt voll ausschöpfen zu können, reicht es nicht, wenn die Web-Präsenz nur über Dokumente in einer einzigen, sei es auch eine im Internet stark verbreitete Sprache

wie Englisch, verfügt. Der für den Aufbau und die Pflege einer mehrsprachigen Web-Präsenz benötigte Aufwand ist um ein Vielfaches höher als bei einer Web-Präsenz, welche sich nur monolingual präsentiert (vgl. dazu Pradka in [Pra01]). Wie in Heuer et al. [HHR+99b] ausgeführt, können bei der Konzeption einer mehrsprachigen Web-Präsenz Symmetrien (vgl. Abb. 3.8 und Abb. 3.7) zwischen den mehrsprachigen Versionen ausgenutzt werden. Für den Informationsarchitekten stellt sich somit zuerst die Frage nach der Symmetrie der Web-Präsenz in den Sprachversionen: „Muss es zu jedem Dokument *D(A)* in der Sprache *A* auch ein Dokument *D(B)* in der Sprache *B* geben, welches über den gleichen Inhalt mit dem gleichen Aktualitätsstand sowie eine analoge Verlinkung verfügt?“ Von der Beantwortung dieser Frage

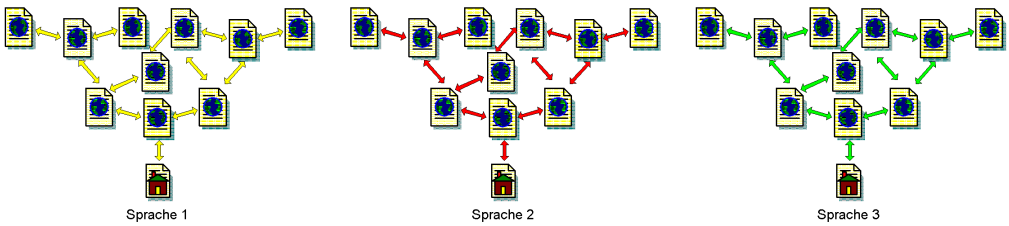

Abbildung 3.7: Eine Web-Präsenz, bei der die Sprachversionen symmetrisch aufgebaut sind.

hängt ab, wie eine automatische Verlinkung der unterschiedlichen Sprachversionen untereinander realisiert wird. Wünschenswert ist es, von jedem Dokument jederzeit in eine andere sprachliche Version des gleichen Dokumentes wechseln zu können, sofern diese vorhanden ist. Für die automatisierte Erstellung solcher Verknüpfungen muss die Sprachversionsbeziehung zwischen den Dokumenten der Web-Präsenz bekannt sein. Verwaltet das System die Sprachbeziehung *SB*: „Dokument *D(en)* ist die englischsprachige Version des Dokumentes *D* in deutscher Sprache *D(de)*“, kann automatisiert über die Regel *R (de, en)* bzw. *R (en, de)* eine Verlinkung zwischen *D(de)* und *D(en)* bzw. *D(en)* und *D(de)* erstellt werden. Solche Beziehungen zwischen Dokumenten können in der Regel nur teilautomatisiert, z.B. über den Mechanismus „englischsprachige Kopie des Dokumentes anlegen und bearbeiten“, oder über Benutzereingaben realisiert werden. Aufwendig werden die Algorithmen für die Verwaltung mehrsprachiger Web-Präsenzen mit sprachübergreifender Verlinkung, wenn keine Symmetrie in den Sprachversionen vorhanden ist. Auf die manuelle Definition von Beziehungen zwischen den Dokumenten kann für den normalen Betrieb verzichtet werden, wenn die Symmetrieeigenschafen der Sprachversionen für die Automatisierung genutzt werden. Aus der Symmetrie lassen sich Regeln für die Verknüpfung zwischen verschieden sprachigen Dokumenten ableiten. Diese können z.B. an der Funktion der Dokumente festgemacht sein: Die Eigenschaft „Deckdokument“ in verschiedenen Sprachen eines Verzeichnisses zu sein, kann genutzt werden, um eine sprachübergreifende Verknüpfung zwischen diesen Dokumenten zu realisieren. Die sprachliche Symmetrie einer Web-Präsenz zu gewährleisten, erfordert einen quantitativ hohen Aufwand bei der Pflege der Dokumente. Andererseits erleichtert die Symmetrie aber auch die Pflege, da für jedes Dokument die Verknüpfungen mit anderen Dokumenten für die verschiedenen Sprachversionen mit einer einfachen Abbildungsvorschrift übernommen werden können. Bei einer verzeichnisbasierten Verwaltung von verschiedenen Sprachversionen gibt es zwei gängige Verfahren (Abb. 3.9): im ersten Verfahren werden die unterschiedlichen Sprachversionen direkt auf oberster Ebene unterschieden. Die darunterliegende Verzeichnisstruktur wird dann für jede Sprachversion separat aufgebaut.

Beim zweiten Verfahren wird eine gemeinsame Verzeichnisstruktur für alle Sprachversionen erstellt; in jedem Verzeichnis werden die unterschiedlichen Sprachversionen in entsprechenden Unterverzeichnissen zusammengefasst. Um Dokumente innerhalb der Web-Präsenz

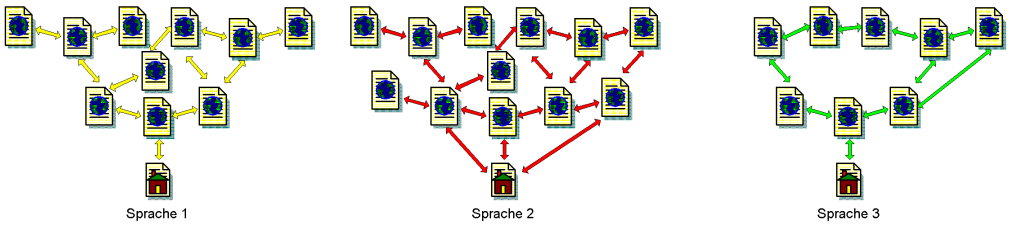

Abbildung 3.8: Eine Web-Präsenz, bei der die Sprachversionen asymmetrisch aufgebaut sind.

einfach in ihrer Sprache unterscheiden zu können, kann die Sprache in den Namen kodiert werden. Ein übliches Verfahren ist hier, vor die den Datei-Typ bestimmende Endung, z.B. „.html“ noch die Sprache, getrennt durch einen Punkt oder Unterstrich, zu setzen. Beispielsweise könnte ein HTML-Dokument, welches in deutscher Sprache verfasst ist, mit dem Namen „dokumentname.de.html“ versehen werden. Diese Art der Namenskonvention ist weit verbreitet und wird, z.B. bei der Content-Negotiation [HM98], d.h. bei der Aushandlung der bevorzugten Darstellungsformate und -sprachen, zwischen Browser und Web-Server angewandt. Der Nachteil des zweiten Verfahrens liegt darin, dass sich die Betreiber der Web-Präsenz auf

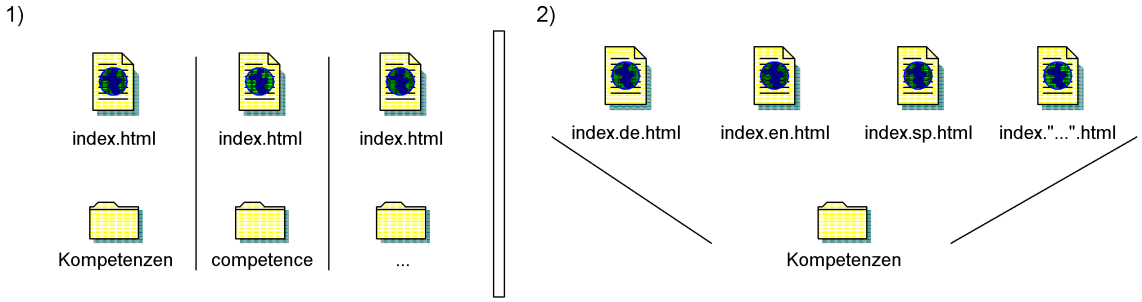

Abbildung 3.9: Organisation der Dokumente einer mehrsprachigen Web-Präsenz durch 1) getrennte Verzeichnisse und 2) Dateinamen.

eine „Master-Sprache“ einigen müssen, in der Verzeichnisse benannt werden.

3.2.2 Lebenszyklus

Der Entwicklung eines Systems zur Web-Präsenz-Verwaltung liegt eine Annahme über den Lebenszyklus zugrunde. Der in dieser Arbeit angenommene Lebenszyklus orientiert sich an den von Takahashi und Liang [TL97b] bzw. Atenzi et al. [AMM+98] entworfenen und integriert die im Praxiseinsatz erworbenen Erfahrungen mit dem sich an die Evolution anschließenden Relaunch der Web-Präsenz. Wie der Name „Zyklus“ schon andeutet, hat sich in der Praxis gezeigt, dass eine Web-Präsenz „nie fertig“ ist. Zum einen wächst die Web-Präsenz unter einer kontinuierlichen Modifikation der Inhalte, zum anderen werden in der schnelllebigen Internet-Welt ständig neue Funktionalitäten notwendig, wenn die Web-Präsenz dem aktuellen Stand der Technik entsprechen soll. Ein praxisnaher Lebenszyklus einer Unternehmens-Web-Präsenz, lässt sich in sechs Phasen aufteilen (Abb. 3.10).

- *Phase 1* Am Anfang einer Web-Präsenz steht die Idee des Unternehmens, sich im Internet bzw. dem WWW zu positionieren. In dieser Phase wird das Projekt „Web-Präsenz“

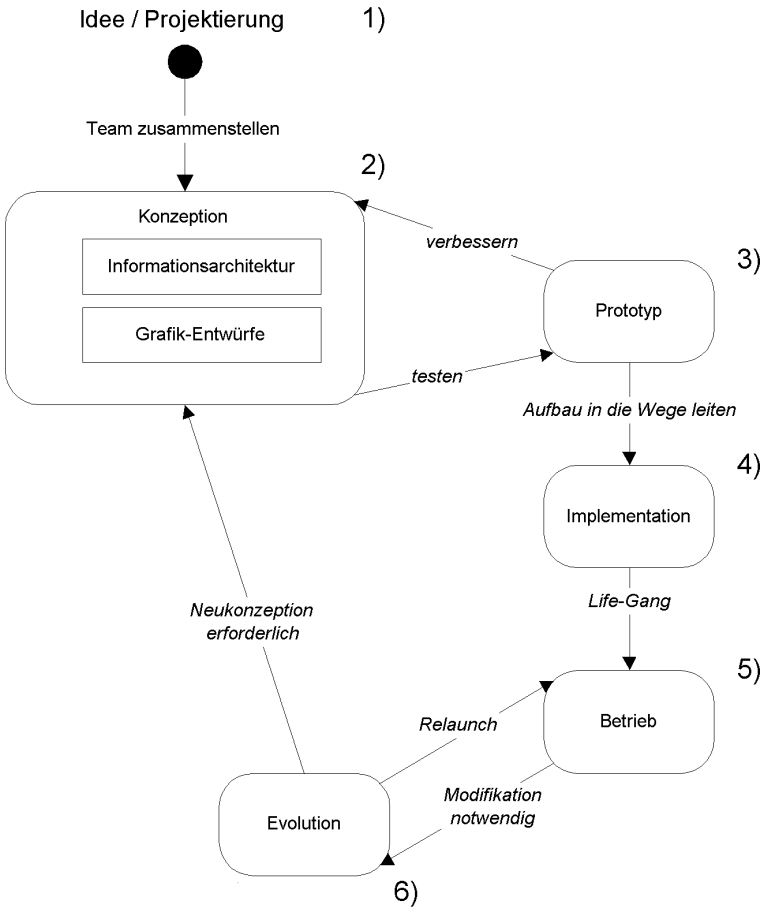

Abbildung 3.10: Der Lebenszyklus einer Web-Präsenz aufgeteilt in sechs Phasen.

initialisiert. Ein Budget wird bereitgestellt und ein Team zusammengesetzt, wie dies einleitend in Abschnitt 2.2.1 auf Seite 27 bei der Problemstellung beschrieben wurde.

- *Phase 2* Das Projektteam hat die Aufgabe, die Web-Präsenz zu konzeptionieren. In dieser Phase wird eine Anforderungsliste erarbeitet, anhand derer die Informationsarchitektur entwickelt werden kann, die die Grundlage der Web-Präsenz dar. Ihre Entwicklung wird anschaulich von Lin et al. [LNHL00] sowie Newman und Landay [NL00] beschrieben. Kennzeichnend für diese Phase ist die hohe Interdisziplinarität: Kompetenz aus Fachabteilungen, Marketing, Grafik & Design und Technik werden zusammengebracht, um eine tragfähige Architektur zu entwerfen, die den individuellen Anforderungen gerecht wird.

- *Phase 3* Eng verbunden mit dem Aufbau der Informationsarchitektur ist die Entwicklung von Prototypen, die die konzeptionellen Vorgaben umsetzen. Der Test der Architektur und der Designvorschläge anhand der Prototypen fließt solange in die Konzeption ein bis ein zufriedenstellendes Ergebnis erzielt wird.

- *Phase 4* Basierend auf diesem Prototypen wird die Web-Präsenz implementiert. Die Architektur manifestiert sich sowohl strukturell und inhaltlich als auch technisch. Abgeschlossen wird diese Phase durch den Life-Gang (Life-Going) der Web-Präsenz. Sie

wird im WWW öffentlich zugreifbar gemacht und erste Besucher von außerhalb des Unternehmens können damit arbeiten.

- *Phase 5* Die Phase, auf die die Projektierung der Web-Präsenz abzielt, ist ihr Betrieb. Die Web-Präsenz ist gemäß den Zielen, die für das Unternehmen mit der Web-Präsenz verbunden sind (vgl. Seite 1.1.1), produktiv. In dieser Phase muss sich das eingesetzte WCMS bewähren. Dokumente ändern sich, werden ersetzt, entfernt und/oder archiviert.

- *Phase 6* Mit dem Betrieb der Web-Präsenz erwachsen neue Anforderungen. Diese werden in einer Evolutions-Phase aufgefangen; so kann beispielsweise die Einbindung neuer Datenquellen erfolgen, eine Umstrukturierung oder ein neues Design notwendig werden. Sofern sich die Modifikationen im Rahmen halten, können sie mit der bestehenden Architektur umgesetzt werden und mit einem Relaunch in den Betrieb überführt werden. Ist mit den bestehenden Konzepten eine Anpassung der Web-Präsenz an die aktuellen Bedürfnisse nicht mehr möglich, wird eine Neukonzeption erforderlich; der Kreis schließt sich.

3.2.3 Kriterien für eine „gute“ Web-Präsenz

Wie schon in den Zielen für das Elektronische Publizieren erwähnt (siehe Seite 23), ermöglicht das Internet einem Unternehmen, die sich von dem Printmedium stark unterscheidende Chance zur Repräsentation und darüber hinaus auch die Möglichkeit der direkten Interaktion mit dem Besucher (Kunden). Damit der Internet-Auftritt eines Unternehmens erfolgreich verläuft, sollte die zu diesem Zweck entwickelte Web-Präsenz die im Folgenden genannten Kriterien (Abb. 3.11) erfüllen.

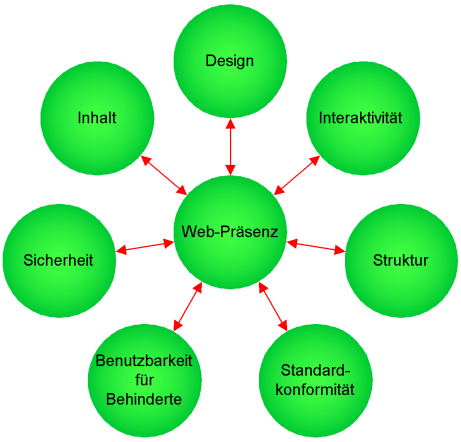

Abbildung 3.11: Meßkriterien für eine Web-Präsenz.

3.2.3.1 Design

Das Design sollte ansprechend sein und sich trotzdem durch eine hohe Funktionalität auszeichnen. Weiterhin sollte es berücksichtigen, dass die im Internet allgemein verfügbare Bandbreite beschränkt ist. Deshalb müssen kurze Ladezeiten (dies ist gleichbedeutend mit dem sparsamen Einsatz von multimedialen Elementen) weit oben auf der Kriterienliste rangieren.

3.2.3.2 Inhalt

Das an Bedeutung nicht zu übertreffende Kriterium ist der Inhalt der Web-Präsenz. Neben dem quantitativen Aspekt sollte vor allem Wert auf Qualität und Aktualität gelegt werden. Eine webgerechte Aufteilung in kleine bildschirmgerechte Dokumente ist ebenso wichtig wie eine hinreichende Verknüpfung der Inhalte durch sinnvolle Hyperlinks. Über einen Index und eine Übersicht sollten alle Dokumente problemlos auffindbar sein.

3.2.3.3 Struktur

Die Struktur der Web-Präsenz sollte ausgewogen sein, d.h. nicht zu stark in eine hierarchische Tiefe, aber auch nicht zu weit in die Breite wachsen. Entscheidend ist die schnelle Auffindbarkeit von Dokumenten; diese ist dann gegeben, wenn die Struktur der Web-Präsenz für Besucher einfach nachzuvollziehen ist. Aufgabe eines Informationsarchitekten ist es, eine für die angebotenen Inhalte angemessene Struktur zu entwerfen.

3.2.3.4 Interaktivität

Eine moderne Web-Präsenz bemüht sich, dem Besucher den von seinem Arbeitsplatz gewohnten Komfort zu bieten. Aus diesem Grund werden viele interaktive Komponenten bereitgestellt. Neben überhöhten Ladezeiten stellt aber insbesondere die Anpassung auf die verschiedenen Browser-Typen ein nahezu unüberwindbares Hindernis dar, wenn Java, Java-Script, Plug-ins oder Style-Sheets zum Einsatz kommen sollen. Eine gute Web-Präsenz orientiert sich an dem technischen Umfeld ihrer Anwender. Dazu gehört neben der Internet-Anbindung (Bandbreite) auch ein Überblick über die eingesetzte Hard- und Software. Basierend auf diesen Vorgaben sollte eine einfach zu bedienende, fehlertolerante Technik ausgewählt werden.

3.2.3.5 Behindertengerechte Darstellung

Ein nicht zu unterschätzender Teil der Internet-Gemeinde ist physisch behindert. Damit die Web-Präsenz auch von dieser Gruppe problemlos bedient werden kann, müssen diverse Kriterien bei der Konzeption der Web-Präsenz berücksichtigt werden. Hierzu gehört vor allem eine rein textbasierte Verständlichkeit der Präsenz (Vorlesemodule, sprachgesteuerte Navigation). Allein durch textuelle Navigationspunkte, durch Beschreibung der eingesetzten Grafiken etc. sollte die Web-Präsenz erschlossen werden können. Ein guter Überblick über weitere Möglichkeiten kann anhand der Recommendations des W3C [CVJ99] gewonnen werden.

3.2.3.6 Standardkonformität

Ziel bei dem Aufbau einer Web-Präsenz muss es sein, die auf ihr verfügbaren Dokumente standardkonform aufzubauen. Dazu sind die von den entsprechenden Gremien des WWW (www.w3c.org) definierten Kriterien zu berücksichtigen. Neben der Strukturierung der Dokumente betrifft dies insbesondere die Aufbereitung von Metadaten (siehe Seite 116).

3.2.3.7 Sicherheit

Eine gute Web-Präsenz verlangt nach diversen Sicherheitsvorkehrungen. Dies bedeutet im Normalfall vor allem, dass peinliche „Hacks“ der Web-Präsenz mit allen Mitteln verhindert werden. Dazu gehört selbstverständlich auch, dass über den Web-Server nicht unberechtigt Dokumente abgerufen oder Programme ausgeführt werden können. Sofern die Web-Präsenz transaktionell ausgerichtet ist, müssen die Transaktionsdaten (Personendaten etc.) unbedingt mit in das Sicherheitskonzept einbezogen werden.

3.2.4 Statische und dynamische Web-Präsenzen

Während in der Anfangszeit des WWW typische Web-Präsenzen aus vergleichsweise simplen HTML-Seiten in einem Dateisystem bestanden, die mittels eines Web-Servers an die Browser ausgeliefert werden konnten, hat sich mit dem Fortschreiten der Technik ein Trend zu hochdynamischen Dokumenten ausgebildet. Nach einer kurzen Beleuchtung der Vor- und Nachteile statischer Web-Präsenzen wird im folgenden Abschnitt auf die Möglichkeiten, eine Web-Präsenz zu dynamisieren, eingegangen.

3.2.4.1 Statische Web-Präsenzen

Bei einer statischen Web-Präsenz liegen die Dokumente, die von dem Web-Server ausgeliefert, werden als statische Dateien in dessen Dateisystem. Die Aufgabe des Web-Servers ist darauf beschränkt, die angefragten Dateien an die Browser zu übermitteln; aus Performance-Gründen ist diese Lösung sehr empfehlenswert. Da außer dem Web-Server keine zusätzlichen Software-Komponenten benötigt werden, ist diese Lösung einfach. Weiterhin können weitere Sicherheitsrisiken, wie sie durch ausführbaren Programmcode immer bestehen, vermieden werden.

Nachteilig bei der statischen Lösung sind beispielsweise Design-Änderungen, da alle Dokumente komplett neu geschrieben werden müssen, um eine Modifikation aller Dokumente zu erreichen. Das Einblenden von aktuellen Informationen in alle Dokumente ist genauso wenig möglich wie ihre Personalisierung. Unabhängig von der serverseitig implizierten Dynamik gibt es auch die clientseitige Dynamik (siehe Seite 68), deren Elemente in eine an sich statische Web-Präsenz integriert werden können. Aus Sicht des WCM gibt es für statische Web-Präsenzen aus Konsistenzgründen nur die Möglichkeit des „Staging-Exports“ (vgl. Seite 143); von einem solchen wird gesprochen, wenn eine Web-Präsenz im Rahmen des Publikationsprozesses vollständig neu aus dem internen Dokumentenbestand generiert wird. Ohne eine solche vollständige Neugenerierung kommt es ansonsten schnell zu Inkonsistenzen im Dokumentenbestand (vgl. Seite 30).

3.2.4.2 Dynamische Web-Präsenzen

Dynamische Web-Präsenzen sind flexibler als statische Web-Präsenzen; sie ermöglichen das Einblenden dynamischer Informationen, z.B. Nachrichten-Ticker, Aktienkurse etc. in die ausgelieferten Dokumente. Layout-Änderungen sind direkt auf alle Dokumente zu übertragen. Bei geeigneter Software ist die Personalisierung der Dokumente möglich, dies kann neben persönlichen Layout-Präferenzen bis zu individuell determinierten Inhalten gehen. Gute Beispiele für diese Richtung sind die Web-Mail-Dienste oder die Online-Banken.

Nachteilig an dynamischen Web-Präsenzen ist jedoch deren enormer Performance-Bedarf. Da die ausgelieferten Dokumente erst bei der Anfrage zusammengebaut werden und dafür unter Umständen auf eine Vielzahl von Datenquellen zugegriffen werden muss, konsumiert dieser Vorgang deutlich mehr Rechenleistung als dies bei statischen Dokumenten der Fall ist. Technische Möglichkeiten, eine Web-Präsenz zu dynamisieren, wurden bei der Besprechung von Web-Servern auf Seite 39 vorgestellt.

3.2.4.3 Clientseitige Dynamik

Der Vollständigkeit halber sei an dieser Stelle auch auf die clientseitige Dynamik eingegangen. Unabhängig von dem Web-Server können in die Dokumente Elemente eingebettet werden, die dem Benutzer der Web-Präsenz Interaktionsmöglichkeiten bieten bzw. auf clientseitige

Anforderungen reagieren. Die im Folgenden genannten Elemente stellen lediglich einen kleinen Ausschnitt dar, geben aber einen Überblick über die vorhandenen Möglichkeiten. Allen gemeinsam ist die Sicherheitsproblematik, die mit ihrer Anwendung einhergeht. Da es sich um ausführbaren Code handelt, muss der Benutzer der Web-Präsenz immer darauf vertrauen, dass der Code keine unerwünschten Aktionen, z.B. Löschen der Festplatte, ausführt. Die Web-Browser geben dem Benutzer daher die Möglichkeit, alle diese aktiven Komponenten nicht auszuführen. Ist eine Web-Präsenz auf solche interaktive Komponenten, z.B. für die Navigation, angewiesen, so kann ein sicherheitsbewußter Benutzer diese Web-Präsenz nicht nutzen. Aus diesem Grund sollten aktive Komponenten sparsam, und dann auch nur an nicht kritischen Stellen (ausgenommen Webapplikationen), eingesetzt werden. Für eine kurze Übersicht gängiger Techniken (vgl. Seite 38) sind die folgenden vier Vorgehensweisen relevant:

- *Java-Script*: Die aktuell wohl am weitläufigsten eingesetzten interaktiven Elemente bestehen aus Java-Script-Code (vgl. [Fla97]). Mit diesem lassen sich multimediale Effekte wie z.B. das Austauschen eine Bildes, wenn der Mauszeiger sich darüber bewegt, genauso realisieren wie kleinere Anwendungen.

- *Dynamic HTML*: Dynamic HTML (DHTML) ist eine Marketing-Bezeichnung für anpassungsfähige Web-Seiten. Realisiert wird die Anpassungsfähigkeit durch Java-Script (*Netscape*). *Microsoft* übernahm diese Technik als JScript. Parallel dazu bietet *Microsoft* im *Internet Explorer* eine zweite, Visual-Basic-basierte, Makrosprache mit dem Namen VBScript an. Als Oberbegriff für die beiden Sprachen (JScript und VBScript) hat Microsoft den Begriff „Active Scripting“ eingeführt. Besonderen Reiz für die Entwickler von Web-Seiten bringt die in Dynamic HTML eingeführte Layer-Technik. Layer teilen eine Web-Seite in verschiedene übereinanderliegende Ebenen ein; diese können einzeln ein- und ausgeblendet werden. Damit ergeben sich für den Entwickler vielfältige Designmöglichkeiten, die durch kleinere Elemente mit Anwendungscharakter (z.B. ein einfacher „Taschenrechner“) ergänzt werden.

- *Java-Applets*: Eine weitere, mehr für Anwendungen genutzte Möglichkeit, Interaktivität auf dem Client bereitzustellen, sind die Java-Applets. Als kleine Applikationen sind sie speziell für die Ausführung auf den Client-Maschinen konzipiert. In Frühzeiten wurden sie häufig eingesetzt, um Animationen auf Web-Präsenzen bereitzustellen und Mehrwert-Funktionen, z.B. Navigationsunterstützung, zu gewähren.

- *Plug-ins*: Eine flexible Möglichkeit, eine Web-Präsenz auf der Client-Seite mit Interaktivität zu versehen sind, die Plug-ins – ein Weg, den auch *Microsoft* mit einem Plug-in für ActiveX-Komponenten beschritten hat, um eine große Browser-Abdeckung realisieren zu können. Ein anderes sehr gängiges Plug-in stammt von *Makromedia*. Mit dem zugehörigen Datenformat „Flash“ lassen sich dynamische multimediale Anwendungen für die Client-Seite realisieren. Über Plug-ins können Web-Browser die für verteilte Architekturen notwendigen Komponenten aufnehmen. DCOM, CORBA oder RMI können dann für die Kommunikation zwischen den auf Server- und Client-Seite verteilten Objekten herangezogen werden.

3.2.5 Sicherheit

Eine Web-Präsenz stellt für das sie betreibende Unternehmen das Tor zur großen weiten Internet-Welt dar – Daten fließen durch dieses Tor in beide Richtungen. Nicht alle Daten, die durch dieses Tor kommen, sind von den Betreiber der Web-Präsenz erwünscht. Da die Web-Präsenz in direktem Kontakt mit dem Internet steht, ist sie sehr exponiert und wird leicht

das Ziel von Angriffen. Je nach Art dieser Attacken bleiben dem Anbieter verschiedene Mittel und Wege zur Abwehr. Neben der selbstverständlichen Sicherung des Systems beim Aufsetzen sowie dem regelmäßigen Einspielen von Software-Updates ist die Web-Präsenz beim normalen Betrieb zu überwachen – sämtliche Serverfunktionen müssen regelmäßig überprüft werden, um bei Problemen so schnell wie möglich reagieren zu können [Ger01].

3.2.5.1 Mögliche Angriffe

Betreiber einer Web-Präsenz müssen mit einer Reihe von verschiedenartigen Angriffen rechnen, deren Opfer die Web-Präsenz werden kann. Der Schaden kann dabei von der Nicht-Verfügbarkeit der Web-Präsenz im Internet über die Modifikation der publizierten Daten bis hin zum Datendiebstahl variieren.

3.2.5.1.1 Denial of Service (DoS)

In letzter Zeit, durch die prominenten Opfer sehr deutlich gemacht, wurde die Gefahr, die von den Denial of Service-Attacken ausgehen. Bei dieser Art von Angriff werden die Web-Server über eine längere Zeit mit unsinnigen Anfragen von einer größeren, im Internet verteilten Anzahl von Rechnern ausgelastet. Dies hat zur Folge, dass der Web-Server für die eigentlich sinnvollen Anfragen von Kunden keine Zeit mehr findet und somit seinen Dienst nicht mehr anbieten kann. Denial of Service-Attacken sind für die Betreiber deshalb sehr unangenehm, weil es ein sehr komplexer Vorgang ist, den Angriff abzuwehren. Je nachdem, wie gut die Attacke ausgeführt wird, kann es sehr schwer werden, die erwünschten von den unerwünschten Anfragen an den Web-Server zu unterscheiden. Da diese Art von Angriffen im Normalfall von einer großen Anzahl von Rechnern verteilt (Distributed Denial of Service, DDoS) ausgeführt wird, kommt erschwerend hinzu, dass es nicht ausreicht, bestimmte Netzwerk-Kanäle zu blockieren. Die Betreiber der angreifenden Rechner, welche normalerweise nicht mit den Angreifern identisch sind, müssen identifiziert und auf die Angriffe hingewiesen werden, die ihre Rechner ausführen. Um Denial of Service-Attacken schnell zu bemerken, ist ein ständiges Monitoring der Web-Server-Leistung unerlässlich.

3.2.5.1.2 Hack der Web-Präsenz

Eine Web-Präsenz wird als „gehackt“ bezeichnet, wenn die von dem Web-Server weitergereichten Dokumente nicht mehr die sind, die der Web-Präsenz-Betreiber vorgesehen hatte, sondern von Dritten eingestellte Dokumente. Dies setzt voraus, dass sich Unbefugte auf dem Web-Server Zugang zu dem Verzeichnis verschafft haben, in dem die Dokumente abgelegt sind. Möglich wird dies z.B. wenn die Zugangskennung zum Web-Server „abgehört“ oder „durch Ausprobieren“ gefunden wurde. Da die modifizierten Dokumente relativ schnell wieder durch die Originale ersetzt werden können, ist hier der größte Schaden der Image-Verlust des Web-Präsenz-Betreibers. Auch in diesem Fall hilft außer der vorbeugenden Sicherheit nur die regelmäßige Überprüfung der Dokumente. Dies kann z.B. ein Vergleich der Verzeichnisinhalte mit einer Referenz ermöglichen. Etwas rabiater, deshalb aber nicht weniger effektiv, ist das regelmäßige Überschreiben der Daten mit einer Referenzkopie aus einem anderen Rechner. Gegebenenfalls werden so Modifikationen an der Web-Präsenz einfach wieder durch die ursprünglichen Daten ersetzt.

3.2.5.1.3 Eindringen in das interne Netz

Je nachdem, wie gut die Sicherheitsarchitektur des Web-Präsenz-Betreibers ist und wie gut die einzelnen Komponenten konfiguriert sind, kann es passieren, dass der Web-Server das Tor

in das interne Netz des Betreibers darstellt. Diese Gefahr besteht insbesondere dann, wenn auf dem Web-Server arbeitende Anwendungen regelmäßig mit Komponenten im internen Netz (z.B. Datenbanken) kommunizieren müssen.

3.2.5.1.4 Datendiebstahl

Viele Web-Anwendungen arbeiten heutzutage mit sensiblen Daten. Insbesondere Online-Banking, Online-Shops etc. stellen bevorzugte Angriffsziele dar, wenn es um den Diebstahl von Daten geht. Abgesehen davon, dass bei Bekanntwerden eines solchen Vorfalls das Unternehmen einen immensen Image-Schaden erleidet, können die gestohlenen Daten missbräuchlich eingesetzt werden. Mit den in einem Online-Shop gespeicherten Daten, dem Namen der Kunden in Verbindung mit ihrer Kreditkartennummer sowie deren Verfallsdatum, können ohne weiteres Einkäufe bei anderen Online-Shops getätigt werden. Die elektronische Verfügbarkeit der Daten und die dabei u.U. sehr große Anzahl von Datensätzen potenziert den entstehenden Schaden um ein Vielfaches.

3.2.5.2 Systemanforderungen

Für die Hardware, die für den Betrieb eines Web-Servers in Frage kommt, gibt es viele verschiedene Anforderungen. Aus Sicherheitsüberlegungen werden dabei andere Aspekte favorisiert als aus Performance- oder Ausfallsicherheitsüberlegungen. Einen Ansatz zum Aufbau sicherer Web-Server stellt das Minimalismus-Prinzip dar. Auf dem Rechner wird ausschließlich diejenige Software eingesetzt, die zum Betrieb des Servers unbedingt notwendig ist. Sämtliche Hilfsprogramme, welche potentielle Gefahrenstellen implizieren, werden vom Rechner entfernt. Von dem Betriebssystem wird eine Minimalkonfiguration, günstigstenfalls performance-optimiert, eingesetzt. Die Hardware des Rechners verzichtet auf beschreibbare Datenträger wie z.B. Festplatten. Das System wird komplett aus dem Netz oder gegebenenfalls von einer CD-Rom gebootet; die zu veröffentlichenden Daten werden in regelmäßigen Abständen über das Netz in den Hauptspeicher geladen (RAM-Disk).

3.2.5.3 Administration/Personal

Da der Rechner, der die Web-Präsenz im Internet sichtbar macht, zu diesem Zweck selbst mit dem Internet verbunden sein muss, stellt seine sichere Konfiguration für die Administratoren eine echte Herausforderung dar. Nur ausreichend qualifizierte Administratoren sind in der Lage, die Konsequenzen ihrer Konfigurationsarbeiten wirklich abzuschätzen.

Aber auch ein an sich sicher konfiguriertes System unterliegt oft durch seine Benutzer Sicherheitsrisiken. Nachlässiger Umgang mit Konsolen und der Zugriff auf den Rechner über ungeschützte Zugänge sind häufig auftretende Risiken, die trotz sicherer Passworte die Rechner angreifbar machen. Abhilfe kann hier nur geschultes zuverlässiges Personal bringen. Mitarbeiter, die abschätzen können wie groß die Verantwortung ist, die ihnen durch den Umgang mit dem Rechner übertragen wurde, sind in der Lage, die Sicherheit des Systems zu gewährleisten.

In vielen Fällen, wenn die Web-Präsenz nicht auf einem Rechner des Unternehmens publiziert wird, sondern bei einem Internet-Provider, wird die Verantwortung an diesen weitergeleitet. Dies hat den Vorteil, dass von vornherein klar ist, wer für eventuelle Sicherheitsprobleme verantwortlich zeichnet. Dem Betreiber der Web-Präsenz wird lediglich das Recht eingeräumt, die Web-Präsenz auf die Maschine des Service-Providers zu publizieren. Die Administration dieser Maschine obliegt diesem Dienstleister.

Kapitel 4

Ausgangslage – Komponenten für die Entwicklung von JDaphne

Um der historischen Entwicklung von JDaphne Rechnung zu tragen, werden in den folgenden Abschnitten die Konzepte und Implementationen ausgewählter Projekte am Institut für Telematik, die die Ausgangslage und zum Teil auch die Basis für die Entwicklung von JDaphne waren, dargestellt. Wie alle im projektgetriebenen Umfeld stattfindenden Entwicklungen fand auch die von JDaphne in mehreren Zyklen statt. Einer ersten Konzeptionsphase mit Machbarkeitsstudien folgte eine Implementation die die Evaluation des Systems im praktischen Betrieb ermöglichte. Mit wachsenden Anforderungen wurde eine umfangreiche Erweiterung des Systems nach den Anforderungen und Bedürfnissen der Projektpartner durchgeführt; insbesondere die Anpassung an unternehmenseigene Infrastrukturen und Workflows fielen in diese Phase. Funktionen, die wünschenswert, aber nicht in das System integrierbar waren, wurden in separaten Projekten als eigenständige Komponenten konzeptioniert und implementiert. Schließlich wurde in der aktuellen Konzeption und Implementation von JDaphne der Versuch angetreten, die so entstandenen Komponenten mit den Erfahrungen aus den bis dato erfolgten Implementationen zusammenzubringen und so ein dem aktuellen Stand der Technik entsprechendes System als Plattform für weitergehende Entwicklungen im produktiven Einsatz zu erstellen.

An zentraler Stelle dieses, die Ausgangslage der Entwicklung von JDaphne beschreibenden Kapitels, steht dabei DAPHNE [HZEM99], das Online-Redaktionssystem, mit dessen Entwicklung und Betrieb wertvolle Erfahrungen gemacht wurden, die in die Implementation von JDaphne eingeflossen sind. Einerseits wurden mit DAPHNE bereits im Vorfeld Konzepte evaluiert, die in JDaphne zum Einsatz kommen, andererseits gehen aus projektbedingten Kompatibilitätszwängen zu Beginn der Entwicklung die Einflüsse von DAPHNE sogar soweit, dass Teile von DAPHNE's Datenmodell von JDaphne übernommen werden mussten und sich dementsprechend Restriktionen in der Gestaltungsfreiheit ergaben.

Zu den Erfahrungen, die aus den Projekten, bei denen DAPHNE entwickelt und eingesetzt wurde, gewonnen wurden, gehört die Erkenntnis der unbedingten Notwendigkeit eines ausgereiften Hyperlink-Managements für den Betrieb einer Web-Präsenz. Als alleinstehende oder integrierbare Komponente entwickelt, stellt die im Rahmen des Hyperlink-Management-Projektes (HLM) [RHH+99] – von der Stiftung Innovation Rheinland-Pfalz finanziert – aufgebaute Hyperlink-Management-Komponente einen wichtigen Baustein bei der Entwicklung von JDaphne dar. Die während des HLM-Projektes entworfenen Gedanken und Mechanismen schlagen sich zu einem großen Teil darin nieder.

Abschließend findet mit den Assistenten- und Gateway-Komponenten ein Themenfeld Eingang in diese Arbeit, das sich zwischen Web-Server auf der einen und Web-Browser auf

der anderen Seite platziert. Auf der Proxy-Technik aufsetztend realisiert diese Komponente Mehrwertdienste für den Benutzer am Web-Browser. Schwerpunktmäßig ist bei dieser Technologie die weitentwickelte Parse- und Filtertechnik von Interesse. Aber auch die clientseitigen Entwicklungen zur Erfassung und Abfrage von Daten sowie deren Zusammenarbeit mit den Web-Browsern sind wichtige Hilfen bei dem Aufbau von JDaphne gewesen. Gerade die dynamische Weiterverarbeitung von Daten auf dem Weg von Web-Server zum Web-Browser – das Gateway-Konzept – stellt ein interessantes Vorgehen dar, um ohne auf dem Web-Server einen Einfluss auf die Daten zu nehmen diese dennoch persönlich für den Benutzer am Web-Browser aufzubereiten. Über die Mechanismen zur digitalen Signierung durch den Gateway übertragener Daten ist zudem eine unabhängige Nutzung des Systems durch unabhängige Institutionen in Form eines „elektronischen Notars“ (vgl. [DH89]) realisierbar. Schließlich und endlich bietet der Einsatz der Assistenten-Komponenten in Kombination mit JDaphne den Autoren ein gerade für die Recherche im Internet geeignetes Tool, mit dem das Auf- und Wiederfinden von bereits in der Historie besuchten Hyperlinks auf externe Web-Präsenzen gezielt möglich wird.

Während die aktuelle Version von JDaphne im sich anschliessenden Kapitel in aller Ausführlichkeit beschrieben wird, dienen die vorherige Version DAPHNE sowie die beiden anderen erwähnten Komponenten der Diskussion des Entwicklungsprozesses in der Übersicht und bedienen sich daher einer eher globaleren Beschreibung der zugrundeliegenden Konzepte und Funktionalitäten.

4.1 DAPHNE

Basierend auf den vielfältigen, dem Institut für Telematik zugänglichen Erfahrungen mit dem elektronischen Publizieren von Dokumenten, die ihre Wurzeln in den Projekten *„Electronic Colloquium on Computational Complexity“* (ECCC) – Research reports, surveys and books in computational complexity [BDM96] – und des *„Zentrums für Wissenschaftliches Elektronisches Publizieren“* (WEP) der Universität Trier haben, wurde schnell das Potential von Internet-basierten Redaktionssystemen erkannt. Über die Nikolaus-Koch-Stiftung fand sich zusammen mit einer Trierer Tageszeitung die Möglichkeit des Aufbaus eines solchen Systems [GZEM97] für den produktiven Einsatz zur Erstellung einer tagesaktuellen Online-Ausgabe der Zeitung (siehe Abb. 4.1). Ziel des Projektes war die Konzeption eines Systems, das den Import bereits für die Printausgabe erstellter Artikel (SGML-Format) ermöglichte und diese nach einer internet-gerechten Aufbereitung über einen an das Zeitungswesen angelehnten Workflow in den entsprechenden Ressorts der Web-Präsenz veröffentlichte. Mit dem Akronym DAPHNE für „Distributed Authoring and Publishing of Hypertexts in a Network Environment“ wurde ein sprechender Name für das im Rahmen des Projektes der Online-Zeitung am Institut für Telematik entwickelte Redaktionssystem [ZHEM99a, ZHEM99b] gefunden. Mittlerweile in verschiedenen Varianten sowohl für den Aufbau von Web-Präsenzen als auch die Verwaltung von Intranets im Einsatz, wurden die ursprünglichen Entwicklungen für den produktiven Einsatz in den verschiedenen Projekten entsprechend den konkreten Anforderungen modifiziert. Da sich die eingesetzten Techniken und die zugrundeliegenden Modelle im Laufe der Entwicklung stark geändert haben, wird auf die genaue Beschreibung der Modelle des ersten Prototypen weitgehend verzichtet und stattdessen auf die detailliert beschriebene, aktuelle Version des Redaktionssystems in Form von JDaphne verwiesen. Die folgenden Abschnitte liefern Einblicke in die grundlegenden Überlegungen und Konzepte, die mit DAPHNE umgesetzt werden konnten. Interessante Detail-Lösungen sind dabei in dem Abschnitt „weitere Ausbaustufen“ (Seite 79) aufgeführt. Sie zeigen anschaulich die aus dem Projekt-Umfeld erwachsenden Anforderungen, für die ein solches System beim produktiven Einsatz Lösungen

bereitstellen muss.

4.1.1 Konzeption

Wie bereits in der Einleitung erwähnt, sind bei der Entwicklung von DAPHNE die Bedürfnisse einer Online-Zeitung maßgeblich in die Beschreibung der Entwicklungsziele eingeflossen. Im Vordergrund stand der Aufbau eines Systems, das einen effizienten Workflow für den Import bestehender und die Erstellung neuer Artikel sowie deren Publizierungsvorgang bietet. An dem Workflow sind neben den Autoren (Reporter/Redakteure) auch die Verantwortlichen (Chefredakteure) und der Webmaster beteiligt. Im Unterschied zu dem als WCMS zu wertenden System JDaphne konzentriert sich die Konzeption von DAPHNE aufgrund des sich täglich komplett erneuernden Dokumentenbestandes für die tägliche Ausgabe der Online-Zeitung vor allen Dingen auf den schnellen initialen Publikationsvorgang eines ganzen Dokumentenbestandes. Über die Zuordnung zu Ressorts werden die Dokumente automatisiert über Übersichtsseiten in die Web-Präsenz integriert. Das weitergehende Management der publizierten Dokumente, insbesondere deren Überarbeitung, stellt nur bedingt einen Teil der Konzeption dar. Allein durch Dereferenzierung der Artikel in den Übersichtsseiten werden diese als veraltet abgestempelt, bleiben aber zu Archivierungszwecken weiterhin auf der Web-Präsenz verfügbar. Damit erfordert DAPHNE in seiner Ausprägung für die Online-Zeitung weder ein komplexes Hyperlink-Management noch über die initiale Erstellung hinausgehende Mechanismen für die Dokumentenbearbeitung. Diese Anforderungen ergaben sich instantan beim Einsatz des Systems in einem anderen Kontext und führten dementsprechend zu der Entwicklung von Zusatzkomponenten und schließlich zu dem Entschluss, mit JDaphne ein System zu entwickeln, das über für diese Zwecke ausgelegte Mechanismen verfügt.

Um die heterogene IV-Struktur, bestehend aus einer großen Anzahl von Apple-Rechnern und einer kleineren Menge von PCs, zu integrieren, wurde aus Gründen der Investitionssicherheit der Einsatz von Internet-Technologie als integrierendes Element angestrebt. In Kombination mit der stark serverzentrierten Ausrichtung hat dies den weitergehenden Vorteil, dass auf das System von beliebigen Arbeitsplätzen im Internet, z.B. auch über die Einwahl bei einem Provider, zugegriffen werden kann – eine entsprechende Konfiguration der Unternehmens-Firewall vorausgesetzt. Da keinerlei Client-Installation gefordert wird, ist verteiltes Arbeiten auch von z.B. heimischen Rechnern aus ermöglicht. Die Redakteure und auch Reporter sind

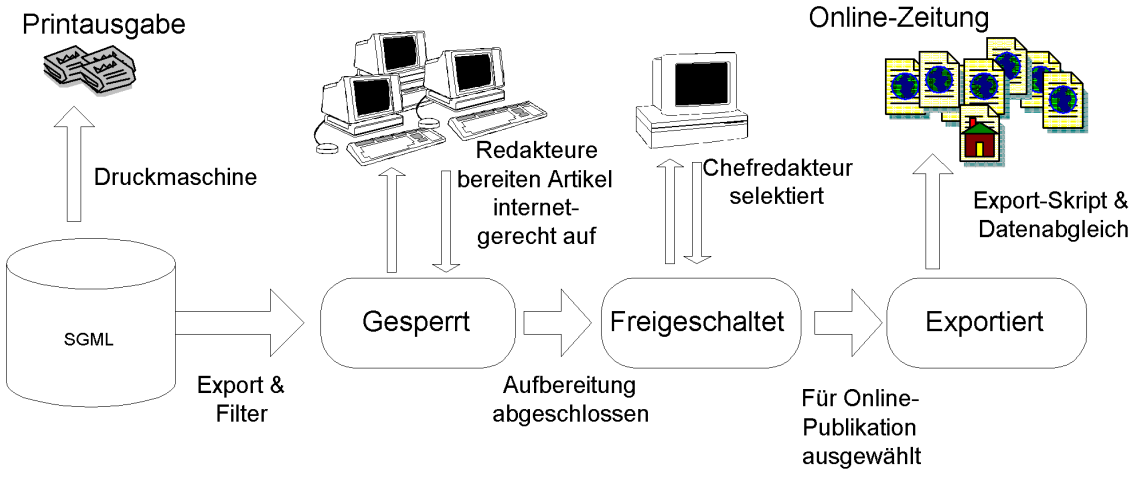

Abbildung 4.1: DAPHNE im Einsatz bei einer Online-Zeitung. Der prinzipielle Workflow.

ferner auf diese Weise in der Lage, aktuelle Meldungen von beliebigen Plätzen weltweit in die Online-Ausgabe zu schreiben, sofern sie über einen Web-Browser mit Intenet-Zugang verfügen.

Da unter den Aufgaben des Redaktionssystems die serverseitige Filterung und Modifikation von Dokumenten einen großen Anteil haben und bei der Skriptsprache PERL mächtige reguläre Ausdrücke zur Verfügung stehen, wurde in der Konzeptionsphase trotz der Nachteile, die Skripte mit sich bringen, eine skriptbasierte Lösung favorisiert. Der langsameren Ausführungsgeschwindigkeit und der schlechteren Wartbarkeit stand weiterhin eine größere Flexibilität bei der Reaktion auf neue und geänderte Anforderungen positiv gegenüber.

Um die Benutzer des Systems zu authentifizieren, sieht die Konzeption von DAPHNE die Integration der Web-Server-seitig implementierten Authentifizierung vor. Das Abgreifen der Authentifizierungsinformation aus der Skript-Umgebung bringt es mit sich, dass für das Redaktionssystem keine eigenständige Login-Funktionalität benötigt wird. Durch die Konfiguration von Zugriffsberechtigungen auf Verzeichnisse können so der Zugriff auf Dokumente und die Ausführberechtigung für Arbeitsschritte verwaltet werden. Bei DAPHNE wird an Hand von Rollen die Autorisierung für Aktionen verwaltet. Jeder Rolle wird ein eigenständiges CGI-Verzeichnis zugeordnet. Über die Berechtigung, auf ein CGI-Verzeichnis zuzugreifen, erhält der Benutzer die Autorisierung, eine bestimmte Funktion aufzurufen. Ob die Aktion letztendlich ausgeführt wird, hängt von der Zugriffsberechtigung des Benutzers auf die von der Funktion verwendeten Objekte ab. Diese Berechtigung hängt im wesentlichen an den Strukturelementen. Entscheidendes Struktur- und Ordnungselement innerhalb von DAPHNE sind Ressorts. Angelehnt an den Sprachgebrauch der Presse sind diese verzeichnisgleichen Objekte parallel Träger der Zugriffsberechtigungsinformation auf Dokumente und Strukturierungsmerkmale für das erzeugte Produkt, die Web-Präsenz. Innerhalb des Dateisystems existieren die Ressorts nicht; bei der Verwaltung und der Navigation in der exportierten Web-Präsenz haben sie die Funktion von virtuellen Verzeichnissen.

Während DAPHNE nach außen eine Struktur bezüglich der Dokumente und Verzeichnisse bereitstellt, werden auf der technischen Ebene intern nur für den Status der Dokumente genutzte Verzeichnisstrukturen unterstützt. Alle Dokumente, die sich im gleichen Bearbeitungsstatus (s.u.) befinden, werden in dem gleichen Verzeichnis gespeichert. Dieses Vorgehen begünstigt die Verwaltung der Verknüpfungen zwischen den Dokumenten im Vergleich zu einer verzeichnisorientierten Lösung erheblich. Da sich alle Dokumente auf der gleichen Verzeichnisebene befinden, sind sämtliche Links relativ und ohne Verzeichnisnamen. Insbesondere wenn ein Dokument von einem virtuellen Verzeichnis (Ressort) in ein anderes verschoben wird, ist keinerlei Aufwand in der Link-Anpassung notwendig.

Der globalen Konsistenzwahrung in DAPHNE dient die Verwendung von eindeutigen Ziffernfolgen (Nummern) als Dokumentnamen. Sobald ein Dokument, egal welchen Typs, eingespielt wird, erhält es vom System eine fortlaufende Nummer. Diese Konzeption vermindert die Probleme, die aus inkonsistent oder fehlerhaft eingegebenen Dateinamen in Hyperlinks entstehen, deutlich. Dem Benutzer nimmt es Arbeit ab, da die Nummer des referenzierten Dokumentes viel schneller eingegeben werden kann als der u.U. sprechende Name des Dokuments. Weiterhin lassen sich solche Namen beim automatisierten Import von Dokumenten am leichtesten automatisch generieren.

Der Workflow von Dokumenten innerhalb von DAPHNE wird über einen Dokumentstatus [ZHZ+00] gesteuert; in der ursprünglichen Version waren dies die Statuszustände „gesperrt“, „freigeschaltet“ und „exportiert“, die jedes Dokument durchlaufen musste, um auf der Web-Präsenz sichtbar zu werden. Mit einer Statusänderung eines Dokumentes geht dieses in den Zuständigkeitsbereich einer anderen Rolle über. Jedem Status innerhalb von DAPHNE ist ein eigenes Verzeichnis zugeordnet. Deshalb liegen von jedem Dokument verschiedene Versionen,

jeweils dem Status entsprechend, vor. Für andere Einsatzfelder von DAPHNE und insbesondere auch JDaphne wurde dieses Konzept unter Erweiterung der möglichen Statuswertemenge beibehalten.

4.1.2 Komponenten

Bei DAPHNE handelt es sich um eine Client-Server-Entwicklung (siehe Abb. 4.2). Im Folgenden sind die server- und clientseitig eingesetzten Komponenten erläutert.

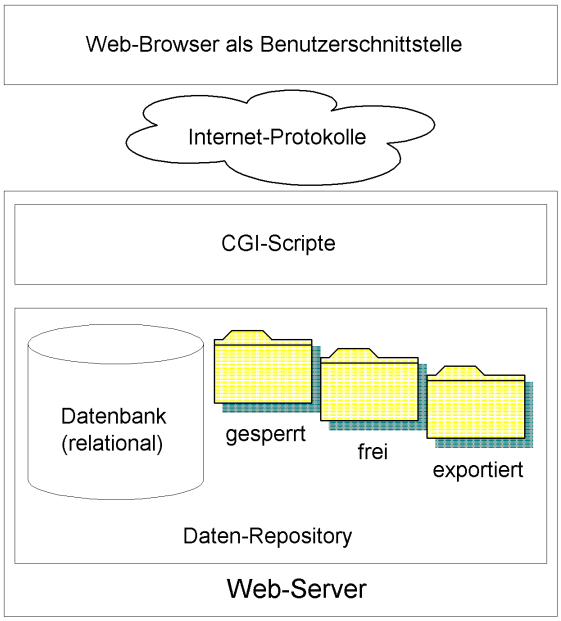

Abbildung 4.2: Komponenten und Architektur von DAPHNE.

4.1.2.1 Server

DAPHNE setzt zwei verschiedene Server ein: für die Dokumentenverwaltung eine Datenbank und für das Ausführen der Skripte und die Verteilung von Dokumenten einen Web-Server. DAPHNE ist nur bedingt abhängig von der eingesetzten Datenbank. Das Datenmodell, welches auf der Basis der mit der GNU Public Licence vertriebenen *MySQL*-Datenbank, gemäß dem ANSI Standard II entwickelt wurde, lässt sich verhältnismäßig einfach auf andere Datenbanken übertragen. Diese Portabilität ergibt sich durch den geringen Grad von Logik, der innerhalb der Datenbank abgelegt ist. Da die *MySQL* nur Ansätze einer integritätswahrenden Logik bereitstellt, wurde diese in den Skripten realisiert. Bei der Anpassung an andere Datenbanken hilft dieser Punkt erheblich, weil die von der Datenbank bereitgestellten Möglichkeiten von Hersteller zu Hersteller variieren. Durch den Einsatz einer allgemeinen Datenbank-Schnittstelle für die verwendete Skript-Sprache werden außerdem die Datenbankzugriffe gut gekapselt.

Je nach dem, wie aufwendig die Anpassung an eine andere Datenbank gestaltet werden soll, kann natürlich auch deren interne Logik verstärkt eingesetzt werden. Dazu muss im wesentlichen nur das Datenbankmodul modifiziert werden. Als Web-Server, der die Arbeitsumgebung der Skripte bereitstellt, wird der *Apache* Web-Server eingesetzt. Neben der freien

Verfügbarkeit unter der GNU Public Licence zeichnet er sich durch einfache Konfigurierbarkeit und hohe Performance aus. Innerhalb der Dokumenten-Root gibt es unter DAPHNE für jeden Status der Dokumente ein eigenes Verzeichnis, aus der historischen Entwicklung heraus mit „gesperrt“, „frei“, „exportiert“ bezeichnet. In ihnen liegen die Dokumente, die sich in dem korrespondierenden Status befinden. Der lesende Zugriff auf diese Verzeichnisse mit dem Web-Browser wird über die im Web-Server verfügbare Authentifizierung geregelt. Wie in der Konzeption beschrieben verwaltet, der Web-Server für jede Rolle ein eigenständiges CGI-Verzeichnis. Die Verzeichnisnamen sprechen dabei für sich:

- „cgi-Autor“
- „cgi-Freischaltungsberechtigter“
- „cgi-Webmaster“
- „cgi-Administrator“

Analog zu den Dokumenten-Verzeichnissen wird über die Server-Authentifizierung bestimmt, welcher Benutzer berechtigt ist, in einer Rolle aktiv zu werden, d.h. auf ein bestimmtes CGI-Verzeichnis zuzugreifen. Von technischer Seite her interessant ist dabei, dass gemeinsame Komponenten, d.h. Module oder Skripte, über symbolische Links für die jeweiligen Verzeichnisse verfügbar gemacht werden und so eine Code-Multiplikation verhindert wird.

4.1.2.2 Client

Auf Client-Seite ist DAPHNE betriebssystemunabhängig. Außer einem Web-Browser und entsprechenden Editor-Programmen für die Dokumente werden keinerlei clientseitige Komponenten benötigt. Die Benutzerschnittstelle für DAPHNE ist ein Web-Browser. Die vollständig HTML-basierte Schnittstelle bringt einen hohen Grad an Unabhängigkeit von der eingesetzten Browser-Version mit sich. Eine zwingend erforderliche Funktionalität, die das System von dem Web-Browser erwartet, ist der Datei-Upload: Dokumente werden vom lokalen Arbeitsplatz des Autors von dem Web-Browser an die Server-Komponente übertragen. Fehlt dem Browser diese Funktionalität, ist das Einspielen von Dokumenten direkt aus dem Browser heraus in das System nicht möglich. Ein entsprechender Workaround mit selbst entwickelten Hilfsprogrammen, die diesen Upload abwickeln können, ist in der Beschreibung weiterer Ausbaustufen (siehe Seite 79) skizziert.

Als Editoren können von den Benutzern beliebige Programme eingesetzt werden. Innerhalb von DAPHNE werden zusätzlich eigene Mime-Types definiert. Wird ein Dokument zur Bearbeitung aufgerufen, so übermittelt der Web-Server mit dem Dokument den entsprechenden, den Aufruf des im Web-Browser eingestellten Editors bewirkenden, Mime-Type mit. Der Web-Browser startet dann den Editor als sogenannte Hilfs-Applikation mit dem zum Bearbeiten aufgerufenen Dokument. Problematisch bleibt allein das Wiedereinspielen der bearbeiteten Dokumente in das System zurück. Hier ist von dem Autor manuelle Arbeit erforderlich. Er muss das bearbeitete Dokument wie ein neu erstelltes Dokument mit dem Browser dem System wiederzuführen. Bei den weiteren Ausbaustufen (siehe Seite 79) wird ein für diese Problematik entwickelter Workaround beschrieben.

4.1.3 Publizieren mit DAPHNE

Über welchen Workflow genau mit DAPHNE Dokumente auf die Web-Präsenz publiziert werden, hängt von der betrachteten Implementation ab. Allen im Laufe der Zeit in den produktiven Einsatz überführten Versionen von DAPHNE ist gemeinsam, dass Autoren die

Dokumente erstellen und bearbeiten. Dokumente, die gerade frisch importiert wurden, neu erstellt werden oder sich in Bearbeitung befinden, werden als „gesperrt“ betrachtet. Sind sie fertiggestellt, so werden sie von den Autoren „freigeschaltet“. Freigeschaltete Dokumente stehen für die Publikation zur Verfügung. Ob die Dokumente nun direkt publiziert werden oder noch einen weiteren Abzeichnungsschritt durchlaufen, unterscheidet sich je nach Anforderung der Projektpartner, bei denen DAPHNE zum Einsatz gekommen ist. Aus der Sicht der Online-Zeitung betrachtet, wählt beispielsweise der Chef-Redakteur aus dem Bestand der freigeschalteten Dokumente diejenigen aus, welche für die aktuelle Ausgabe der Zeitung in Frage kommen; diese werden von DAPHNE publiziert. Publizieren bedeutet, dass die Dokumente mit Layout versehen werden. Neben Tag-Ersetzungsregeln, die eine Umformatierung der Inhalte ermöglichen, findet der Einbau dieser formatierten Inhalte in die gewünschten Layout-Templates statt. Sowohl in Tabellenstrukturen publizierte Web-Präsenzen als auch Frame-Sets können beim Schreiben der Dokumente erzeugt werden. Gleichzeitig mit dem Publikationsprozess wird eine automatisierte Deckseite für jedes Ressort erstellt. Sie enthält vom Redakteur konfigurierbar Referenzen auf diejenigen Dokumente, die publiziert wurden und die Teil der aktuellen Web-Präsenz sein sollen.

Die in periodischen Zeiträumen erstellte Web-Präsenz wird in einer Archiv-Datei verpackt. In komprimierter Form wird sie auf den externen, d.h. im Internet befindlichen Rechner mit dem Web-Server transferiert. Dort wird sie ausgepackt und steht damit in einer neuen Version zum Abruf über Web-Browser bereit. Wie bereits erwähnt, werden bei DAPHNE alle Dateien mit Nummern als Dateinamen in der Dokumenten-Root des Web-Servers abgelegt. Die Navigationselemente der publizierten Dokumente bieten den Benutzern der Web-Präsenz über ihre Navigationsleisten virtuelle Verzeichnisstrukturen an. Diese spiegeln sich jedoch nicht in der im Web-Browser angezeigten URI wieder, in der nur die Dokumentnummern erscheinen.

4.1.4 Weitere Ausbaustufen

Nach dem Aufbau von DAPHNE zum Einsatz in der Online-Redaktion kam das System in der Folge auch in anderen Umfeldern zum Einsatz. Hier musste die auf die Online-Zeitung ausgelegte Konzeption erweitert werden. Absehend von kleineren Modifikationen, wie sie zunächst in Form von Wartungs- und Anpassungsarbeiten notwendig wurden, die hier aber aufgrund ihrer Insignifikanz keine Erwähnung, hat das System für den projektorientierten Einsatz auch zum Teil substantielle Änderungen durchlaufen. Diese in erweiterten Ausbaustufen von DAPHNE implementierten Funktionalitäten und Änderungen im Konzept sollen hier, insbesondere aufgrund ihrer richtungsweisenden Funktion für JDaphne, erläutert werden.

4.1.4.1 Unterverzeichnisse/Sub-Ressorts

Die wohl einschneidenste Modifikation betrifft die von DAPHNE bereitgestellte Strukturierung einer Web-Präsenz durch Ressorts. Während für den Einsatz im Online-Zeitungsbereich die von DAPHNE initial bereitgestellte flache Ressorthierarchie gewünscht war, erforderte die Generalisierung des Systems alsbald die Entwicklung von dateisystemanalogen Ressortstrukturen. Aus diesem Grund wurde das Strukturelement „Ressort“ um die Sub-Ressorts bzw. Unterverzeichnisse erweitert. Mit diesen Elementen ist es möglich, eine beliebige Verzeichnistiefe zu konstruieren. Da sowohl Ressorts als auch Sub-Ressorts ihren virtuellen Charakter bezüglich des Dateisystems behalten haben, hatte diese Ausbaustufe keinerlei Auswirkung auf die Verlinkung. Die den Sub-Ressorts zugeordneten Dokumente wurden nach wie vor auf der zentralen Verzeichnisebene abgelegt. Die Berechtigungsverwaltung, die bislang an die

Ressorts geknüpft war, wurde in diesem Schritt ebenfalls an die Sub-Ressorts geknüpft. Dadurch ergab sich eine feinere Granularität bei der Einstellung von Benutzerberechtigungen. Größere Modifikationen waren in den die Benutzerschnittstellen darstellenden Skripten und dem Export in die Web-Präsenz notwendig. Hier bedeutete diese Ausbaustufe einen großen Fortschritt für das Gesamtsystem.

4.1.4.2 Upload durch Hilfsapplikationen

Eine weitere richtungsweisende Modifikation von DAPHNE, die von dem Konzept her auch in JDaphne wiederzufinden ist, betrifft den Editiervorgang der serverseitig gespeicherten Dokumente auf der Client-Seite. Wie oben angedeutet, hat der großmaßstäbliche Einsatz offener Standards im Fall von DAPHNE deutliche Einbußen im Bedienkomfort zur Folge. Besonders signifikant schlägt sich dies im Upload bearbeiteter Dateien nieder. Der Benutzer muss sich „merken“, wo er die bearbeitete Datei abgespeichert hat, und, nach der Auswahl des entsprechenden Formulars, das alte Dokument in DAPHNE manuell ersetzen. Abgesehen davon, dass einige ältere Web-Browser eine Upload-Funktion überhaupt nicht anbieten, ist der vom Benutzer verlangte Verwaltungsaufwand für die temporär auf seinem lokalen Arbeitsplatz abgelegten Dateien, wie die Erfahrung gezeigt hat, zu groß. Dies wurde zum Anlass genommen, eigenständige Programme zu erstellen, die als Hilfsapplikationen vom Browser aufgerufen, auf jedem Arbeitsplatz lokal die benötigte Funktionalität gekapselt bereitstellen. Am Beispiel der „Bearbeiten-Hilfsapplikation“ seien die implementierten Schritte erläutert: Die Hilfsapplikation wird gestartet durch die Übersendung eines speziell für diesen Zweck definierten Mime-Types. Als Parameter, der von dem Web-Browser an die Applikation übergeben wird, dient ein virtueller Dateiname, der von den Skripten auf dem Web-Server generiert wird. Die Applikation parst diesen virtuellen Dateinamen, um die Parametrisierung auszulesen. Initialisiert mit den übergebenen Parametern und einer zusätzlichen Konfigurationsdatei verbindet sich das Programm mit dem Web-Server und fordert mittels HTTP von dem entsprechenden Skript in DAPHNE die zu bearbeitende Datei an. Basierend auf dem für das Dokument übergebenen Mime-Type wird die in der Konfigurationsdatei konfigurierte Editor-Anwendung mit dem lokal abgelegten Dokument geöffnet. Während der Benutzer das Dokument editiert bleibt die Hilfsapplikation im Hintergrund und wird erst dann wieder aktiv, wenn er den Editor beendet. Mit der Beendigung dieses Prozesses liest die Hilfsapplikation das bearbeitete Dokument ein und spielt es nach Rückfrage beim Benutzer wieder in das DAPHNE-System zurück. Der erfolgreiche Einsatz dieser Hilfsapplikation setzt voraus, dass das bearbeitete Dokument vom Benutzer nicht umbenannt oder an eine andere Stelle gespeichert wurde, weil sonst die Hilfsapplikation nicht mehr auf die Datei zugreifen kann. Analog zu der „Bearbeiten-Hilfsapplikation“ wurden Applikationen zum initialen Upload und dem ersetzenden Upload geschrieben, da der eingesetzte Web-Browser nativ nicht über die notwendige Upload-Funktionalität verfügte. Insgesamt hat sich im praktischen Einsatz das Konzept der Hilfsapplikationen von DAPHNE bewährt. Es nimmt dem Benutzer die allein über den Web-Browser nicht anzubietende Verwaltung der Datei auf dem lokalen Arbeitsplatz ab und kann im Fall fehlender Web-Browser-Funktionalität als Workaround genutzt werden.

4.1.4.3 Internet/Intranet

Auch wenn, wie in der Konzeption beschrieben, DAPHNE nicht konkret für den Einsatz im Intranet zum Zweck der Dokumentenverwaltung entworfen wurde, ist dennoch ein Einsatz in dieser Richtung möglich. Projektorientiert wurde DAPHNE dahingehend erweitert, dass den speziellen Anforderungen des Intranets Rechnung getragen wird. Diese liegen zum großen Teil in den hier zum Einsatz kommenden Dokumententypen. Während für das Inter-

net primär HTML und PDF als Formate für Textdokumente eingesetzt werden, findet sich im Intranet die volle Bandbreite der im Unternehmen eingesetzten Formate. Mit im Vergleich zum Internet-Bereich geringeren Anforderungen an das Mehr-Augen-Prinzip erfolgt der Export dieser Dokumente im Intranet direkt von den Kontrollinstanzen. Dies trägt der deutlich höheren Aktualisierungsrate des Dokumentenbestandes für den internen Gebrauch Rechnung. Auch die Ressortstruktur befindet sich im ständigen, beispielsweise an die Projekte des Unternehmens angepassten, Änderungsprozess.

4.1.4.4 Ressortmanager

Für Änderungen der Ressortstruktur bietet DAPHNE keine Schnittstelle an. Die Erstellung und Modifikation der Struktur erfolgt bei der Initialisierung des Systems in der Datenbank durch qualifiziertes Personal. Spätere Varianten von DAPHNE haben an dieser Stelle angesetzt und eine skriptbasierte Lösung geschaffen, mit der neue Ressorts von der Administration eingefügt werden können. Um die im Intranet-Bereich häufig anfallenden Modifikationen der Ressort-Struktur komfortabler durchführen zu können, wurde ein eigenständiges Programm, der Ressortmanager, entwickelt. Dieses als Java-Applet mit direktem Datenbankzugriff realisierte Programm erlaubt die Modifikation der Ressortstruktur analog zu den Dateimanagern im Dateisystem. Weiterhin kann die Ansicht so umgestellt werden, dass auch die in den Ressorts abgelegten Dokumente in tabellarischer Form visualisiert werden. Die Administrationsoberfläche des Ressortmanagers, die in übersichtlicher Form die Inhalte des Redaktionssystems visualisiert, kann als Urvater der mit JDaphne realisierten Java-Bedienoberfläche betrachtet werden.

4.1.4.5 Verwaltung mehrsprachiger Dokumente

Eine im grenznahen Umfeld von Trier (Deutschland, Luxemburg, Frankreich) häufig an das Redaktionssystem herangetragene Forderung ist der Wunsch nach der Fähigkeit, Dokumente in verschiedenen Sprachversionen verwalten und in die Web-Präsenz publizieren zu können. In der Implementierung von DAPHNE wurde daraufhin das Datenmodell um Sprachattribute erweitert. Um den Modifikationsaufwand gering zu halten, wird die Sprache in erster Linie als Eigenschaft der Ressort-Entität betrachtet. Die verschiedenen Sprachversionen eines Ressorts behalten den gleichen Schlüssel und unterscheiden sich in dem Sprachattribut sowie dem Ressortnamen in der jeweiligen Sprache. Alle Dokumente bekommen als Attribut die Sprache der Sprachversion des Ressorts, dem sie angehören. Den am Dokumentenworkflow beteiligten Rollen wird die jeweilige Sprache als eigene Spalte in den Übersichtstabellen visualisiert.

Für die Verzeichnisstruktur erwartet die mehrsprachige Version von DAPHNE eine vollständige Sprachensymmetrie: jedes Ressort muss in jeder Sprache verfügbar sein. Für die in diesen Ressorts mit Sprachausprägung abgelegten Dokumente wird keinerlei Symmetrie erwartet. Jedes Ressort kann in jeder Sprache beliebige Dokumente aufnehmen, die nicht zueinander in Beziehung stehen müssen. Beim Export werden die vorhandenen Sprachversionen für jede Sprache in separate Verzeichnisse publiziert. Für die so generierte Web-Präsenz bedeutet dies, dass über eine gemeinsame Einstiegsseite direkt unter der Dokumenten-Root in die verschiedenen Sprachversionen verzweigt wird.

4.1.5 Bewertung/Zusammenfassung

Mit DAPHNE ist zu einem sehr frühen Zeitpunkt ein mit Internet-Technik arbeitendes System zum Erstellen einer Online-Zeitung realisiert worden. Durch den sukzessiven Einsatz in anderen Produktionsumgebungen mit den damit verbundenen Anpassungen hat sich die

Funktionalität des Systems stark erweitert. Die in DAPHNE implementierten Konzepte haben sich als hinreichend für die initialen Anforderungen erwiesen und blieben auch nach den dargestellten Modifikationen tragfähig. Nichtsdestotrotz hat sich gezeigt, dass dem System wichtige Eigenschaften – der Workflow für das Bearbeiten von bereits publizierten Dokumenten und eine daran angepasste Hyperlink-Verwaltung – fehlen. Weiterhin hat sich die Robustheit und Einfachheit des Systems im produktiven Einsatz durchaus bewährt, der Einsatz von Skripten als Programmierlösung jedoch schnell zu einer ständig wachsenden Unübersichtlichkeit des Programmcodes geführt. Der Verzicht auf objektorientierte Programmierung in Verbindung mit dem Fehlen von Compilern gestaltete die Weiterentwicklung und Modifikation des Programm-Codes zunehmend schwieriger. Aus diesen Gründen wurde die im Vordergrund dieser Arbeit stehende Neuimplementation „JDaphne“ in die Wege geleitet. Aus Kompatibilitätsgründen für die Übergangsphase mit einem analogen Datenmodell arbeitend, versucht JDaphne die Konzepte und Erfahrungen von DAPHNE neu in objektorientierter Form mit dem Ziel einer höheren Benutzerfreundlichkeit umzusetzen.

4.2 Hyperlink-Management-Komponente

Wie bei der Konzeption von DAPHNE beschrieben, bietet dieses System nur minimale Fähigkeiten für ein Hyperlink-Management; lediglich die Umsetzung von Dateinamen beim Import in die internen Dokumentennummern wird von DAPHNE unterstützt. Der Einsatz von DAPHNE zum Web-Präsenz-Management erfordert neben den angesprochenen Modifikationen im Dokumenten-Workflow vor allem eine Hyperlink-Management-Funktionalität. Eine solche wurde am Institut für Telematik in einem separaten Projekt entwickelt.

Bei der Konzeption des Redaktionssystems wurde die Verknüpfung von Dokumenten mittels der in HTML verfügbaren Hyperlinks als Basis genommen. Diese Entscheidung befreit davon, „eigene“ Anzeige-Programme entwerfen zu müssen, die das Browsen nach proprietären Standards verknüpfter Dokumente ermöglichen. Stattdessen wurde der Standard des WWW übernommen. Dokumente können somit von allen derzeit verfügbaren Web-Browsern angezeigt werden. In diesem Abschnitt wird ie Konzeption und Entwicklung einer solchen Hyperlink-Management-Komponente, deren Notwendigkeit aus dem Redaktionssystemumfeld erwächst, beschrieben.

4.2.1 Ziele

Im Rahmen des Projektes „Hyperlink-Management-Komponente“ (HLM) sollte die Entwicklung eines Systems stattfinden, das effiziente Mechanismen zur Verwaltung von Hyperlinks in multilingualen Web-Präsenzen bereitstellt. Für ein solches System kann eine Reihe von Zielen formuliert werden, die im Folgenden erläutert werden. Neben globalen Zielen wird dabei auch auf konkret mit der im Rahmen des Projektes verknüpfte Forderungen eingegangen.

4.2.1.1 Hyperlink-Management

Zum Hyperlink-Management gehören im wesentlichen zwei Ziele: die Konsistenzhaltung und die Generierung von sinnvollen Verknüpfungen durch entsprechende Vorschläge, die dem Autor eines Dokumentes offeriert werden (vgl. dazu beispielsweise den Aufsatz von Wilkinson und Smeaton [WS99]).

4.2.1.1.1 Link-Konsistenz

Wenig sticht Besuchern einer Web-Präsenz direkter ins Auge als ein nicht funktionierender Hyperlink. Wenn statt des erwarteten Dokuments auf die Auswahl eines Hyperlinks folgend eine Fehlerseite zurückkommt, ist dies für den Besucher ein unmittelbares Signal dafür, dass die Web-Präsenz in sich nicht konsistent ist. Diese formale Inkonsistenz bewirkt eine negative Einschätzung der auf der Web-Präsenz veröffentlichten Inhalte.

Dabei sind die Ursachen solcher Link-Inkonsistenzen oft normale Modifikationen des publizierten Dokumentenbestandes, jedoch ohne Berücksichtigung sämtlicher von den Modifikationen betroffener Dokumente. Derartige Inkonsistenzen ergeben sich in der täglichen Arbeit häufig durch das unbefangene Löschen, Umbenennen oder Verschieben von publizierten Dokumenten . Die einzige Möglichkeit, das Auftreten von Link-Inkonsistenz wirkungsvoll zu vermeiden, besteht darin, im voraus alle relevanten Dokumente zu selektieren und zu modifizieren.

Alternativ müssen Funktionen entwickelt werden, die regelmäßig eingesetzt, nach fehlerhaften Hyperlinks suchen und diese zur Korrektur vorschlagen. Effizient ist eine solche Link-Konsistenz-Überwachung, wenn diese präventiv eingesetzt werden kann. Bevor ein Dokument von der Web-Präsenz entfernt oder in einer anderen Art und Weise in seiner URI modifiziert wird, können alle durch diese Aktion betroffenen Dokumente gefiltert und modifiziert werden; so wird das Auftreten von Inkonsistenzen a priori vermieden. Insbesondere vermindert ein solches Instrumentarium für die Verantwortlichen die Risiken bei der Arbeit am Dokumentenbestand.

Link-Konsistenz kann automatisiert überwacht und erhalten, aber nicht ohne weoiteres automatisch erzeugt werden, da hierzu die Kenntnis der Autorenintention notwendig wäre. Durch interaktive Rückfrage beim Autor im Fall aufgetretener Inkonsistenz kann aber eine automatisierte Unterstützung für die Konsistenzhaltung vorgehalten werden.

Ein weiterer Punkt der Link-Konsistenz bezieht sich auf die inhaltliche Relevanz der Hyperlinks. Selbst wenn ein Link formal auf die korrekte URI verweist, kann der Link aus inhaltlicher Perspektive „ins Leere gehen“, sofern sich der Bezug geändert hat. Konsistenz dieser Art kann automatisiert nur schwer überwacht werden; statistische Analysen der mit Hyperlinks verknüpften Dokumente sind zwar möglich, können aber nur unzulänglich das inhaltliche Verständnis der Dokumente und damit eine Beurteilung der Relevanz eines Hyperlinks möglich machen.

4.2.1.1.2 Link-Vorschläge

Wird eine neues Dokument erstellt, so steht der Autor in vielen Fällen vor dem Problem, welche anderen Dokumente auf der Web-Präsenz das neue Dokument referenzieren und umgekehrt, welche vorhandenen Dokumente auf das neue Dokument verweisen sollten. Um solche Vorschläge auf semantischer Basis abzuwickeln, ist ein hoher Grad von Aktivitäten durch die Autoren notwendig (Metadaten); weniger erfolgreich, dafür aber im größtenteils automatisiert durchzuführen, sind statistische Verfahren. Basierend auf Analysen des neuen Dokuments und der bestehenden Dokumentenbasis können Hyperlinks vorgeschlagen werden, die der Autor manuell ausgewerten und bei Gefallen umgesetzen kann. Ziel der Hyperlink-Management Komponente in Bezug auf Hyperlink-Vorschläge [HHR+99a] ist die Implementation eines Algorithmus, der für Dokumente unter Ausnutzung der vorhandenen Meta- und Linkinformationen der Dokumentenbasis unter Ausnutzung von Case-Based-Reasoning Techniken [HRH+00b] auf statistischer Basis Hyperlink-Vorschläge erstellt.

4.2.1.2 Multilinguale Dokumente

Mit der fortschreitenden Internationalisierung und dem wachsenden Interesse gerade kleinerer Unternehmen, durch die mit der Internet-Plattform mögliche Internationalisierung ihren Markt zu vergrößern, steigt das Interesse an multilingualen Web-Präsenzen. Deshalb ist eine weitere Zielsetzung bei der Entwicklung der Hyperlink-Management-Komponente die Fähigkeit, mit multilingualen Versionen eines Dokumentes umgehen zu können [RHH+99]. Erstrebenswertes, weil auch von kleineren Unternehmen machbares Szenario ist die Existenz einer Hauptsprache der Web-Präsenz, in der die Dokumente erstellt und verknüpft werden, während alle anderen Sprachversionen durch Übersetzer abgedeckt werden. Für deren Hyperlink-Verwaltung soll die Hyperlink-Management-Komponente eine Strategie entwickeln, wobei die Verlinkung der Dokumente in der Hauptsprache für sie automatisiert adaptiert werden soll.

4.2.1.3 Anwendungsszenarien

Für die Anwendung der Hyperlink-Management-Komponente wurden zwei Szenarien definiert: das erste dieser Anwendungsszenarien beinhaltet die Integration dieser Komponente in ein bestehendes System (DAPHNE) und erfordert einen entsprechenden Anpassungsaufwand auf der Seite des integrierenden Systems, die zweite Variante zielt auf die unabhängige Anwendung der Komponente als eigenständiges Programm auf eine existierende Dokumentenbasis einer Web-Präsenz ab. Beide zusammen decken somit ein breites Spektrum des theoretisch möglichen Anwendungsfeldes ab. Im Folgenden werden sie konkretisiert.

4.2.1.3.1 Integration

Bei der Integration von Hyperlink-Management-Komponente und Dokument-Management-System müssen Schnittstellen definiert werden, über die Daten ausgetauscht und Funktionalitäten abgerufen werden können. Die Hyperlink-Management-Komponente wird im Allgemeinen als kleinere Komponente die über Konfiguration an das Dokumenten-Management anzupassende Komponente sein. Deshalb ist bei der Konzeption auf flexible Konfigurierbarkeit Wert zu legen, wobei eine Trennung von Hyperlink-Verwaltung und benötigter Dokumentenbasis erstrebenswert ist. Als Benutzerschnittstelle sollte, soweit möglich, die Schnittstelle des Dokument-Management-Systems dienen; diese muß gegebenenfalls um die Funktionalität von der Hyperlink-Management-Komponente erweitert werden.

4.2.1.3.2 Stand-Alone

Anders stellt sich die Sache bei der Stand-Alone-Anwendung „Hyperlink-Management-Komponente“ dar. Hier ist eine eigene Benutzerschnittstelle zu schaffen, die den Benutzern die Funktionalitäten des Systems zugänglich macht. Entsprechend dem Szenario wird als zugrundeliegende Dokumentenbasis der „finale“ Dokumentenbestand einer Web-Präsenz betrachtet. Dieser wird dem System durch, nach Modifikationen im Dokumentenbestand jeweils erneut notwendig werdende, Scan-Operationen bekanntgemacht. Das System muss die erforderlichen Informationen direkt dem Dokumentenbestand entnehmen können und seiner internen Datenbasis die extrahierten Informationen zugänglich machen. Anhand dieser Datenbasis können die benötigten Operationen auf der Dokumentenbasis ausgeführt werden, beispielsweise das Verschieben eines Dokumentes mit anschließender Anpassung sämtlicher betroffener Hyperlinks. Die Dokumentenbasis wird dabei von der Hyperlink-Management-Komponente modifiziert. Falls die Web-Präsenz von einem anderen System generiert wurde,

ist danach ein Rückimport der modifizierten Dokumente in dieses erforderlich, um damit die Änderungen zu übernehmen.

4.2.1.4 Zu verwaltende Hyperlinks

Obwohl mehr und mehr Web-Präsenzen mit dynamischen Hyperlinks arbeiten, beschäftigt sich die hier besprochene Hyperlink-Management-Komponente aufgrund deren immernoch hoher Relevanz nur mit statischen Hyperlinks. Sofern deren Konsistenz gegeben ist, lassen sie sich durch geeignete Export-Mechanismen als Basis für dynamische Hyperlinks verwenden. Beispielsweise lassen sich alle internen URIs vor dem Download durch den Benutzer mit einem Sitzungsschlüssel erweitern, ohne dass dies die Konsistenz gefährdet. Hyperlinks, die durch clientseitige Skript-Sprachen erst auf dem Rechner des Benutzers zusammengebaut werden, sind von dem hier angestrebten Hyperlink-Management analog zu den dynamischen ausgeschlossen. Der Aufwand, der beim Parsen solcher Skripte notwendig wird, steht in keinem Verhältnis zum erzielbaren Gewinn. Steht der Hyperlink hingegen innerhalb des Dokumentes im Klartext in dem Skript , so wird er analog zu dem HTML-Text gelesen und geparst.

4.2.2 Konzeption

Anhand der Anforderungen, die das System erfüllen soll, wird deutlich, dass zwar Einschränkungen der Funktionalität vorgesehen sind, trotz allem jedoch eine breite Funktionspalette implementiert werden muss. Um mit dem System die gewünschten Funktionalitäten einfach und flexibel bereitstellen zu können, müssen in der Konzeption Wege gefunden werden, die einen Rahmen vorgeben, ohne das System in seiner Implementation zu behindern. Im Folgenden werden die Gedanken beschrieben, die der Implementation des HLM-Systems zugrundeliegen.

4.2.2.1 Plattform- und Systemunabhängigkeit

Bei der Konzeption der Hyperlink-Management-Komponente wurde großer Wert auf ein breites Einsatzfeld gelegt. Oberste Prämisse war daher die Nutzbarkeit dieser Komponente auf verschiedenen Plattformen und die Anwendbarkeit auf „beliebige“ Web-Präsenzen. Diese sollte auch dann gegeben sein, wenn die Web-Präsenz mit einem völlig unabhängigen System erstellt wurde. Aus diesem Grunde stellen sowohl die Plattformunabhängigkeit als auch die System-Unabhängigkeit die herausragenden äußeren Merkmale der Konzeption dieser Komponente dar.

Um die Hyperlink-Management-Komponente mit so vielen Systemen wie möglich kombinieren zu können, wurde die konzeptionelle Entscheidung getroffen, das Dateisystem zum „Master“ zu erklären. Dies bedeutet, dass die Hyperlink-Management-Komponente bereits ihre Funktionalität zur Verfügung stellen kann, wenn sie mit einer bestehenden, in einem Dateisystem abgelegten Web-Präsenz initialisiert wird. Durch einen „Scan“ des Dateisystems kann sie jederzeit neu initialisiert werden und steht danach für Verwaltungsfunktionen zur Verfügung.

Die Komponente arbeitet damit bei Bedarf auch autark von der Datenstruktur des Web-Präsenz-Verwaltungs-Systems, an das sie gekoppelt wird. Der Datenaustausch erfolgt über die Auswertung der in dem Dateisystem abgelegten, ggf. „finalen“ Web-Präsenz, die mit diesem erzeugt wurde. Durch Anwenden der Schnittstellen, die die Hyperlink-Management-Komponente bietet, können Daten ebenfalls direkt aus dem anderen System einfließen. Dieser Schritt erfordert allerdings Adaptationen in dem System, welches durch die Hyperlink-Management-Komponente ergänzt werden soll oder in den Schnittstellen, welche von der

HLM-Komponente angeboten werden.

4.2.2.2 Schnittstellendefinition/Abstraktion

Die Konzeption der Hyperlink-Management-Komponente sieht eine strikte Trennung von Dokumenten und zugehörigen Metadaten, insbesondere Hyperlink-Informationen (vgl. dazu Pitkow und Jones Atlas-System [PJ96]), daher im Umfeld der Hyperlink-Management-Komponente als „Hyperdaten“ bezeichnet, vor. Die Namen für die beiden Bereiche, „Multi-Document-Pool“ und „Hyper-Data-Pool“, sind selbst erklärend. Um eine Abstraktionsschicht zu schaffen, wurden separate Schnittstellen definiert, die getrennt Zugriff auf Dokumente und Metadaten erlauben. Im Folgenden werden die beiden Schnittstellen, als „Multi-Document-Pool-Interface“ (MDPI) und „Hyper-Data-Pool-Interface“ (HDPI) bezeichnet, beschrieben (Abb. 4.3). Sie stellen die Basis für die Integration mit anderen Anwendungen bzw. den Aufbau der in den Zielen formulierten Stand-Alone-Anwendung dar.

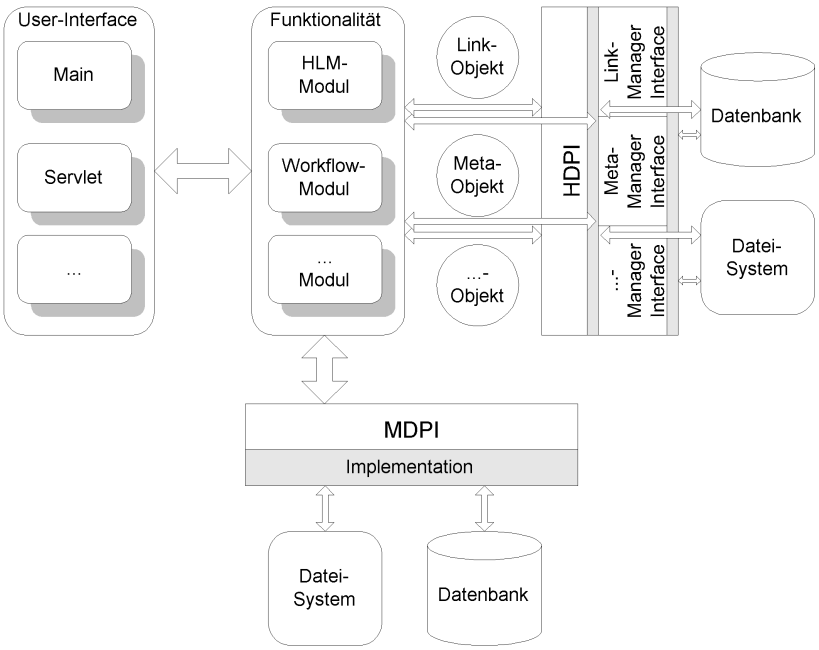

Abbildung 4.3: Der konzeptionelle Aufbau der HLM-Komponente.

4.2.2.2.1 MDPI

Das MDPI stellt für die Hyperlink-Management-Komponente die Funktionalitäten eines Dateisystems bereit. Aktionen wie Lesen und Schreiben eines Dokumentes oder Anlegen und Auslesen von Verzeichnissen sind solche Funktionalitäten. Die Definition dieser Schnittstelle dient der Schaffung einer Abstraktionsebene zwischen interner Datenstruktur und Speichermedium. Die Implementation dieses Interfaces kann die Daten deshalb sowohl in einem gewöhnlichen Dateisystem als auch in einer Datenbank halten. Da alle anderen Module über die Schnittstelle auf die Dokumente zugreifen, bleibt die interne Datenstruktur nach außen verborgen.

4.2.2.2.2 HDPI

Das HDPI stellt die Schnittstelle zu der Ablage und Abfrage von Hyperdaten dar; lediglich Basisfunktionen werden über dieses Interface abgedeckt. Spezielle Funktionalitäten werden an sogenannte „Manager“ abgegeben. Das HDPI bietet nur die grundlegenden Methoden um Instanzen solcher Manager zu erzeugen. Der für die Hyperlink-Management-Komponente interessante Manager ist der „LinkManager“. Er bietet, basierend auf HDPI und MDPI, die Funktionalitäten zum Verwalten von Hyperlinks an.

4.2.2.3 Dokumentausprägungen

Im Rahmen der Hyperlink-Management-Komponente werden Dokumente analog zum WWW über ihre URI identifiziert. Zusätzlich wird intern diese URI um weitere Parameter erweitert, die die spezielle Ausprägung (englisch: peculiarity=Seltsamkeit) bezeichnen. Solche Ausprägungen sind beispielsweise die Dokumentsprache, der Dokumentstatus und die jeweilige Version des Dokuments. Die Konzeption sieht die Erweiterung dieser Sammlung von Ausprägungen vor, auf die über die oben definierten Schnittstellen zugegriffen werden kann. Die Begründung für die Einführung solcher Dokumentausprägungen, die über URI erweiternde Parameter zugreifbar sind, liegt in dem angestrebten Hyperlink-Management: Dokumente referenzieren einander ohne Aussage über die Ausprägung des referenzierten Dokuments zu treffen, das System wahrt die Kenntnis darüber, welche konkrete Ausprägung in dem Kontext des Dokumentes relevant ist.

Referenziert beispielsweise Dokument A in der Ausprägung mit deutscher Sprache Dokument B (ebenfalls in deutscher Sprache) über eine URI, so bleibt diese URI auch dann gültig, wenn in Dokument A in die Ausprägung mit englischer Sprache die URI auf Dokument B zeigt. Unabhängig davon, ob Dokument B auch in englischer Sprache existiert, verweist Dokument A in beiden Ausprägungen auf ein vorhandenes Dokument. Findet ein Export des Datenbestandes statt, so wird in erster Linie versucht, innerhalb der gleichen Ausprägung zu verlinken. Ist dies nicht möglich, weil ein Dokument in einer Ausprägung nicht verfügbar ist, so wird über in einer definierten Reihenfolge die „nächst-beste“ Ausprägung als Hyperlink-Ziel genutzt. Das Umsetzen von Hyperlinks findet immer anhand der URI statt, ändert sie sich, so wird sie in allen Ausprägungen des Dokumentes angepasst.

Neben der speziellen Nutzbarkeit für multilinguale Web-Präsenzen, bei denen ein symmetrischer Dokumentenbestand gewünscht, aber nur teilweise verfügbar ist, eignet sich dieses Konzept der mehrdimensionalen Hyperlinks, wobei jede Ausprägung einer Dimension entspricht, darüber hinaus auch für die Versionierung von Dokumenten und Hyperlinks.

4.2.2.4 Hyperlink-Management

Das eigentliche Hyperlink-Management, die Verwaltung von Hyperlinks in einer Web-Präsenz, findet in mehreren Schritten statt. In einem ersten wird der Ist-Bestand der Web-Präsenz ermittelt. Durch entsprechende Scanner wird ihr Dokumentenbestand in die Datenpools eingelesen. Dieser initial zeitaufwendige Prozess besorgt das Parsen der Dokumente und die Extraktion von Metadaten sowie den Hyperlinks. In einem nächsten Schritt werden die Daten analysiert; dies kann automatisiert oder auf Benutzeraktion hin geschehen. Für einzelne Dokumente oder die gesamte Web-Präsenz werden z.B. die inkonsistenten Hyperlinks angezeigt. Durch den Benutzer angestoßen, sind Aktionen wie das Verschieben oder Löschen von Dokumenten möglich. Die Hyperlink-Management-Komponente unterstützt dabei das Aktualisieren von betroffenen Hyperlinks.

4.2.3 Implementation

Basierend auf der soeben in Übersichtsform erläuterten Konzeption wurde am Institut für Telematik zu Evaluierungszwecken ein Prototyp des HLM-Systems in Java implementiert. Um ihn evaluieren zu können, wurde neben der Hyperlink-Management-Komponente selbst zusätzlich eine grafische Benutzerschnittstelle entworfen, die ihn als Stand-Alone-Lösung einsetzbar macht. Zu den wichtigsten und umfangreichsten Bausteine der Hyperlink-Management-Komponente zählen ihre Dokumentparser. Durch die objektorientierte Implementation ist der Re-Use der HTML-Parser in den anderen Komponenten gegeben.

4.2.3.1 Klassen/Objekte

Bei der Umsetzung der Konzeption in einem objektorientierten Ansatz wurden eine Reihe von Klassen entworfen. Zu unterscheiden sind diejenigen Klassen, die hauptsächlich der Kapselung der Daten dienen – beispielsweise die Klassen „Document“ oder „Hyperlink“ – von denen, die Funktionalitäten bereitstellen. Zu diesen gehören neben den Implementationen der Schnittstellen, die Methoden zum Lesen und Ablegen der Daten bereitstellen, die Manager-Klassen. Ferner werden basierend auf diesem Framework, die Module aufgebaut, die die interne Verwaltungslogik umsetzen und über Methoden verfügen, die die Hyperlinks aus den Dokumenten herausparsen, sie zwischen den Dokumenten nachführen, neue Hyperlinks vorschlagen etc. In den sich anschliessenden Abschnitten folgt ein Überblick über die wichtigsten Bausteine der Hyperlink-Management-Komponente.

4.2.3.1.1 Datenstrukturen

Im Folgenden sind die allgemeine Datenstrukturen darstellenden Klassen beschrieben, die die Daten-Basis (vgl. Abb. 4.4) der Implementation kapseln:

- *URI*
  Im Gegensatz zu der nativen Java URI bietet diese Klasse lediglich Unterstützung für die Unterscheidung zwischen Dokumenten und Verzeichnissen anhand der URI. Diese basiert auf gängigen Konventionen, die bei Bedarf angepasst werden können. Weiterhin kann sie als Filter auf der Datenbasis eingesetzt werden. Die in Objekten dieser Klasse gesetzten Parameter, erweitert durch Platzhalter, können entweder positiv oder negativ zum Filtern des URI-Bestandes verwendet werden.

- *Peculiarity/ies*
  Die Ausprägung, englisch Peculiarity, ist eine Klasse, die über verschiedene Schlüssel Wert, Zuordnung und Eigenschaften kapselt. Da insbesondere für Dokumente diverse Ausprägungen gleichzeitig bestehen, kapselt die Klasse „Peculiarities“ mehrere solcher Schlüssel-Wert-Paare für den einfacheren Gebrauch. Die letztendliche Zahl und Belegung von Peculiarities wird über die Konfigurationsdateien des Systems geregelt. Im speziellen wurden für die Arbeit mit dem System drei verschiedene Versionierungsparameter eingesetzt: die Sprache, die Version und der Status des Dokumentes. Mit diesen Parametern wird bereits ein Raum aufgespannt, der eine hinreichende Basis für die Bewährung der Hyperlink-Management-Komponente bietet. Insbesondere die Sprache stellt auf Grund der multilingualen Ausrichtung des Systems einen Basis-Parameter dar.

- *PURI*
  Die PURI stellt die Kombination von URI und einem Satz von Peculiarities dar. Pecu-

liarities und URI zusammen liefern den internen Zugriffsschlüssel auf Dokumente und Verzeichnisse.

- *CPURI*
  Die CPURI stellt die URI auf ein Verzeichnis („C“ für Container), identifiziert durch den Satz „Peculiarities“, dar.

- *DPURI*
  Die DPURI ist die Klasse derjenigen Dokumente, die durch URI und Peculiarities bestimmt werden.

- *Meta*
  Die Klasse Meta kapselt die zu einer PURI verfügbaren Meta-Daten. Dazu gehören die Schlagworte in ihrer Häufigkeit genauso wie die Autoren, die an der Erstellung des Dokumentes beteiligt waren. Fernerhin werden Datumsinformationen, z.B. Erstellungs- und Veröffentlichungsdatum, verwaltet.

- *Entry*
  Für den Zugriff auf die Inhalte der Dokumente und Verzeichnisse wurde die Klasse „Ent-

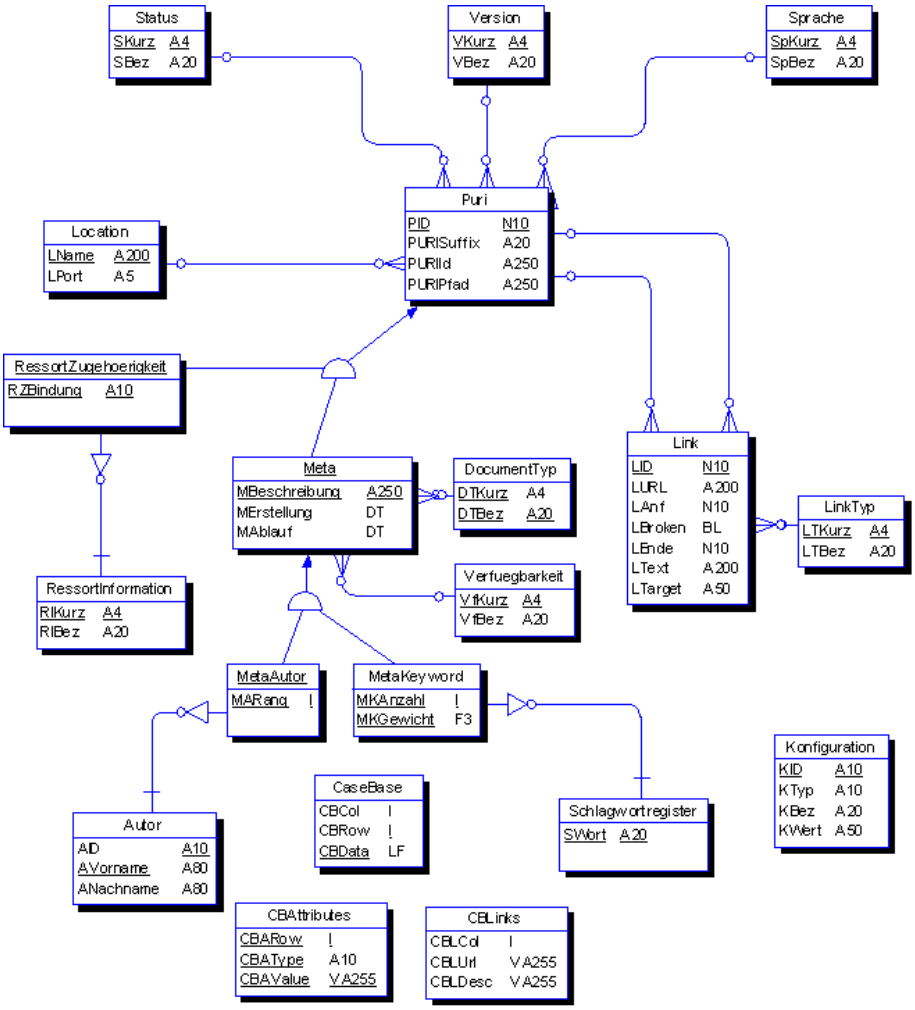

Abbildung 4.4: Ein Überblick über das konzeptionelle Datenmodell des HLM-Moduls.

ry“ entwickelt. Diese bietet entsprechende Ableitungen „Document“ und „Container“, die parametrisiert mit einer PURI, Methoden zum physikalischen Lesen und Schreiben von Daten bereitstellen.

- *Link*
  Gemäß der Zielsetzung der Implementation ist die Link-Klasse eine der umfangreichsten. Neben Ziel- und Ausgangspunkt des Links, beschrieben mittels der oben definierten PURIs mit ihrer entsprechenden Position innerhalb des Dokumentes, werden weitere Parametrisierungen, die den Link charakterisieren, vorgenommen. Diese enthalten z.B. den Status des Links – intern oder extern – sowie die aktuelle Korrektheit.

- *DocAuthor(-liste)*
  Der „DocAuthor“ ist von „Author“ abgeleitet. Neben Namen und Kennzeichen verfügt er über einen Rang, der seine Beteiligung an der Erstellung des Dokumentes relativ zu den anderen Dokumentautoren angibt. Alle Autoren eines Dokumentes werden in der DocAuthorList verwaltet. In Meta wird die DocAuthorList mit den anderen Metadaten zusammengeführt.

- *DocKeyword(-liste)*
  Die Klasse „DocKeyword“ (abgeleitet von Keyword, einer mehr oder weniger einen String kapselnden Klasse) stellt die Beziehung zwischen einem Dokument und einem Schlagwort her. Sie liefert Informationen über die Häufigkeit des Auftretens und der Gewichtung des Schlagworts im Kontext des Dokumente. Da ein Dokument über eine Vielzahl von Schlagworten verfügt, werden diese in einer Liste zusammengefasst. Diese „DocKeywordList“ dient als Container für alle Schlagworte eines Dokumentes und wird in Meta als Schlagwort-Basis übernommen.

4.2.3.1.2 Schnittstellen-Implementationen

Die in der Konzeption beschriebenen Schnittstellen für den Datenzugriff dienen der Abstraktion. Ihre Implementation bietet die Basis, auf der die eigentlichen Funktionen der Hyperlink-Management-Komponente arbeiten. Während sich für die Verwaltung der Hyper-Daten, von der Art der anfallenden Zugriffe her, eine Datenbank basierte Implementation geradezu anbietet, kommt für den Multi-Document Pool sowohl eine dateisystem- als auch eine datenbankbasierte Lösung in Frage. Konkret implementiert wurde die datenbankbasierte HDP-Schnittstelle sowie eine datenbank- und eine dateisystembasierte MDP Schnittstelle. Während die datenbankbasierte Lösung die Verwaltung der Peculiarities vereinfacht und beschleunigt, dafür aber weniger für den Umgang mit den Dokumenten geeignet ist, mussten in der dateisystembasierten Lösung die Peculiarities in Datendateien abgelegt werden. Dafür ist hier der Zugriff auf die Dateien/Dokumente selbst problemloser. Auf eine Kombination beider Lösungen in einer Implementation des MDPI wurde verzichtet, obwohl diese vermutlich die effizienteste Lösung darstellen würde.

4.2.3.1.3 Datastore und Manager

Die auf den Datenstrukturen unter Verwendung der oben beschriebenen Schnittstellen arbeitenden Komponenten liefern die eigentliche Funktionalität des Systems. Anders als bei den Schnittstellen, welche über eine größere Allgemeingültigkeit verfügen und vorrangig der Abstraktion von der zugrundeliegenden Datenablage dienen sollen, handelt es sich bei den im Folgenden beschriebenen Komponenten um native Implementationen, die sich untereinander nicht ohne weiteres separieren lassen und nur im Zusammenspiel eingesetzt werden können. Weiterhin führen sie die Verknüpfung von Hyper-Daten und Dokumenten durch. Die

Aufteilung der Komponenten in separate Klassen erfolgte der Übersichtlichkeit halber und vereinfachte auf diese Weise die verteilte Implementation.

- *Datastore*
  An erster Stelle zu nennen ist das Datastore; diese Klasse stellt die Methoden zum Zugreifen und Ablegen von Hyper-Daten bereit. Bei dem HDP-Datastore handelt es sich um eine native Implementation, die mit den oben beschriebenen Datenstrukturen arbeitet. Insbesondere werden für die im Folgenden beschriebenen Manager die entsprechenden Methoden zum Ablegen und Speichern der benötigten Daten bereitgestellt. Diese sind auf einem durchweg hohen integrativen Niveau, d.h. arbeiten direkt mit komplexen Klassen, z.B. CPURI, DPURI und Link, und erleichtern somit die Implementation der Manager erheblich, da diese dann unmittelbar mit diesen hochwertigen Objekten ihre Aktionen ausführen können.

- *Manager*
  Die Einführung der Manager-Klassen dient lediglich der Abstraktion. Ihre Funktionalitäten liegen zu einem hohen Anteil in dem HDP-Datastore verborgen. Sie liefern definierte Schnittstellen zum Zugriff auf spezielle, höherwertige, da aufbereitete, Daten und erleichtern durch die Trennung in separate Klassen die verteilte Entwicklung.

- *Link-Manager*
  Der Link-Manager ist zuständig für die Speicherung und Abfrage von Hyperlinks. Er bietet Konvertierungen von PURI zu URI und umgekehrt – in den Fällen eines Web-Präsenz-Scans eine zwingend benötigte Funktionalität. Hauptsächlich werden aber von dem Link-Manager Links, die von einem Dokument ausgehen bzw. auf ein Dokument zeigen, geliefert.

- *Meta-Manager*
  Der Meta-Manager übernimmt die Verwaltung von Metadaten zu Dokumenten. Methoden zum Speichern, Löschen und gezielte Abfragen von Meta-Informationen werden von ihm gekapselt. Die Ausführung der eigentlichen Aktionen verbleibt, wie schon in den anderen Fällen, bei dem HDP-Datastore.

- *Similarity-Manager* Basierend auf den Metadaten und Hyperlinks ist der Similarity-Manager in der Lage, durch Anwendung von Case-Based Reasoning-Techniken [HRHM00, HRH+00c] die Ähnlichkeit zwischen Dokumenten zu bestimmen. Seine Methoden werden verwendet, um zu einem Dokument D eine Menge ähnlicher Dokumente aus dem Dokumentenbestand zu identifizieren. Aus dieser Menge werden dann Vorschläge für neue Hyperlinks für das Dokument D generiert.

4.2.3.1.4 Module

Die im nächsten Abschnitt beschriebenen Funktionalitäten des Systems sind in Modulen zusammengefasst, welche die oben beschriebenen Datenstrukturen mit Leben füllen, indem sie z.B. Dokumente parsen. Sie nutzen die Schnittstellen und Manager, um sie zu speichern und auf sie zuzugreifen. Weiterhin stellen sie die Logik bereit, die benötigt wird, um die gewünschten Funktionalitäten liefern zu können.

4.2.3.2 Funktionalitäten

Die für die Hyperlink-Management-Komponente relevanten Funktionalitäten sind in einem Modul, dem HLM (*H*yper*L*ink-*M*anagement)-Modul, zusammengefasst. Um die Komponen-

te, wie gefordert auch als Stand-Alone-Lösung einsetzen zu können, wurde rund um das HLM-Modul ein Programm entwickelt. Eine kommandozeilenbasierte Konsolen-Variante dient dem initialen Einlesen von Daten aus dem Dateisystem der zu bearbeitenden Web-Präsenz. Mit ihr erfolgt der ständige Abgleich zwischen MDP und HDP. Durch Textausgabe auf der Konsole können die einzelnen Funktionen ebenfalls abgerufen werden. Da sich dies verhältnismäßig „unhandlich“ gestaltet, wurde zusätzlich eine Benutzeroberfläche geschaffen, mit der die wichtigsten Funktionen auf dem Datenpool ausgeführt werden können. Analog zu bekannten Darstellungen des Dateisystems in Form einer Baumstruktur stellt die HLM-Benutzeroberfläche auf der linken Seite den Verzeichnisbaum dar und listet auf der rechten Seite die in dem selektierten Verzeichnis aktiven Dokumente auf.

4.2.3.2.1 Linkkonsistenz

Die Hauptfunktion der Hyperlink-Management-Komponente ist die Konsistenzüberwachung der Web-Präsenz hinsichtlich ihrer Verknüpfungsstruktur. Diese Aufgabe gewinnt insbesondere dann an Bedeutung, wenn die Web-Präsenz nicht von einer einzelnen Person, sondern einer größeren Zahl von Mitarbeitern gepflegt werden soll. Das HLM-Modul ist nach dem Einlesen der Web-Präsenz in die beiden internen Datenpools, in der Lage Dokumente zu identifizieren, die Hyperlinks auf nicht existierende Dokumente enthalten. Weiterhin können nicht referenzierte Dokumente gelistet werden. Auf eine graphische Visualisierung der Hyperlink-Struktur wurde verzichtet, die Benutzeroberfläche zeigt jedoch in zwei Tabellen die auf das Dokument zeigenden und die von dem Dokument ausgehenden Hyperlinks an. Defekte Hyperlinks werden von dem System nicht „repariert“, diese Aufgabe obliegt dem für das Dokument zuständigen Bearbeiter.

4.2.3.2.2 Linktransformation/Nachführung

Im Normalfall führt das Verschieben einzelner Dateien von einem Verzeichnis in ein anderes Verzeichnis zu funktionsuntüchtigen Hyperlinks (Broken-Links), so dass die Konsistenz der Web-Präsenz zerstört wird. Durch Nutzung des HLM-Datei-Verschiebe-Mechanismus wird die Konsistenz der Web-Präsenz konserviert. Relativ komfortabel mit Techniken wie „Drag and Drop“ lassen sich Dokumente innerhalb der Web-Präsenz umpositionieren. Die dahinterliegende Logik sorgt dabei für das Nachführen aller betroffenen Hyperlinks: Alle Dokumente, die Links auf das verschobene Dokument enthalten, werden gelesen, modifiziert und anschließend erneut abgespeichert. Desgleichen werden alle Hyperlinks in dem verschobenen Dokument auf analoge Weise nachgeführt.

4.2.3.2.3 Link-Vorschläge

Da der Grad der Verknüpfungen, der zwischen den Dokumenten einer Web-Präsenz vorliegt, als Kriterium für die Güte einer Web-Präsenz betrachtet werden kann, wurde in dem HLM-Modul eine spezielle Hyperlink-Vorschlags-Funktion implementiert. Bereits bei den Managern Erwähnung fand das Case-Based-Reasoning-Verfahren, welches in das HLM-Modul zum Vorschlagen von Hyperlinks integriert wurde. Bei ihm wird der Hypertext in den eigentlichen Text und die enthaltenen Hyperlinks aufgetrennt. Der Text, zusammen mit den verfügbaren Metadaten, wird als das „Problem“ betrachtet, die von diesem ausgehenden Hyperlinks als seine „Lösung“. Eine Auswahl an Dokumenten aus der Web-Präsenz bildet die Fallbasis. Ein neues Dokument wird auf seine Ähnlichkeit zu anderen dem System bekannten Dokumenten bezüglich des Textes und der Metadaten untersucht. Die Hyperlinks, die von den „ähnlichsten“ Dokumenten ausgehen, werden nach Relevanz sortiert als Hyperlink-Vorschläge für die Autoren ausgegeben. Zu der Qualität des Trainingsprozesses und der Linkvorschläge, ausgedrückt

in Recall und Precision, nimmt die Dissertation von Haffner [Haf01] Stellung.

4.2.3.3 Anwendung auf „www.ti.fhg.de“

Sämtliche Funktionalitäten die im Rahmen des HLM-Projektes implementiert worden sind, wurden auf die Web-Präsenz des Instituts für Telematik, die zu diesem Zeitpunkt zum Teil mit DAPHNE und zum Teil manuell verwaltet wurde, angewendet. Die rund 1000 Dokumente, die sich zu diesem Zeitpunkt auf der Web-Präsenz befanden, haben von dem Link-Management profitiert. Neben der Identifikation einer Zahl von Inseldokumenten, die über keinerlei Referenzierung mehr verfügten und deshalb von den Benutzern nur bei Kenntnis der genauen URI abgerufen werden konnten, wurden ebenso in vielen Dokumenten „ins Leere“ verweisende Hyperlinks korrigiert.

Während bei dem von DAPHNE verwalteten Dokumentenbestand eine flache Verzeichnisstruktur vorherrschte und bis auf die beschriebenen Fälle kaum Vorteile aus dem System gezogen werden konnten, stellte sich die Lage bei dem manuell verwalteten und in Verzeichnisstrukturen abgelegten Dokumentenbestand anders dar. Hier konnte die Hyperlink-Nachführung bei der Verschiebung von Dokumenten innerhalb der Verzeichnisse hilfreich eingesetzt werden. Ebenfalls als eine wertvolle Ressource zur Evaluation des Systems erwies sich die Web-Präsenz des Instituts für Telematik bei der Erprobung der Hyperlink-Vorschlag-Algorithmen. In der Arbeit von Haffner [Haf01] finden sich zu diesem Punkt ausführliche Vergleiche aus dem Einsatz des Systems resultierend. Insbesondere komparative Darstellungen mit für andere Web-Präsenzen generierten Linkvorschlägen bieten die Möglichkeit zur Abschätzung des Potentials dieses Systems.

4.2.3.4 Zusammenfassung

Mit der Hyperlink-Management-Komponente wurde eine für die Web-Präsenz-Verwaltung äusserst wichtige Komponente entwickelt. Von ihrer Konzeption her lässt sie sich sowohl unabhängig auf bestehende Web-Präsenzen anwenden, als auch als Baustein für ein Komplettpaket nutzen. Durch die Auslegung mit dem Dateisystem der Original-Web-Präsenz als „Master“ handelt es sich im praktischen Einsatz um ein sehr robustes System. Wie auch zuvor können die Autoren, bzw. Verantwortlichen die Web-Präsenz nach Wunsch auch ohne das System modifizieren, ohne dass die Hyperlink-Management-Komponente involviert werden muss. Anschließend wird lediglich ein neues „Scannen“ der Web-Präsenz bzw. der betroffenen Dokumente notwendig. Das System passt daraufhin seinen Datenbestand an.

Die Erfahrung hat gezeigt, dass ohne ein solches flexibles Vorgehen der Akzeptanzgrad für ein derartiges System bei den Benutzern deutlich sinkt. Insgesamt sind die mit dem System erzielten Resultate zufriedenstellend und prädestinieren es für den produktiven Einsatz. Mit JDaphne findet sich ein ebenfalls in Java entwickeltes System, bei dem sich die Integration dieser Komponente geradezu aufdrängt. Durch den DAPHNE-üblichen Einsatz von Dokumentnummern als Dateinamen, bei JDaphne nur für den internen Gebrauch eingesetzt, gestaltet sich die Nutzung des durch die Verwaltung von Dokumentausprägungen erzielbaren Vorteils für die Hyperlinks (Mehrdimensionalität mit „Sprache“ als Dimension) allerdings als schwierig. Direkten Mehrwert bietet das System hingegen bezüglich der verschiedenen Repositories für die Dokumente mit unterschiedlichem Status und für die Versionierung von Dokumenten wie sie bei JDaphne verwendet werden.

4.3 Mehrwert durch Assistenten- und Gateway-Komponenten

Eine weitere Komponente, die als Baustein bei der Entwicklung von JDaphne betrachtet werden kann, ist die Gateway-Komponente. Ihr liegt der Gedanke zugrunde, eine Anwendungskomponente in die Kommunikation zwischen Web-Browser auf der einen und Web-Server auf der anderen Seite einzuschleifen. Ziel der Komponente ist die Web-Browser-unabhängige Bereitstellung von Mehrwertdiensten. Mehrwertdienste sind dabei einerseits Funktionalitäten, die im Web-Browser für die Nutzung des WWW sinnvoll und hilfreich wären (vgl. [RC99, HM99a]), aber nicht implementiert sind. Andererseits lassen sich damit über Dritte Dienste anbieten, die helfen Probleme, allgemeiner Art, wie z.B. die fehlende Persistenz von Online-Referenzen (vgl. [HLM00b]), zu lösen. Für JDaphne wertvoll an dem Gateway-Konzept sind die dort implementierten Funktionalitäten zum Aufbereiten und Sichern der über den Gateway vermittelten Daten. Datenstrukturen zur Verknüpfung von Inhalten sowie die Versionierung von Dokumenten und Hyperlinks über die Zeitschiene hinweg mit einer grafischen Aufbereitung des Navigationspfades sind sinnvoll nutzbare Konzepte.

Auch wenn die Gateway-Komponente nicht als direkter Bestandteil von JDaphne in das System integriert ist, können seine Nutzer von JDaphne durch das System einen realen Mehrwert erfahren. Besonders bei der gezielten Recherche zu bestimmten Themengebieten und bei der Organisation bzw. dem Management von aus dem Internet geladenen Dokumenten ist die derzeitige Unterstützung im Standard-Web-Browsern bei weitem noch nicht hinreichend. Unterstützung kann im einfachsten Fall z.B. in dem automatischen Sichern der auf die lokale Plattform geladenen Dokumente bestehen, wie dies die Web-Browser zu Caching-Zwecken temporär praktizieren. Auch das Speichern sämtlicher in ein Dokument eingebundene Teildokumente (z.B. Grafiken) erleichtert die Arbeit deutlich. Das Visualisieren der Dokumentenhierarchie einer Web-Site sowie der gezielte, einfache Zugriff auf die in vorherigen Sitzungen geladenen Dokumente kann eine nicht unerhebliche Erleichterung beim Navigieren und (Wieder-)Auffinden von Ressourcen darstellen. Durch das Kommentieren und die Verschlagwortung entsteht ein wertvolles, lokales Archiv.

Alle diese Zusatzfunktionen, die den Umgang mit der gewaltigen Ressource WWW erleichtern, können entweder in die gängigen Programme integriert oder als Zwischenlösung auf dem Weg bis zur Integration als externe Funktionen bereitgestellt werden. Im letzteren Fall, der den weiteren Vorteil hat, dass diese Zusatzfunktionen bzw. die daraus resultierenden Daten auch von anderen Programmen genutzt werden können, wird die Gateway-Komponente im weiteren als Assistenten-System bezeichnet (vgl. "Web Browser Intelligence"[BMK97]), das platziert neben dem Web-Browser dem Benutzer die Arbeit mit dem WWW erleichtert.

4.3.1 Technisches Konzept

Assistenten-Systeme müssen, um dem Benutzer sinnvoll assistieren zu können, die Kommunikation, die zwischen dem Web-Browser und den Web-Servern im Internet stattfindet, kennen. Gegebenenfalls müssen sie weiterhin auch in die Kommunikation eingreifen und den Datenverkehr modifizieren. Dem Konzept des Assistenten-Systems liegt daher das Einschleifen [BM98] einer eigenständigen Komponente zwischen dem Web-Browser und den Web-Servern im Internet zugrunde (Abb. 4.5). Diese Komponente, im weiteren als Gateway bezeichnet, führt den Datentransfer zwischen dem Web-Browser und den Web-Servern im Internet durch. Damit sind beide Anforderungen an das Assistenten-System prinzipiell erfüllt. Da es die Daten selbst transferiert, kennt es die Kommunikation zwischen Web-Browser und Web-Servern. Weiterhin ist es in der Lage, die Daten vor dem Transfer in beiden Richtungen zu modifi-

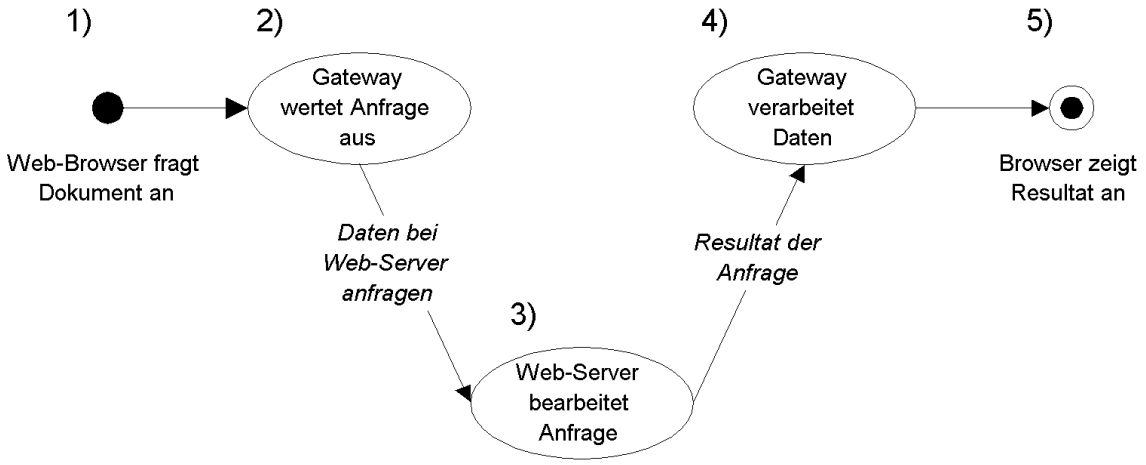

Abbildung 4.5: Grundprinzip eines Gateways.

zieren. Die im Gateway verfügbaren Daten über die Kommunikation zwischen Web-Browser und Internet können in einer zweiten Komponente (Abb. 4.6), z.B. einer Datenbank oder einem Datei-System, in aufbereiteter Form abgelegt werden. Diese Komponente hat zum einen die Aufgabe, die aufbereiteten Daten innerhalb des Assistenten-Systems verfügbar zu halten, zum anderen kann sie auf einfache Art und Weise auch dem Austausch von Daten mit anderen Systemen dienen. Für die Darstellung der aufbereiteten Daten innerhalb des

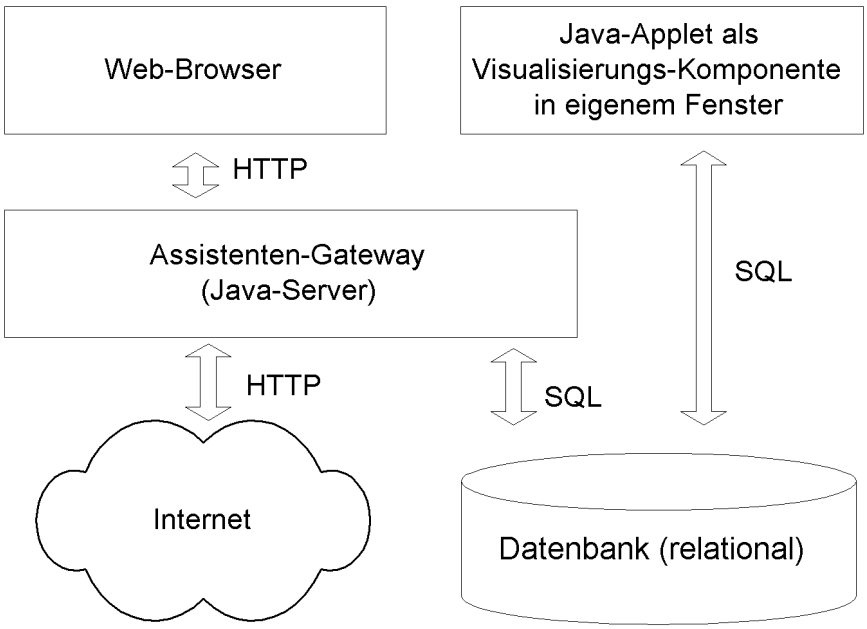

Abbildung 4.6: Der Web-Browser-Assistent. Komponenten und Aufbau.

Assistenten-Systems gibt es zwei Varianten: einerseits kann der Gateway selbst die aufbereiteten Daten verwenden, um transferierte Daten zu modifizieren oder zu ergänzen. Durch speziell an den Gateway gerichtete Anfragen kann er aber auch gezielt diese transferierten

Daten für den Web-Browser aufbereiten. Andererseits sieht das Konzept eine dritte Komponente (Abb. 4.6) vor, die unabhängig vom Web-Browser die Darstellung und eventuell eine Weiterbearbeitung der aufbereiteten Daten übernimmt. Diese Komponente stellt die eigentliche Benutzerschnittstelle des Assistenten-Systems dar. Mit ihr kann ggf. auch die Funktion des Gateways gesteuert werden. Ein wichtiger Punkt des Konzeptes ist die freie Wahl der Platzierung der Gateway-Komponenten. Möglich ist die Platzierung als lokaler Gateway, der dem Benutzer in Form eines Assistenten-Systems einen Mehrwert bietet; in diesem Fall benutzt ausschließlich der Nutzer am lokalen Arbeitsplatz die Möglichkeiten des Systems für den eigenen Gebrauch. Um die Funktionalitäten innerhalb z.B. einer Arbeitsgruppe oder eines Unternehmens zu nutzen, ist eine lokale Installation im Intranet die geeignete Variante. Um als Dienstleister internetweit Mehrwertdienste anbieten zu können, ist eine Platzierung im Internet bei der registrierte oder unregistrierte Benutzer den Dienst abfragen können, geeignet. Schließlich lässt sich als letzte Variante der Gateway direkt vor dem Web-Server platzieren. Bei dieser Konfiguration gehen alle Anfragen an den Web-Server durch den Gateway, so dass dieser benutzerbezogen die Anfragen der Clients und die Antworten des Web-Servers in geeigneter Art und Weise modifizieren und aufbereiten kann.

4.3.2 Anwendungsszenarien – Datenverarbeitung durch den Gateway

Wie schon erwähnt, liegt eine der Hauptanwendungen des Gateway-Konzepts in den Assistenten-Systemen. Die in diesem Bereich erforderliche Aufbereitung von Dokumenten findet durch flexible Parse-Mechanismen statt. Hauptanforderung in diesem Bereich ist das Extrahieren von Hyperlinks und das entsprechende Umsetzen auf andere, z.B. lokale Ziele. Diese Anforderungen sind analog zu der Problematik im Redaktionssystem-Umfeld; auch hier sind die Verknüpfungen der Dokumente aus implementatorischer Sicht einer der Hauptpunkte. Weiterhin stellt die Verschlagwortung zum Wiederauffinden von Dokumenten sowohl im Redaktionssystem-Umfeld als auch bei den Browser-Assistenten eine wichtige Aufgabe dar. Die Dokument-Parser werden dementsprechend für beide Bereiche gleichermaßen benötigt. Die Art und Weise, in der die Daten von den entsprechenden Umgebungen verwaltet werden, unterscheidt sich in Details. Da die Implementation von JDaphne, der Hyperlink-Management-Komponente und der Browser-Assistenten in Java erfolgte, ist die Wiederverwendung der Parse-Funktionalitäten abgesehen von kleineren Anpassungen an die speziellen Umstände problemlos möglich. Die Implementation von Signatur-Konzepten im Rahmen des Redaktionssystems sowie die exemplarische Anbindung an eine PKI findet ihr Pendant im übrigen in der Implementation des „Signing-Gateway“. Der Gateway wird sowohl von dem Web-Browser als auch von den Web-Servern im Internet als Proxy [Luo98] wahrgenommen. Proxies sind Programme, die nach bestimmten Regeln Dokumente, die aus dem WWW häufig angefordert werden, lokal zur schnelleren Bereitstellung zwischenspeichern. Je nach Konfiguration hält ein Proxy eine lokale Sammlung von Dokumenten vor, die er dem Benutzer auf Anfrage sehr schnell präsentieren kann. Nur in bestimmten Fällen, bei der Verwendung spezieller HTTP-Anfragen, speichert der Proxy die Dokumente nicht zwischen und fordert sie direkt von den ursprünglichen Web-Servern an. Normalerweise gibt sich der Gateway den Web-Servern gegenüber als Proxy zu erkennen, aber nur in wenigen Ausnahmefällen wird sich dadurch das Verhalten der Web-Server ändern. Web-Präsenzen, die kein Interesse haben, einen Proxy zu bedienen, werden dies auch bei dem Gateway nicht anders handhaben. In solchen Fällen muss der Benutzer auf den Mehrwert durch das Assistenten-System verzichten und sich direkt, d.h. ohne den Umweg über den Gateway, mit der Web-Präsenz im Internet verbinden. Im Normalfall wird durch den Gateway die Kommunikation zwischen Web-Browser und Web-Servern jedoch nicht behindert. Weil der Gateway die gesamte Kom-

munikation zwischen Web-Browser und Web-Servern übermittelt, hat er sämtliche Optionen zum Verarbeiten der transferierten Daten.

Ausgangsbasis für jedwede Zusatzfunktionalität, die dem Benutzer geboten werden kann, ist die Datenverarbeitung im Gateway. Grundsätzlich können sechs verschiedene Varianten dieser Datenverarbeitung im Gateway unterschieden werden. Der einfachste und nur der Form halber zu nennende Fall ist der Datentransfer ohne irgendeine Aktion: Der Gateway verwaltet lediglich die Kommunikation zwischen Web-Browser und Web-Server. Von Wert hinsichtlich der Bereitstellung von Zusatz-Funktionalitäten sind die Varianten: Überwachung, Prefetching, Filtern, Modifizieren sowie Aufzeichnen und Archivieren des (aufbereiteten) Datentransfers. Dabei ist durchaus die Verkettung mehrerer der genannten Varianten anzustreben. Im Folgenden werden die Varianten im einzelnen hinsichtlich ihrer Anwendbarkeit erläutert. Dabei ist zu beachten, dass jede der Varianten sowohl auf die Anfrage, welche vom Web-Browser zum Web-Server gesandt wird, als auch auf die entsprechende Antwort des Web-Servers, angewandt werden kann. Je nach Bedarf kann beides gleichzeitig erfolgen. Dabei ist jedoch zu berücksichtigen, dass nicht alle Dokumente, die vom Web-Server an den Web-Browser zurückgesandt, werden in gleicher Art und Weise für die Weiterbearbeitung geeignet sind. Während Text-Dokumente verhältnismäßig einfach bearbeitet werden können, ist dies bei binären Dateien, z.B. Grafiken, zumindest den Inhalt betreffend, deutlich schwieriger und in der hier beschriebenen Version des Gateway nicht möglich.

4.3.2.1 Überwachung

Da der Gateway die gesamte Kommunikation zwischen Web-Browser und Web-Server kennt, kann er ohne weiteres als Überwachungskomponente dienen. Bei einer geeigneten Visualisierungskomponente kann so dem Benutzer sichtbar gemacht werden, welche Verbindungen sein Web-Browser gerade geöffnet hat. Während die Web-Browser nur sehr undifferenziert anzeigen, wieviele Daten sie gerade aus dem Internet empfangen, kann dies mittels der Visualisierungskomponente sehr detailliert für jedes einzelne Dokument dargestellt werden. Der Benutzer bekommt so klar gezeigt, welche Datenmengen für von ihm angefragte Dokumente übertragen werden müssen. Anhand der Darstellung der offenen Verbindungen kann der Benutzer direkt ersehen, welche Verbindungen seitens des Browser nicht korrekt geschlossen wurden. Speziell in solchen Fällen, aber auch in anderen Situationen kann der Benutzer einzelne Verbindungen aus dieser Liste auswählen und gezielt schließen.

Eine geeignete Konfiguration des Gateways kann auch Teile der Funktionalität einer „Firewall“ abbilden. In diesem Aufbau überprüft der Gateway, ob die Verbindung zu dem vom Browser gewünschten Rechner geöffnet werden darf. Verbietet die Einstellung des Gateway den Aufbau der Verbindung, so wird dem Browser eine entsprechende Meldung übermittelt. Im Gegensatz zu einer Firewall bietet der Gateway aber nur einen stark verminderten Sicherheitstandard. Er überwacht nämlich nur diejenigen Verbindungen (Protokolle), die im Web-Browser konfigurierbar sind. Baut eine aktive Komponente, z.B. ein Applet, selbstständig eine Verbindung zu einem anderen Rechner auf, so ist ein Eingreifen des Gateway ausgeschlossen. Dennoch ist es durchaus sinnvoll, an zentraler Stelle Verbindungen des Web-Browsers/der Web-Browser zu blockieren. Während solche Funktionen lokal vom Benutzer zumeist auch im Web-Browser selbst konfiguriert werden können, ist es bei der gleichzeitigen Benutzung verschiedener Browser (Typen) günstiger, die Einstellungen zentral am Gateway zu verwalten. Gerade beim Einsatz des Gateway für verschiedene Benutzer, z.B. in einer Abteilung, ist diese gemeinsame Konfiguration viel einfacher zu verwalten. Auch wenn die Funktionalität einer Firewall bei weitem nicht erreicht wird, kann so auf einfache Art und Weise verhindert werden, dass sich der Web-Browser automatisch mit bestimmten Web-Präsenzen verbindet.

Mittels der Überwachung ist es ebenso ohne weiteres möglich, Statistiken über Internet-Zugriffe zu erstellen. Diese Statistiken, gewonnen aus Log-Dateien oder mittels anderer Zählverfahren, können dem Benutzer bei Bedarf Aufschluss über sein Browse-Verhalten geben. So können gezielt einzelne Sitzungen ausgewertet werden. Über längere Zeit hinweg betrachtet können die Statistiken wertvolle Hinweise bezüglich der Optimierung seines Verhaltens liefern. Handelt es sich um einen persönlichen Gateway, lässt sich auf Basis dieser Daten eine einfache Prefetching-Funktionalität des Gateway realisieren. Werden die Statistiken dem Benutzer dagegen für jedes Dokument beim Laden visualisiert (z.B. einfach im Kopf des Dokumentes, s.u.), kann er aus der Abrufhäufigkeit eines Dokumentes dessen Relevanz für seine Arbeit ableiten.

4.3.2.2 Prefetching

Unter Prefetching wird eine Funktionalität gefasst, die Dokumente aus dem Internet sozusagen auf Vorrat im voraus anfordert. Wenn der Browser dann auf das Dokument zugreifen möchte, muss es nicht mehr aus dem Internet angefordert werden, sondern ist bereits lokal verfügbar. Dies beschleunigt den Zugriff auf Dokumente erheblich. Anhand einer verlässlichen Statistik ist es möglich, den Gateway mit einer solchen Prefetching-Funktionalität auszustatten. Basierend auf einer detaillierten Statistik, die Auskunft darüber gibt, wann und wie häufig ein Dokument in der Regel geladen wird, kann die Ladezeit verkürzt werden, wenn der Gateway dieses Dokument bereits kurz vorher aktualisiert hat. Dieser einem Proxy sehr ähnliche Dienst kann entweder basierend auf den Statistiken oder vom Benutzer selbst spezifizierten Angaben betrieben werden. Insbesondere ist hier zu beachten, dass sich mittels eines solchen Prefetching-Mechanismus' für den Benutzer auf ad hoc eine vollkommen andere Situation ergeben kann. Er erlebt nämlich bei geeigneter Visualisierung der automatisch geladenen Dokumente so etwas wie einen Push-Vorgang, eine Art Abonnement. Ihm wird analog zu einem Mail-Eingang mitgeteilt, wenn ein neues Dokument verfügbar ist. Diese Funktion wird um so wertvoller, wenn abgesehen vom Prefetching zusätzlich ein Dokumenten Abgleich stattfindet. So kann der Benutzer im speziellen nur dann auf das neue Dokument aufmerksam gemacht werden, wenn es sich gegenüber der ihm bekannten Version signifikant geändert hat.

Ein anderer Prefetching-Mechanismus basiert auf den in den transferierten Dokumenten enthaltenen Links. Ein geeigneter Parse-Mechanismus extrahiert diese Links und gibt dem Gateway die Möglichkeit, die verlinkten Dokumente bereits im voraus anzufordern, vgl. [HRH+00c]. Gegebenenfalls können Regeln dem Gateway vorgeben, in welcher Reihenfolge er die Dokumente anfordern soll. Beispielsweise können Dokumente, die auf dem gleichen Web-Server gelagert sind wie das Dokument, das den Link enthält, zuerst angefordert werden. Dies würde die Annahme voraussetzen, dass Dokumente, die innerhalb der Web-Präsenz liegen, den Benutzer vorrangig interessieren.

4.3.2.3 Filtern

Unter dem Begriff Filtern wird das gezielte Entfernen von Daten aus der Menge der insgesamt transferierten Daten zusammengefasst. Offensichtlich geht mit dem Filterprozess auch eine Modifikation der übertragenen Daten einher. Diese ist aber im Gegensatz zu der unten beschriebenen Variante auf das Entfernen von Daten beschränkt. Dieses bezieht sich dabei sowohl auf die Anfrage durch den Browser als auch auf die Antwort, welche der Web-Server liefert. Der Filtermechanismus kann auf der Anfrageseite z.B. dazu eingesetzt werden, vor dem Web-Server bestimmte Informationen zu verstecken, die der Web-Browser bereitwillig preisgibt. Unter anderem kann es dem Benutzer aus Sicherheitsgründen wichtig sein, dem Web-Server nicht mitzuteilen, auf welcher Betriebssystem-Plattform sein Web-Browser ar-

beitet. Solche der Anonymität und somit der Sicherheit beim Browsen im Internet dienenden Filter können durch weitere, die auf die Anfrageresultate angewendet werden, ergänzt werden. Bei den Antworten der Web-Server kann es ebenfalls geeignete Ansatzpunkte für Filter geben. Während unabhängig vom Datenformat der übermittelten Daten der Antwortkopf in jedem Fall gefiltert werden kann, ist bei der Übertragung von binären Daten im Anfragekörper dem Filtermechanismus letztendlich eine kaum überwindbare Grenze gesetzt. Bei den Daten, die aus dem Dokument selbst herausgefiltert werden können, gibt es ein breites Spektrum. Aus Datentransferüberlegungen (Bandbreite) heraus ist es effizient, wenn auf Werbe-Banner gerichtete Links aus dem Dokument entfernt werden. Durch Sicherheitsbedenken motiviert sein kann das Entfernen oder auch lediglich das Überprüfen von aktiven Dokumentinhalten. Bei diesen kann es sich z.B. um Java-Skript-Code handeln. Während sich das Entfernen von solchen Dokumentinhalten relativ einfach bewerkstelligen lässt, treten bei der Evaluierung des eingebetteten Codes teilweise unlösbare Probleme auf, die eine sinnvolle Klassifizierung nahezu unmöglich machen.

4.3.2.4 Modifizieren

Das Modifizieren der übertragenen Daten ist im Vergleich zu dem reinen Filtervorgang deutlich anspruchsvoller. Offenkundig können sowohl die Anfragen des Browsers als auch die Antworten der Web-Server modifiziert werden. Bei der Modifikation der Anfrage stehen dem Benutzer mit dem Gateway sämtliche Optionen offen: Wie beim Filtern können einzelne Parameter des Anfragekopfes entfernt werden, gegebenenfalls können sie jedoch auch durch gezielte Fehlinformationen ersetzt werden. Diese der Anonymisierung dienende Möglichkeit kann zusätzlich noch dazu verwendet werden, gezielt die Beziehung zwischen Anfrage und zurückgeliefertem Dokument zu evaluieren. Beispielsweise unterstützen viele Web-Server die Funktion „Content-Negotiation“ bezüglich der bevorzugten Dokumentensprache. Oft ist aber nicht die Einstellung der entsprechenden Sprache im Browser relevant, sondern die Sprache der Web-Browser-Version ausschlaggebend. Solche und ähnliche Modifikationen lassen sich mittels des Gateway einfach vornehmen.

Bei vielen aktuellen Web-Präsenzen werden die Dokumente nicht mehr statisch vorgehalten, sondern aktuell für den Benutzer z.B. aus einer Datenbank generiert. Dabei wird häufig versucht, dem zustandslosen HTTP eine Sitzung, etwas Zustandsbehaftetes, hinzuzufügen. Dies geschieht, indem die Links in dem Dokument mit einer Sitzungsinformation versehen werden. Diese Informationen bestehen häufig aus Buchstaben- und/oder Ziffernkombinationen, die in die URI eingebaut oder an die URI angehängt werden. Interessanter als die Modifikation der Anfrage ist in den meisten Fällen die Modifikation der Antworten der Web-Server. Während erstere der Anfrage verhältnismäßig eingeschränkt sind, kann die Antwort in vielfältigster Form modifiziert werden. Wie bei der Filter-Funktionalität beschrieben, kann es sinnvoll sein, Werbe-Banner auszublenden. Durch geeignete Ersatztexte oder -bilder lässt sich ein für den Benutzer nach wie vor voll funktionsfähiges Dokument realisieren. An der Position der ausgeblendeten Werbe-Banner sieht er dann z.B. jeweils ein Symbol, welches ihm anzeigt, dass dieser Platz zuvor zu Werbezwecken benutzt wurde. Bei Bedarf kann er auf das Symbol klicken und bekommt dann dennoch das entsprechende Banner angezeigt. Mit etwas Aufwand auf der Seite des Gateway kann das Werbe-Banner, wie im Originalzustand, mit einem Link, der zu dem Werbe-Treibenden führt, versehen werden.

Die oben erwähnte statistische Information, die der Gateway über die übertragenen Dokumente bereitstellen kann, können entweder in einer separaten Benutzerschnittstelle angezeigt werden oder in das übertragene Dokument (sofern es sich um ein simples textuelles Format, z.B. HTML, handelt) eingefügt werden [HM99b]. Alternativ können Links auf separat abrufbare Statistiken in dem Dokument platziert werden. Da der Gateway auch in der Lage ist,

Links umzusetzen, können in die bereits im Dokument enthaltenen Links auch statistische Daten über die referenzierten Dokumente eingebaut werden. So kann der Benutzer vor dem Abrufen eines verlinkten Dokuments ersehen, ob und wenn ja, wie häufig er das Dokument bereits aufgerufen hat, wie groß es war etc. Unter Einbindung von anderen, eventuell externen Diensten, wie z.B. Top-Blend [CDHV00], können dem Benutzer die Modifikationen visualisiert werden, die das Dokument seit der letzten Ansicht erfahren hat. Neben dem Einblenden statistischer Informationen sind auch Aktionen wie das Einfügen von Übersetzungen oder das Ersetzen von Fremdwörtern denkbare Dienste.

4.3.2.5 Aufzeichnen und Archivieren

Neben den Filter- und Modifikationsvorgängen stellt insesondere die Aufzeichnung und Archivierung der übertragenen Daten eine wertvolle Ausgangsbasis für Assistenten-Systeme dar. Geeignet aufbereitet können die Daten für die sofortige Visualisierung oder den späteren Zugriff abgelegt werden. Dabei sind neben den übertragenen Dokumenten auch die Protokoll-Informationen von großer Bedeutung. Sie können eine Analyse des Navigationsvorgangs genauso ermöglichen, wie die einfache Reproduzierung einer Sitzung. Die übertragenen Dokumente können je nach Dokumentenformat unterschiedlich für die Archivierung aufbereitet werden, wobei dies für Text-Dokumente deutlich besser realisierbar ist, weil aus dem Text über den Dokumententyp hinausgehende Informationen gesammelt werden können. Dies können z.B. Metadaten sein, die in dem Dokument zur Beschreibung enthalten waren. Aber auch das Extrahieren von Schlagworten und, speziell bei HTML-Dokumenten, von Hyperlinks ist bei ihnen möglich. Zusammen mit dem Dokument abgelegt ist so später ein gezielter Zugriff auf einzelne Dokumente erlaubt. Je intensiver die Archivierung von übertragenen Daten betrieben wird, desto größer wird das anfallende Datenvolumen. Unabhängig von der Art der Speicherung und Indexierung kann allein die Datenmenge einen hinreichend schnellen Zugriff auf Daten verhindern. Aus diesem Grund muss das System entweder auf globale Speicherung aller Daten verzichten und nur bei Bedarf durch den Benutzer aktiviert oder die Stammdatenbasis durch die geeignete Auslagerung von Daten möglichst klein gehalten werden.

4.3.3 Prototypische Implementationen

Basierend auf obiger Konzeption wurde in Java eine prototypische Implementation einer Gateway-Komponente durchgeführt. Sie wurde in zwei verschiedenen Ausrichtungen aufgebaut: in der einen Variante als Mehrwertdienst im Sinne eines elektronischen Notars (vgl. [DH89]), in der anderen Variante als Assistenten-System zur Unterstützung bei der Recherche im Internet. Beide Varianten basieren auf einer analogen Server-Klasse, die, als Web-Proxy angesprochen, die Daten zwischenverarbeitet, bevor sie weitergeleitet werden.

4.3.3.1 Signierender Gateway

Das Ziel des signierenden Gateway [HLM00a] ist die Beglaubigung von aus dem Internet geladenen Dokumenten durch den Betreiber des Gateway als vertrauenswürdigem Dritten. Für die Beglaubigung der elektronischen Dokumente transferiert der Benutzer diese nicht direkt von dem eigentlichen Web-Server, sondern fordert sie über den Gateway (Abb. 4.7) an. Dieser sendet die Anfrage an den Web-Server weiter. Das Ergebnis der Anfrage wird bei dem Gateway in einen Zwischenspeicher gelegt und anschliessend an den Benutzer weitergeleitet. In einem zweiten Schritt kann sich der Benutzer mit seinem Web-Browser an einem speziellen Port mit dem Gateway verbinden. Dieser arbeitet in diesem Moment als eine Art Web-Server und bietet dem Benutzer eine Maske an, mit der er auswählen kann, welche der

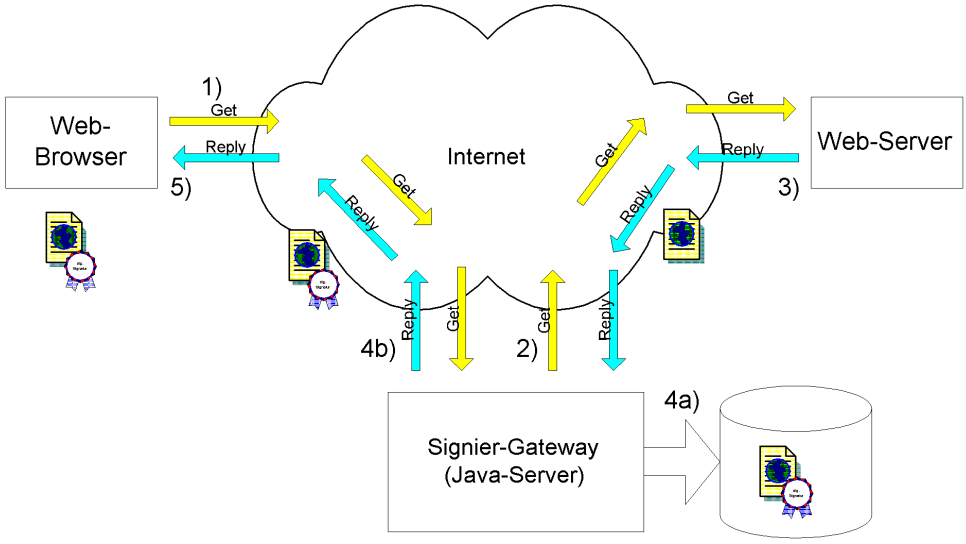

Abbildung 4.7: Signierender Gateway als beglaubigende Instanz.

übertragenen Dokumente er beglaubigt bekommen möchte. Diese werden vom Gateway aus dem Zwischenspeicher geladen und in ein Java-Archiv [Eck00] verpackt. Dieses wird vor dem Versand an den Benutzer mit dem privaten Schlüssel des Gateway-Betreibers signiert. Das Java-Archiv mit den signierten Dateien und der Gateway-Signatur wird an den Benutzer übermittelt. Dieser hat nun eine Datei in der Hand, deren Herkunft elektronisch von einer dritten Partei, im Normalfall einer vertrauenswürdigen Instanz, bestätigt wird.

Die Implementation des signierenden Gateway erfolgte vollständig in Java, wobei die Gateway-Komponente als eigenständige Server-Komponente betrieben wird. Um eine standardkonforme Lösung zu entwickeln, wurden die Java-Archive als Container für die zu signierenden Dateien ausgewählt. Diese Entscheidung wurde auf Grundlage von zwei Vorteilen gefällt, die dieses Format bietet:

1. Java-Archive sind direkt von „Zip“-Dateien abgeleitet. Sie sind darauf ausgelegt, viele Dateien zu einer großen zusammenzufassen und bieten zusätzlich Kompressionsmöglichkeiten. Da es sich bei dem Zip-Format um einen Standard handelt, kann jeder Rechner ohne Probleme in die Lage versetzt werden, diese Archive zu entpacken.

2. Java-Archive sind darauf vorbereitet, signiert zu werden (Code-Signing). Dementsprechend gibt es mit dem „Jar-Signer“ ein frei verfügbares Werkzeug, mit dem digital signierte Archive verifiziert werden können.

Weiterhin bietet Java die Möglichkeit, auch bei der Signatur mit RSA und MD5 (vgl. Seite 53) und X.509-Zertifikaten standardkonforme Anbindungen aufzubauen. In der konkreten Implementation des hier beschriebenen Gateway enthalten die Java-Archive für jede transferierte Datei noch vier zusätzliche Dateien, die ebenfalls signiert werden. Bei ihnen handelt es sich um die zwischen dem Gateway und dem Web-Server ausgetauschten HTTP-Header. Durch sie wird nachvollziehbar, welche Anfrage bei dem Web-Server zu dem signierten Dokument geführt hat. Sofern ein Zeitstempeldienst für den Gateway zur Verfügung steht, kann ein Hash-Wert der zu signierenden Datei zusätzlich mit einem solchen Zeitstempel versehen werden. Dem Benutzer ist es dann möglich, nachzuweisen, wann er von dem Web-Server ein

bestimmtes Dokument unter einer bestimmten URI auf seine Anfrage hin erhalten hat – ein Dienst, der im Internet zu mehr Verbindlichkeit führen kann.

Ein entsprechendes Szenario für eine Anwendung dieses bislang prototypisch implementierten Dienstes kann in dem Einsatz für die Konservierung von Online-Referenzen [HLM00b] liegen; bei der Erstellung einer wissenschaftlichen Arbeit wird immer stärker die Ressource Internet eingesetzt, um Informationen zu bestimmten Themengebieten zu recherchieren. Bestätigt wird dies in der Untersuchung von Lawrence [Law01], die belegt, dass online verfügbare Artikel häufiger zitiert werden als nur offline verfügbare. Das Problem mit Online-Referenzen ist deren temporärer Charakter. Experimente wie das „Persistent URL"-Projekt (PURL) [Sha96] oder der WebLinker [AHK+95] offenbaren die Notwendigkeit von persistenten Online-Referenzen. Während Ansätze, Broken-Links im Allgemeinen zu vermeiden, wie der der W3Objects [ICL96], an der unüberschaubaren Anzahl online-verfügbarer Dokumente scheitern müssen, stellt der hier vorgestellte Ansatz des „Signierenden Gateway" eine wahre Alternative dar. Nach der Klärung des Copyrights ist es möglich, parallel mit dem publizierten Artikel in Addition die online-referenzierten Artikel analog zu Grafiken in signierter Form zu veröffentlichen. Damit verweist die Online-Referenz neben der ursprünglichen URI auf das von einer vertrauenswürdigen Partei beglaubigte, zum Zeitpunkt der Erstellung der Arbeit vorliegende Dokument. Auf diese Weise können zwei Probleme behoben werden: zum einen ist es immer möglich, das Dokument zu lesen, auf das ein Artikel verweist, auch wenn dieses mittlerweile nicht mehr unter der URI abzurufen ist, unter der es ursprünglich lag, zum anderen kann über die beglaubigte Kopie ermittelt werden, ob sich das referenzierte Dokument seit der Referenzierung in seinem Inhalt geändert hat – ein Problem, das vor allem bei Verweisen auf nichtwissenschaftliche Artikel, z.B. in Online-Zeitungen, besteht. Ausschlaggebend bei der Nutzung des Beglaubigungsdienstes ist dabei die Integrität der von dem Web-Server angeforderten Daten. Während für den wissenschaftlichen Einsatz auch mit offenen, d.h. nicht verschlüsselten Verbindungen zwischen Gateway und Web-Server gearbeitet werden kann, sind für den kommerziellen Einsatz solcher Konzepte „sichere Verbindungen" unabdingbar. In diesem Fall ist eine Erweiterung der Gateway-Komponente um sichere Verbindungen unerlässlich.

4.3.3.2 Recherche-Environment

Neben dem Problem der Hyperlinks, die nicht mehr verfügbare Dokumente referenzieren, ist eine weitere Problematik – insbesondere bei der Recherche im Internet – der schnelle Orientierungsverlust. Nach einigen wenigen verfolgten Verknüpfungen ist beim Benutzer häufig keine Orientierung mehr vorhanden [CP95, TG97]; er weiß nicht mehr, über welche Stationen er zu dem aktuell in dem Web-Browser geöffneten Dokument gelangt ist. Weiterhin ist für die Recherche im Internet charakteristisch, dass der Benutzer sich schnell über die durch den Hypertext bereitgestellten Hyperlinks von einem Dokument entfernt und sich erst im Nachhinein entsinnt, dass er dieses Dokument zuvor nicht lokal auf seinem Computer abgelegt hat. Selbst wenn der Benutzer die Datei lokal abgespeichert hat, fällt es ihm schwer, sie ex post auf der Festplatte zu lokalisieren und der Web-Präsenz, von der er sie geladen hat, zuzuordnen. Beides sind Problemstellungen, die sich durch die Kombination von dem „History-Browse-Assistant" [HM99a] mit dem „Hyperlink-Focused-Browse-Assistant" [HM99b, HHRM00] lösen lassen. Bei der Recherche im Internet über die Gateway-Konstruktion werden die von den Web-Servern angeforderten Dokumente automatisch indiziert und in einer Datenbank abgelegt. Sie stehen über die auf der Datenbank arbeitende Benutzerschnittstelle – in Form eines lokalen Java-Applets – dem Benutzer jederzeit wiederauffindbar zur Verfügung. Neben der Suche nach Schlagworten in den Dokumenten, wie sie auch im Dateisystem möglich sind, ist bei dem Assistenten-System zusätzlich noch die URI verfügbar, von der das Dokument mit dem

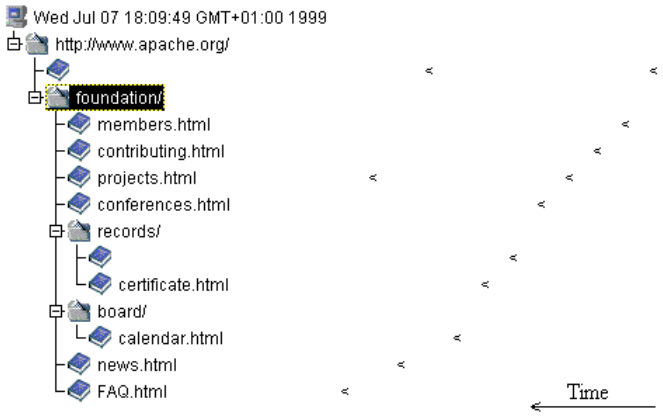

Abbildung 4.8: Die Darstellung der Browser-History als Baumstruktur kombiniert mit einer Zeitreihe.

Web-Browser geladen wurde. Zusammen mit der Eingrenzung des Zeitraums, Teilen der URI und einzelnen Schlagworten ist der Benutzer mit der Recherche-Umgebung jederzeit in der Lage, auch lange Zeit nach dem ersten Besuch Dokumente wiederzufinden. Ergänzt wird diese Wiederauffindbarkeit von Dokumenten durch die Visualisierung der einzelnen, während der damals abgehaltenen Sitzung abgerufenen URIs, in einer gegen eine Zeitreihe aufgetragenen Baumdarstellung (Abb. 4.8).

4.3.4 Zusammenfassung

Gerade im Redaktionssystem-Umfeld mit dem großen Anteil an Autorentätigkeit können die in diesem Abschnitt dargestellten Mehrwertdienste eine reale Arbeitserleichterung bieten. Dazu gehört mit den hier beschriebenen Assistenten-Komponenten auf der einen Seite ein Werkzeug, dass nicht nur dem Online-Autor die Arbeit erleichtert, sondern auch den „normalen“ Benutzer bei seiner Tätigkeit im Internet unterstützt. Auf der anderen Seite lassen sich gerade im wissenschaftlichen Umfeld mit dem Konzept der automatisch beglaubigten Dokumente durch den „Signierenden Gateway“ wichtige Ressourcen konservieren. Beiden hier beschriebenen Mehrwertdiensten gemeinsam ist die auf dem Einsatz offener Standards beruhende einfache Integrierbarkeit in bestehende Systeme und Architekturen: ein simples „Einklinken“ der neuen Komponenten kann jederzeit stattfinden.

Kapitel 5

Das verteilte, Java-basierte Online-Redaktionssystem JDaphne

Nach der Erläuterung der Problemstellung, der Einführung grundlegender Techniken und dem Überblick über die Vorarbeiten in den bisherigen Kapiteln beschreibt dieses die Konzeption und Implementation eines verteilt arbeitenden, in Java implementierten Online-Redaktionssystems. Die von DAPHNE abgeleitete Basis-Konzeption schlägt sich in dem Namen „JDaphne“ nieder. Wie das vorangegangene Kapitel erläutert, hat der Praxiseinsatz von DAPHNE gezeigt, dass die für eine Online-Zeitung entworfene Konzeption zwar für die Verwaltung einer Web-Präsenz übernommen werden kann, in einigen wichtigen Punkten jedoch Modifikationen und Ergänzungen notwendig werden.

Von DAPHNEs Konzeption beibehalten wird bei JDaphne die dateibasierte Verwaltung von Inhalten, die ihren Grund in der angestrebten Dokumenten-Bearbeitung durch Sachbearbeiter an deren Arbeitsplatz hat. Zentraler Gesichtspunkt bei der Konzeption und Entwicklung von JDaphne ist dementsprechend auch der Verzicht auf eine eigenständige Eingabekomponente. Statt eines eigenentwickelten Editors werden beliebige Programme in Abhängigkeit von den Wünschen des Benutzers bzw. der Unternehmensstrategie als Eingabemöglichkeiten eingebunden. Bewährt haben sich ferner die in DAPHNE implementierten Rollen *Autor*, *Kontrolle/Freigabeberechtigter*, *Webmaster* und *Administrator* sowie das web-basierte Arbeiten an sich. Da mit dem Dateisystem als Basis – ergänzt durch Metadaten in einer relationalen Datenbank – gute Erfahrungen gemacht wurden, übernimmt JDaphne dieses Vorgehen genauso wie die Identifikation von Dokumenten über eindeutige Nummern.

Die Modifikationen und Erweiterungen schlagen sich neben der vollständigen Implementation in einer objektorientierten, plattformunabhängigen Programmiersprache (Java), die einen verteilten Aufbau des Systems (vgl. Abschnitt 3.1.4.1) ermöglicht, vor allem in einem verbesserten Dokumenten- und Hyperlink-Management nieder. JDaphne stellt auf den Dokumentenbestand von Web-Präsenzen eingestellte Dokumenten-Repository bereit. Das System bietet flexibel konfigurierbare, rollenbasierte Zugriffskontrolle und stellt Workflows für den Weg eines Dokumentes von der Erstellung über die Publikation bis zur Aufhebung der Publikation nach dem Ablauf seiner Aktualitätsphase bereit. Das auf dem aktuell publizierten und für die Publikation vorbereiteten Dokumentenbestand realisierte Hyperlink-Management wahrt dabei die Konsistenz der Web-Präsenz bezüglich ihrer Verknüpfungsstruktur. Neben der statischen Generierung von Web-Präsenzen wird zusätzlich der dynamische Export von Dokumenten unterstützt, ein Vorgehen, dass gerade bei größeren Dokumentenmengen eine deutlich größere Flexibilität bietet und das Integrieren von externen Datenquellen erleichtert.

5.1 Konzeption von JDaphne

In die Konzeption von JDaphne sind viele Erkenntnisse aus dem Management von Web-Präsenzen mit DAPHNE eingeflossen. Obwohl die Konzeption in Grundzügen analog der von DAPHNE ist, sind in wesentlichen Aspekten umfassende Modifikationen durchgeführt worden. Diese beruhen vor allem aus der in der Projekt-Arbeit erlebten Dynamik von Web-Präsenzen (vgl. Phase 5 und 6 in Abschnitt 3.2.2). Durch den Einsatz im Rahmen eines Online-Zeitungs-Projektes geprägt, unterstützt DAPHNE zwar die Publikation von Dokumenten, ist aber nicht darauf ausgelegt, diese Dokumente einem Überarbeitungszyklus zugänglich zu machen – aus der Sicht einer Online-Zeitung, bei der die einmal veröffentlichten Artikel nicht mehr modifiziert zu werden brauchen, ein korrektes Vorgehen. Beim Einsatz des Systems für Unternehmens-Web-Präsenzen mit einem mehr auf Überarbeitung ausgelegten Dokumentenbestand wurden Hilfskonzepte (z.B. das Hyperlink-Management-Modul, Abschnitt 4.2) notwendig, um die benötigte Funktionalität bereitstellen zu können. Während DAPHNE mit der auf den drei Statusversionen beruhenden Konzeption, die direkt den Workflow vorgibt, auskommen musste und damit nur unzureichend auf die Überarbeitung und Entfernung von Dokumenten vorbereitet war, versucht JDaphne diesen Problemen mit deutlichen Erweiterungen im Dokumentenmanagement Rechnung zu tragen. Ohne Zweifel ist durch die nachfolgend beschriebenen Modifikationen die Komplexität des Gesamtsystems deutlich gestiegen. Da die Implementation jedoch im Gegensatz zu DAPHNE mit einer objektorientierten Programmiersprache umgesetzt wurde, konnte ein großer Teil der Probleme, die durch den „gewachsenen“ Skriptcode von DAPHNE entstanden sind, vermieden werden. Die Entwicklung dieser Anwendung in Java stellt aufgrund der speziell für die Anforderungen des Internet vorbereiteten Programmierschnittstellen und der plattformunabhängigen Einsetzbarkeit dieser Programmiersprache eine wichtige Säule des Konzeptes dar. Dies gilt insbesondere für die Erzeugung von auf der Client-Seite ausführbarem Code, der den Bedienkomfort des Anwenders, wie oben beschrieben, trotz der Einbindung beliebiger Eingabekomponenten erhalten soll. Aber auch auf der Server-Seite erlaubt diese Konzeption die breite Einsetzbarkeit mit großer Plattformunabhängigkeit.

5.1.1 Internes Dokumenten-Management

Anders als dies für die Erstellung und Verwaltung von komplexen Hypertexten nach dem Dexter Hypertext-Modell [HS94] notwendig wäre, arbeitet JDaphne nicht auf Komponenten-, sondern auf Dokumentenebene, genauer auf Dateiebene. Dieses große Zugeständnis an die Interoperabilität mit beliebigen Editor-Anwendungen macht sich in der Flexibilität beim Generieren des Gesamt-Dokumentes, der Web-Präsenz, nachteilig bemerkbar, kann aber durchaus im Rahmen der Erstellung web-gerechter Dokumente kompensiert werden. Für die in-

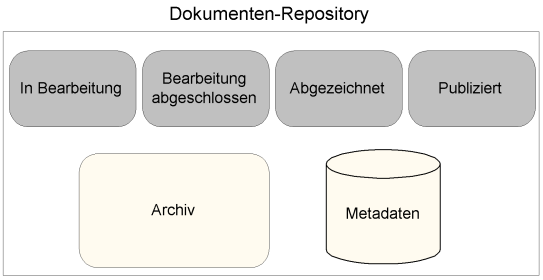

Abbildung 5.1: Das von JDaphne eingesetzte Dokumenten-Repository.

terne Dokumentverwaltung (Abb. 5.1) der Web-Präsenz in Dateiform sieht die Konzeption von JDaphne in der hier beschriebenen Ausbaustufe ein Dokumenten-Repository bestehend aus vier verschiedenen Sub-Repositories, einem Archiv und einer Datenbank für die Metadaten vor. Die vier Sub-Repositories dienen der Ablage von Dokumenten in Abhängigkeit von ihrem Status, das Archiv der finalen Aufbewahrung. Momentan in JDaphne eingesetzt werden Sub-Repositories für die Statusversionen *"in Bearbeitung"*, *"Bearbeitung abgeschlossen"*, *"Abgezeichnet"* und *"Publiziert"*.

Durch diese Aufteilung ist es möglich, Dokumente über alle Versionen hinweg parallel zugreifbar zu halten. Benötigt wird diese Form der Verwaltung, weil die zeitlichen Überschneidungen, die der Mehrbenutzerbetrieb in Kombination mit zeitlichen Verzögerungen im Workflow mit sich bringt, die sequentielle Abfolge der Dokumentenversionen, wie sie auf den ersten Blick möglich erscheint, verhindert. Beispielsweise kann so ein bereits abgezeichnetes Dokument bei diesem parallelisierenden Konzept schon wieder von einem Autor überarbeitet werden, obwohl die abgezeichnete Version noch nicht publiziert wurde. Über die vier Sub-Repositories wird einfach gewährleistet, dass der Workflow konsistent auf die richtigen Dokumentenversionen angewendet wird.

5.1.1.1 Workflow – Dokumentzustände und Aufgaben

JDaphne erweitert die von DAPHNE realisierte, auf Zuständen basierende Workflow-Lösung durch die Einführung von Aufgaben (Tasks). Jeder Zustand (Abb. 5.2) eines Dokumentes stellt einen Schritt im Workflow eines Dokumentes dar. Der Übergang von einem Zustand in einen anderen wird anhand von Tasks in die Wege geleitet. Jede Aktion eines Benutzers, die einen Task abarbeitet, löscht diesen und generiert einen neuen Task. Mit den Tasks einher gehen die Verschiebungen von den Dokumenten aus dem den aktuellen Zustand verwaltenden Sub-Repository in das zu dem neuen Zustand gehörige. Ein Workflow für ein Dokument in JDaphne ist nicht fest vorgegeben. Er ergibt sich aus den Aktionen, die mit einem Dokument in einem bestimmten Zustand ausgeführt werden können. Auf diese Aktionen haben sowohl die Rolle als auch die Rechte der Benutzer einen Einfluss. Wie bei der Beschreibung von Verwaltungsstrukturen in JDaphne in Abschnitt 5.2 erläutert, können Aktionen und Berechtigungen über Rollen und Gruppen konfiguriert werden. Da Dokumente in JDaphne nicht nur sequentielle Workflows durchlaufen können, sondern der parallele Zugriff auf Dokumente in den unterschiedlichen Zuständen möglich ist, steigt die Komplexität des Workflows.

5.1.1.2 Kooperatives Arbeiten mit JDaphne

Unter den Systemen zur computergestützten Gruppenarbeit (CSCW, computer supported cooperative work) ordnet sich JDaphne gemäß Ansatz eins von [RSVW94] ein. JDaphne ermöglicht es Benutzern, einen Gruppenkontext aufzubauen, indem es ihnen gemeinsamen Zugriff auf das Gesamtdokument Web-Präsenz verleiht.

Von den zur Verfügung stehenden Kooperationsmodellen computergestützter Gruppenarbeit [Rod91] setzt JDaphne den Nachrichtenaustausch ein. Schließt ein Benutzer *B1* eine Aktion *A* aus dem Dokumenten-Workflow ab, die die Aktivität eines anderen Benutzers B2 erfordert (z.B. weil Berechtigungen einer anderen Rolle erforderlich sind), so wird automatisch eine Nachricht *N* generiert. Diese wird über SMTP (Simple Mail Transfer Protocol) [Rho97] an die aufgrund ihrer Rolle und Berechtigungen in Frage kommenden Benutzer versandt. Der Inhalt dieser Nachrichten ist informell und enthält neben der Kennung des betroffenen Objekts eine Beschreibung der anfallenden Arbeit. Mit diesen Email-basierten Nachrichtenaustausch realisiert JDaphne ein asynchrones Kooperationsmodell.

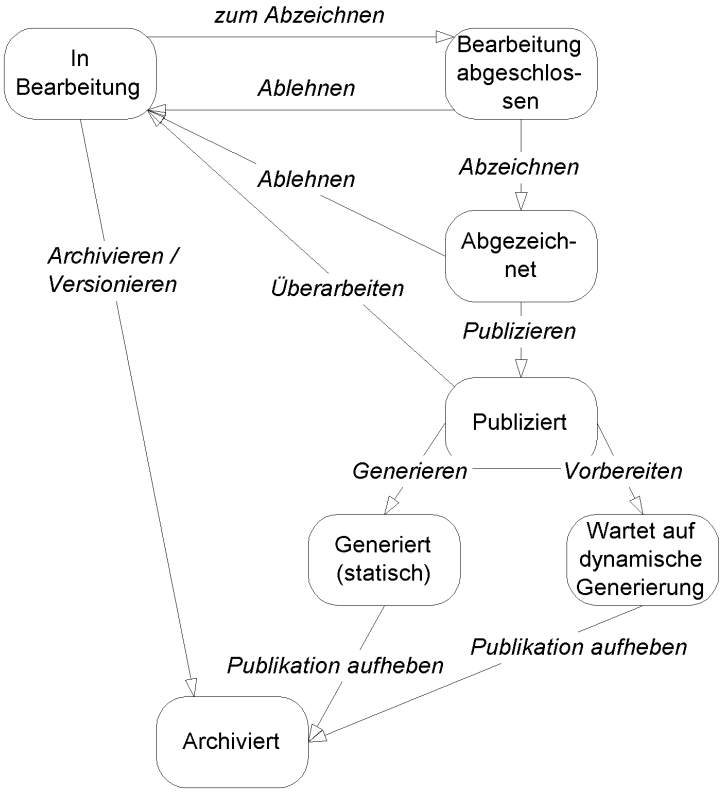

Abbildung 5.2: Ein Dokument in JDaphne kennt aus Sicht des Workflows die hier abgebildeten sieben Grundzustände. Die Übergänge zwischen diesen Zuständen werden, sofern vom System überwacht, über Aufgaben verwaltet, die an die Rollen der Benutzer geknüpft werden.

Ergänzt werden die Kooperationseigenschaften von JDaphne durch gemeinsam genutzte Aufgabenlisten (Information Sharing). Alle sich in einer Gruppe befindlichen Benutzer mit der gleichen Rolle bekommen identische Aufgabenlisten zur Abarbeitung angezeigt. Die internen Sperr-Mechanismen (Document-Locking) verhindern dabei Konflikte bei der Abarbeitung dieser Aufgaben.

5.1.2 Hyperlink-Management

Das bei DAPHNE fehlende Hyperlink-Management stellt bei JDaphne einen großen Teil der neu hinzugekommenen Funktionalität dar. Die Anforderungen an JDaphne's Hyperlink-Management sind vielfältig: Zum einen muss es mit den in JDaphne möglichen Verzeichnisstrukturen kompatibel sein, zum anderen muss es in den Dokumenten-Workflow integriert werden. Konzeptionell unterteilt sich das Hyperlink-Management von JDaphne in drei Bereiche. Dies sind der Bearbeitungsbereich, der Publikationsbereich und die Mehrwertdienste. Zu dem ersten Bereich zählt die initiale Hyperlinkzuordnung. Dokumente, die dem System neu hinzugefügt werden, müssen auf darin enthaltene Hyperlinks untersucht werden. Die aufgefundenen Hyperlinks müssen in der Folge, sofern es sich um interne Hyperlinks handelt, Ressourcen in JDaphne's Datenbestand zugeordnet werden. Im Publikationsbereich liegt die Aufgabe des Hyperlink-Managements in der Wahrung der Hyperlink-Konsistenz. Dokumente, die verschoben oder aus der Web-Präsenz entfernt werden, müssen in den referenzierenden

Dokumenten nachgezogen bzw. eliminiert werden. Der Bereich der Mehrwertdienste umfasst das Vorschlagen von neuen Hyperlinks; bei der Bearbeitung eines Dokumentes wird es dem Benutzer ermöglicht, das System zu konsultieren, um weitere Ziele zu finden, die sein Dokument referenzieren kann.

Das Konzept des Hyperlink-Managements auf dem Dokumentenbestand von JDaphne basiert auf einer grundlegenden Forderung: *Dokumente dürfen nur dann von dem Zustand „in Bearbeitung“ in den Zustand „Bearbeitung abgeschlossen“ überführt werden, wenn sie keine internen Verknüpfungen zu dem System nicht bekannten Dokumenten enthalten.* Dies ist äquivalent zu der Forderung, dass es die Aufgabe des Autors ist, die Konsistenz der Dokumente bezüglich ihrer Hyperlinks herzustellen. Falls JDaphne die von einem Dokument referenzierten Ressourcen nicht automatisch zuordnen konnte, muss der Autor sie eindeutig spezifizieren. Für die Bewahrung externer Ressourcen ist das Hyperlink-Management nicht ausgelegt. Diese können lediglich im Rahmen des Publikations-Prozesses überprüft werden.

5.1.2.1 Statusbedingte Deaktivierung von Hyperlinks

Basierend auf oben genannter Forderung nach Konsistenz kann davon ausgegangen werden, dass sämtliche Verknüpfungen eines Dokumentes im Zustand *“Bearbeitung abgeschlossen“*, *“Abgezeichnet“* und *“Publiziert“* auf Dokumente zeigen, die dem System bekannt sind und über deren Verfügbarkeit in einem Zustand das System eindeutige Aussagen machen kann. Hauptaufgabe des Hyperlink-Managements stellt nun die dauerhafte Wahrung dieser Konsistenz dar. Das Konzept von JDaphne sieht für diese Aufgabe den Mechanismus der *Hyperlink-(De-)Aktivierung* vor. Verweist ein Dokument, dessen Bearbeitung abgeschlossen ist, auf ein Dokument, welches noch in der Bearbeitung befindlich ist, so wird der Hyperlink, der auf dieses Dokument verweist, temporär bis zur Fertigstellung des Dokumentes deaktiviert. Während für den internen Gebrauch deaktivierte Hyperlinks dem Anwender visualisiert werden, wird ein deaktivierter Hyperlink in der publizierten Version nicht als Hyperlink sichtbar.

Die interne Visualisierung erfolgt über eine Grafik, die neben dem Hyperlink eingeblendet wird und so den Deaktivierungszustand anzeigt. Sobald das Dokument, auf das ein deaktivierter Hyperlink verweist, eine Statusänderung durchläuft, die den Hyperlink funktionstüchtig macht, wird dieser wieder aktiviert. Unter Berücksichtigung der verschiedenen Dokumentenversionen in den Sub-Repositories wird auf diese Weise die Hyperlink-Konsistenz zustandsübergreifend gewahrt. Der (De-) Aktivierungsmechanismus bewährt sich im übrigen auch beim Entfernen eines Dokumentes aus einem Repository, weil die Deaktivierung, vor dem Entfernen durchgeführt, verhindert, dass der Hyperlink inkonsistent wird. Nach dem Entfernen dieses Dokumentes, wird der Hyperlink aus den übrigen Dokumenten entfernt.

Anders verhält sich der Hyperlink-Aktivierungs-Mechanismus bei Teildokumenten, die mittels des Source-Tags eingebunden sind. Diese Teildokumente, z.B. Grafiken oder Bilder, werden als essentielle Bestandteile eines fertiggestellten Dokumentes gewertet. Als Teil des Dokumentes begriffen, in das sie eingebunden sind, werden sie automatisch mit diesem Dokument bei einer Zustandsänderung mitberücksichtigt. Der Benutzer kann entweder das Dokument mit seinen Teildokumenten zusammen in einen anderen Status überführen oder muss darauf verzichten.

Neben den Hyperlinks, die in den Dokumenten eingebettet sind und deren Behandlung bereits kurz skizziert wurde, werden für eine Web-Präsenz über die Navigationsstruktur Hyperlinks von dem System generiert. Die Hyperlinks der Navigationsstruktur sind bei JDaphne eng mit den Deckdokumenten der Ressorts korreliert. Über das Publizieren eines Deckdokumentes für ein Ressort entsteht in allen anderen Dokumenten, entsprechend der Template-Vorgaben, ein Hyperlink auf dieses Dokument. Die so entstandenen Hyperlinks werden im weiteren analog zu den internen Hyperlinks behandelt.

5.1.3 Einbindung beliebiger Editor-Anwendungen

Ein wichtiges Element der Konzeption von JDaphne ist, wie schon bei DAPHNE, der Verzicht auf eine eigene Eingabekomponente. Um ohne eigenen Editor auszukommen und stattdessen die Möglichkeit zu schaffen, beliebige Anwendungen einbinden zu können, sind auf der Client-Seite aktive Komponenten notwendig, die die Übergabe der zu bearbeitenden Dokumente an die Anwendung und umgekehrt den Import des Ergebnisses des Bearbeitungs-Prozesses bewerkstelligen. Bei JDaphne ist die normale Benutzerschnittstelle als kleine Java-Applikation (Java-Applet) konzipiert. Sie ist neben der Visualisierung des Dokumentenbestandes und der Aufgaben-Listen für die Anbindung der externen Anwendungen ausgelegt. Selektiert der Benutzer in der Visualisierungskomponente ein Dokument für die Überarbeitung, so holt das Applet, nach Überprüfung der Berechtigungen und der Verfügbarkeit, das Dokument auf den lokalen Arbeitsplatz und legt es dort im Dateisystem ab. Im nächsten Schritt startet das Applet die vom Benutzer zum Bearbeiten von Dokumenten des entsprechenden Formats ausgewählte Applikation parametrisiert mit der lokal auf dem Arbeitsplatz abgelegten Datei. Dadurch öffnet sich die Editoranwendung mit dem zur Bearbeitung lokal abgelegten Dokument. Während dieses Vorgangs zeigt eine eigenständige Übersicht von JDaphne an, welche Dokumente der Benutzer gerade auf seinem lokalen Arbeitsplatz geöffnet hat.

Nach der Ausführung der gewünschten Modifikationen an dem Dokument kann der Benutzer dieses auf Knopfdruck wieder dem System zuführen, um auf Mehrwert-Funktionen von JDaphne, z.B. die Linkvorschläge oder einfach nur eine dem finalen Ergebnis nachempfundene Voransicht, zuzugreifen. Da die Konzeption vorsieht, dass das System Internet-Technik-basiert arbeitet, muss berücksichtigt werden, dass ein Benutzer sich prinzipiell von vielen verschiedenen Arbeitsplätzen aus mit dem System verbinden kann. Deshalb stellt das benutzerbezogene, arbeitsplatzabhängige Konfigurationsmanagement einen wichtigen Punkt bei der Integration von beliebigen Editor-Anwendungen dar. Im System wird verwaltet, welche Anwendung der Benutzer auf welchem Arbeitsplatz als Editor-Anwendung zum Bearbeiten eines bestimmten Dokumentenformats eingestellt hat.

5.2 Verwaltungsstrukturen in JDaphne

Die konsistente Verwaltung von Dokumenten, ihrem Verarbeitungs- und Zugriffsstatus und ihren Hyperlinks sowie den Verzeichnissen, in denen sie abgelegt werden, stellt bei JDaphne den Großteil der implementatorischen Anforderungen dar. Die kooperative Arbeit an dem Dokumentenbestand erfordert die Implementation von Zugriffskontrollen [SD92]. Basis einer Rechteverwaltung, wie sie JDaphne implementiert, ist eine leistungsfähige Benutzerverwaltung. JDaphne realisiert eine interne Benutzerverwaltung auf Datenbank-Basis.

Da sich diese auf die JDaphne internen Attribute beschränkt, findet im Normalfall eine Integration der unternehmensweiten Benutzerverwaltung statt. Über die Anpassung interner Schnittstellen lassen sich statt der internen Datenbank externe Verwaltungen etwa in Verzeichnisdiensten mittels standardisierter Protokolle wie LDAP (Light Weight Directory Access Protocol) [Gro99] nutzen.

5.2.1 Berechtigungen

Für die Verwaltung von Berechtigungen setzt JDaphne eine rollenbasierte Zugriffskontrolle [SCFY96, HHZM00] aufgesetzt auf Verzeichnisstrukturen [ZHH+99] ein. JDaphne unterscheidet dazu intern zwischen der Berechtigung eines Benutzers, eine Aktion überhaupt aufrufen zu dürfen und der Berechtigung, mit einem bestimmten Objekt, z.B. einem Dokument, diese Aktion ausführen zu können. Dies erlaubt eine fein abstimmbare, aber sehr praktikable

Rechtevergabe in JDaphne. Die Zuteilung der Berechtigungen über Rolle und Objekt hat den Vorteil, dass die Benutzerschnittstelle rollenabhängig aufgebaut werden kann. Da beim Anwendungsstart bekannt ist, welche Aktionen prinzipiell von dem Benutzer in seiner Rolle ausgeführt werden können, ist die Anpassung der Benutzerschnittstelle auf genau diese Aktionen möglich. Ein Wechsel der Rolle während der Laufzeit ist dabei nicht vorgesehen. Einem Benutzer werden kumulativ alle Aktionen zur Verfügung gestellt, die den erlaubten Rollen entsprechen. Die Dokumentenübersichten werden so aufgebaut, dass sie nur die für die erlaubten Aktionen relevanten Dokumente anzeigen. Ob ein Benutzer Zugriff auf ein Dokument hat, hängt wie oben beschrieben, von der Berechtigung des Benutzers für das das Dokument beheimatende Verzeichnis ab. Welche Rechte mit einer Rolle verbunden sind, ist

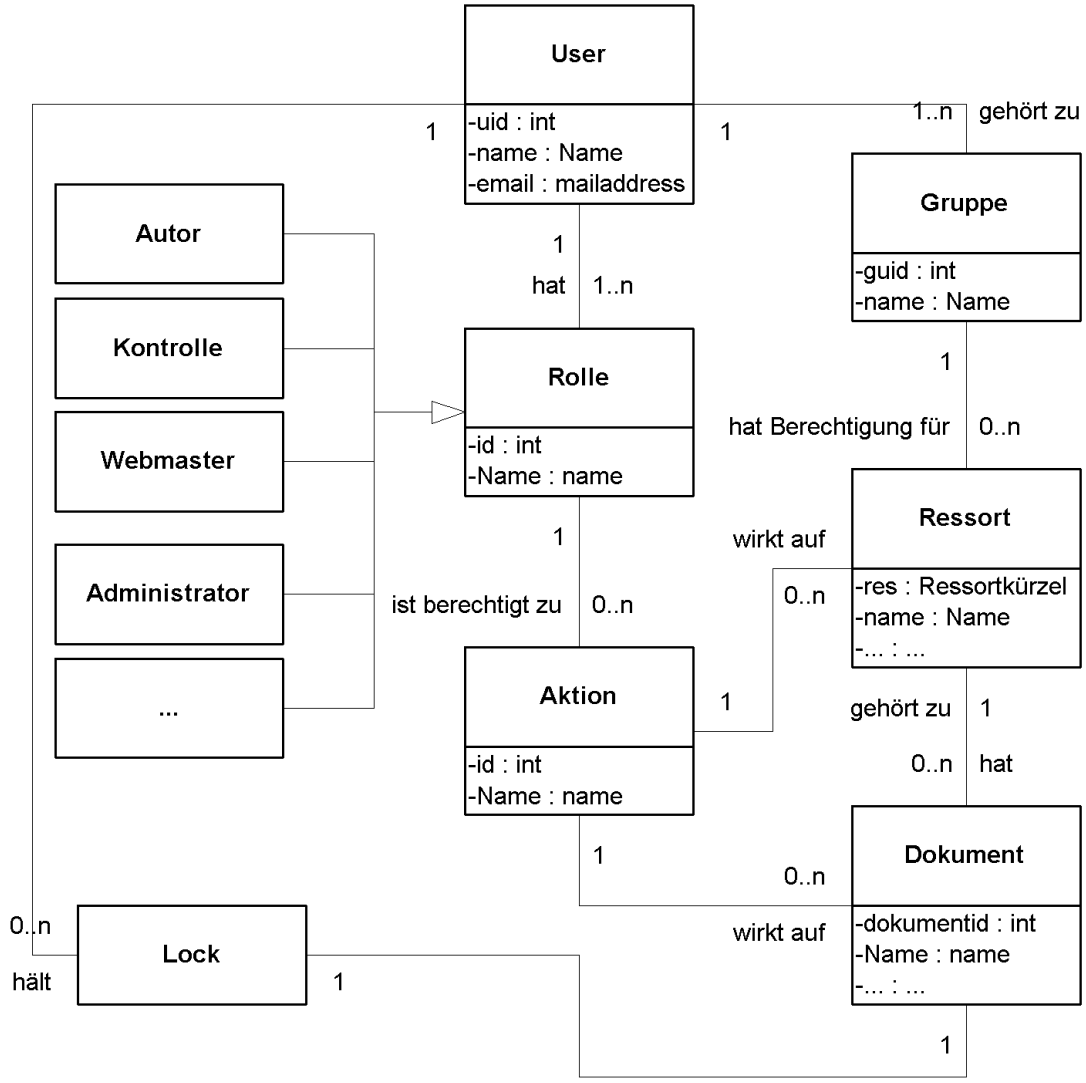

Abbildung 5.3: Berechtigungen in JDaphne dargestellt in Klassendiagramm: User, Rolle, Gruppe, Aktion, Ressort und Dokument

bei JDaphne frei konfigurierbar. Somit ist jederzeit einstellbar, welche Aktionen für welche Rolle erlaubt sind. Dies ist insbesondere deshalb interessant, weil sich dadurch der Workflow von Dokumenten bezüglich der beteiligten Benutzer modifizieren lässt. Einen Überblick über

die an der Verwaltung von Berechtigungen beteiligten Klassen gibt Abbildung 5.3.

5.2.1.1 Aktionen

JDaphne arbeitet in der Berechtigungsverwaltung mit Aktionen. Aktionen im Verständnis von JDaphne beziehen sich im Allgemeinen auf Dokumente in einem bestimmten Zustand. Dokumente im Zustand *„In Bearbeitung“* dürfen beispielsweise editiert (*“Bearbeiten-Aktion“*) oder ersetzt (*“Ersetzen-Aktion“*) werden. Dokumente, die *„Zum Abzeichnen“* vorliegen können entweder *„abgezeichnet“* (*“Abzeichnen-Aktion“*) oder *„abgelehnt“* (*“Ablehnen-Aktion“*) werden. Welche Aktionen auf Dokumente in welchem Dokumentenzustand angewendet werden dürfen, kann über die Rollenberechtigungen modifiziert werden. Eine umfassende Übersicht über die in JDaphne verwirklichten Aktionen folgt bei den technischen Beschreibungen (siehe Seite 114).

5.2.1.2 Rollen

Rollen sind in JDaphne die Träger von Berechtigungen für Aktionen. Bei der Konzeption der in JDaphne konfigurierten Rollen wurde vorausgesetzt, wie schon bei DAPHNE gezeigt, dass eine vergleichsweise kleine Anzahl von Rollen für den produktiven Einsatz hinreichend ist. Über die Konfiguration der Berechtigungen lassen diese sich auf die speziellen Bedürfnisse der Unternehmen anpassen. Geeignet für die Verwendung in einem produktiven Umfeld haben sich die folgenden Rollen gezeigt.

5.2.1.2.1 Autor

Den Inhabern dieser Rolle kommt die grundlegende Aufgabe der Dokumenterstellung und -modifikation zu. Nach dem Konzept des Publizierens vom Arbeitsplatz aus handelt es sich bei den Autoren in erster Linie um fachlich qualifiziertes Personal (Sachbearbeiter), welches aus dem direkten Arbeitsumfeld stammende Informationen im Sinne des Unternehmens für die Web-Präsenz aufbereitet. Innerhalb von JDaphne sollen die von den Autoren erstellten Dokumente wohl strukturiert, aber ohne weitergehendes Layout erstellt werden. Da mit dem Online-Redaktionssystem ein weitgehend einheitliches Aussehen aller Dokumente erzielt werden soll, ist es hilfreich, wenn Dokumente nicht bereits vom Autor mit einem Layout versehen werden. Sicherheitshalber setzt JDaphne frei konfigurierbare Filter-Algorithmen ein, die bei Bedarf unerwünschtes Layout entfernen und durch das vom System vorgegebene ersetzen. Nach dem Einspielen eines Dokumentes, entweder neu oder nach einer Überarbeitung, kann der Autor es in der dynamischen Voransicht unmittelbar überprüfen. Gegebenenfalls wird ein erneuter Bearbeitungsschritt notwendig, wenn das Dokument noch nicht der Zielvorstellung entspricht. Dem Autor obliegt weiterhin, durch das System unterstützt, auch die Verknüpfung seines Dokumentes mit den bereits auf der Web-Präsenz befindlichen Dokumenten. Vereinfacht wird seine Aufgabe durch eine Suchmaske, die anhand der aus den Dokumenten extrahierten Schlagworte arbeitet. Alternativ kann der Autor auch direkt Hyperlink-Vorschläge abfragen.

5.2.1.2.2 Kontrolle/Freischaltungsberechtigter

Zur Gewährleistung eines Mehr-Augen-Prinzips wird eine weitere Rolle benötigt, die der Freischaltungsberechtigten (Kontrolle). Ihre Aufgabe liegt in der Qualitätssicherung. Neben der Realisierung eines Sechs-Augen-Prinzips durch Nacheinanderschalten von Kontrollinstanzen kann auch eine inhaltliche Zuordnung von Freischaltungsberechtigten erfolgen. Diese kann

z.B. für rein inhaltliche Aspekte, sprachliche Aspekte oder rechtliche Belange (z.B. Copyrights, vgl. [End96]) erfolgen. Neben der qualitativen Überprüfung stellt die Rolle des Freigabeberechtigten sicher, dass keine unternehmensinternen Informationen nach außen gelangen. Abgesehen von diesen Aufgaben obliegt es den Freischaltungsberechtigten auch, die inhaltliche Strukturierung ihres Bereiches auf der Web-Präsenz vorzunehmen. Sie haben innerhalb ihrer Gruppe, aus der die inhaltliche Zuordnung erfolgt, die Berechtigung für Aktionen, die Ressorts anlegen oder ändern.

5.2.1.2.3 Webmaster

Die für den Gesamtauftritt der Web-Präsenz zuständige Rolle nimmt der Webmaster ein. Von dem Webmaster, bzw. den Webmastern wird der übergreifende Überblick über die gesamte inhaltliche Struktur der Web-Präsenz verlangt. Je nach Ausprägung dieser Rolle können die Autorisierungen von formalen Berechtigungen, wie z.B. dem endgültigen Export auf die Web-Präsenz im Internet bis zur Gloabalberechtigung für alle Aktionen reichen. Dem Webmaster wäre es dann beispielsweise möglich, in ein Dokument eine weitere Verknüpfung einzufügen oder eine bestehende, aber unpassende zu entfernen. Sinnvollerweise obliegt dem Webmaster auch die Pflege der rahmengebenden inhaltlichen Struktur auf den höheren Ebenen. Da der Webmaster nach außen hin die Verantwortung für die Web-Präsenz übernimmt und als Ansprechpartner bei Problemen mit dieser gilt, sollte diese Rolle von einer entsprechend qualifizierten verantwortungsbewussten Person ausgefüllt werden.

5.2.1.2.4 Administrator

Üblicherweise fallen an einem System administrative Arbeiten an. Neben einer sich ändernden Benutzermenge, welche entsprechende Wartung der Benutzerberechtigungen (Zugang zum System) nach sich zieht, sind auch die Vergabe von Rollen- und Gruppenzugehörigkeiten permanent anfallende Arbeiten.

Über Schnittstellen im System lassen sich administrative Aufgaben und die Konsistenz des Dokumentenbestandes betreffende Aktionen in die Wege leiten. So kann beispielsweise eine Neuerhebung von Schlagworten genauso angestoßen werden wie eine Überprüfung des Verknüpfungszustandes der Web-Präsenz und eine Neuinitialisierung der internen Dokumentenzustände.

Weiterhin umfasst die Administratorenrolle die Wartung der JDaphne zugrunde liegenden Komponenten (Web-Server, Datenbank-Server etc.). Diese Aufgaben können nicht über Schnittstellen innerhalb von JDaphne abgewickelt werden. Deshalb wird von den Personen, die die Administratorenrolle, ausfüllen eine gute technische Kenntnis der einzelnen Komponenten verlangt.

5.2.1.2.5 Designer

Die Aufgaben der Rolle des Designers liegen im Rahmen von JDaphne in der Erstellung und Zuordnung der für das Layout der Dokumente zuständigen Templates und Style Sheets. Weiterhin werden die ggf. für die Navigationsstruktur benötigten Icons und Grafiken vom Designer erstellt und in das System eingespielt. Abgesehen von den Sonderberechtigungen, auf System-Verzeichnisse zugreifen zu können, ist die Rolle des Designers in diesem Punkt mit der der Autoren vergleichbar. Da die Erstellung von Templates jedoch gute technische Kenntnis der Layout-Mechanismen von JDaphne verlangt, ist diese Rolle insgesamt auf der Ebene des Administrators anzusiedeln.

Wie bereits mehrfach erwähnt ist das Design einer Web-Präsenz (Layout etc.) für ihren Erfolg sehr wichtig. Dementsprechend ist die Rolle des Designers nur von Fachkräften zu

besetzen, die über entsprechende Kompetenz auf der einen Seite und technischen Hintergrund auf der anderen Seite verfügen.

5.2.1.2.6 Leser

Neben den aktiv an dem Aufbau und der Verwaltung der Web-Präsenz beteiligten Rollen gibt es prinzipiell auch inaktive Rollen. Eine davon ist beispielsweise der Leser. Die Inhaber dieser Rolle dürfen gemäß der Gruppe, in der sie sich befinden, auf Dokumente lesend zugreifen. Im Unterschied zu den Besuchern der Web-Präsenz haben Leser auch Zugriff auf noch nicht veröffentlichte Dokumente in der Voransicht.

5.2.1.3 Rollen-Aktions-Matrix

Die Zuordnung von erlaubten Aktionen zu den einzelnen Rollen geschieht mittels einer Rollen-Aktions-Matrix. In ihr sind Rollen gegen Aktionen aufgetragen. Für jede Rolle kann frei definiert werden, welche Aktion erlaubt ist; alle nicht erlaubten Aktionen sind verboten. Dieser Ansatz gewährt größere Sicherheit bei der Implementation von neuen Aktionen. Eine mögliche Rollen-Aktions-Matrix für JDaphne ist in Abb. 5.4 dargestellt.

Rolle / Aktion	Autor	Freischaltungs-berechtigter	Webmaster	Administrator	Designer	Leser
Lesen						★ Dokument
Erstellen	★ Dokument	★ Ressort		★ Benutzer	★ Template	
Bearbeiten	★ Dokument	★ Ressort		★ Benutzer	★ Template	
Ersetzen	★ Dokument	★ Ressort			★ Template	
Löschen	★ Dokument	★ Ressort		★ Benutzer	★ Template	
Umbenennen	★ Dokument	★ Ressort			★ Template	
Abzeichnen		★ Dokument	★ Ressort ★ Template			
Publizieren			★			
Pub. aufheben		★ Dokument	★			
...						

Abbildung 5.4: Eine vereinfachte Darstellung der Rollen-Aktions-Matrix von JDaphne. In der Darstellung sind die Aktionen in den Spalten für die Rollen nach dem Objekt unterschieden, auf das sie wirken.

5.2.1.4 Gruppen

Während die Berechtigung zur Ausführung von Aktionen über die Rolle eines Nutzers bestimmt wird, hängt die Berechtigung für ein Objekt von der Gruppe des Benutzers ab (Abb. 5.3). Dementsprechend werden einzelne Benutzer in Gruppen zusammengefasst. Gruppen in JDaphne sind solange die Zugehörigkeiten intern verwaltet werden, nicht hierarchisch organisiert (vgl. [Sik97]). Dies geschieht im Allgemeinen aufgrund von inhaltlichen Zusammenhängen, die sich z.B. aus einem gemeinsamen Arbeitsgebiet ergeben können. Jeder Grup-

pe werden innerhalb des Systems Verzeichnisse (Ressorts) zugeordnet. Auf diese Ressorts mitsamt den darin enthaltenen Dokumenten dürfen die Mitglieder der Gruppe gemäß ihrer Rolle zugreifen.

5.2.2 Verzeichnisse (Ressorts)

Innerhalb von JDaphne stellen Verzeichnisse die ordnenden zusammenfassenden Gliederungsinstrumente für die Dokumente dar. Aus historischen Gründen als Ressorts bezeichnet, bauen diese Container für Dokumente die interne Struktur der Web-Präsenz auf (Abb. 5.5). An die Ressorts sind über die Gruppen die Berechtigungen der Benutzer gebunden (vgl. Abb. 5.3), die oben beschriebenen Aktionen auszuführen. Die interne Verwaltung der Ressorts geschieht anhand einer eindeutigen Kennung, dem Ressortkürzel. Dieses, auch ein aus dem Konzept von DAPHNE entliehenes Vorgehen, besteht aus einer Kombination von vier Buchstaben.

Verknüpft mit jedem Ressortkürzel sind die Metadaten des Ressorts, dessen Namen in den verschiedenen unterstützten Sprachen, das jeweilige Deckdokument und die Verwendbarkeit des Ressorts für den Aufbau der Navigationsstruktur der Web-Präsenz.

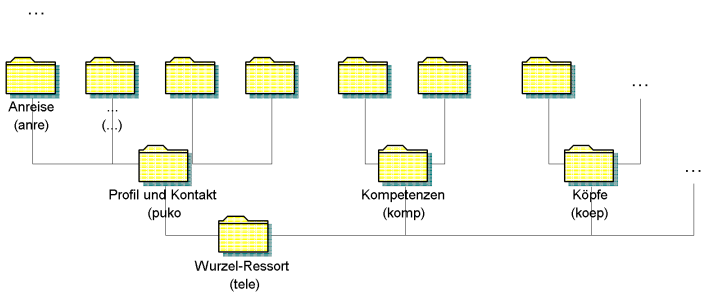

Abbildung 5.5: Eine Ressort-Hierarchie in JDaphne. Ausgehend von einem Wurzel-Ressort werden Ressorts basierend auf den eindeutigen Ressortkürzeln zu einer hierarchischen Struktur zusammengesetzt. Unterhalb des Ressortnamens in der jeweiligen Sprache, hier in Deutsch, zeigt die Abbildung in Klammern jeweils das zugehörige Ressortkürzel.

5.2.2.1 Versionierung

Im Gegensatz zu Dokumenten, die in vier Sub-Repositories gesichert sind, werden bei den Ressorts nur zwei Repositories verwaltet. Dies sind das interne Repository und das externe, d.h. die publizierte Version. Der Übergang eines Ressorts von dem internen Repository in das externe bzw. das Entfernen eines Ressorts aus dem externen Repository ist an die Publikation bzw. das Aufheben der Publikation von Dokumenten gekoppelt.

Die beiden Repositories werden benötigt, damit intern der Aufbau einer neuen Struktur möglich ist. Gibt es ein Ressort bislang nur im internen Repository, wird das Ressort auch in das externe Repository aufgenommen, sobald das erste Dokument aus dem Ressort publiziert wird. Damit einher geht das Anlegen des zugehörigen Verzeichnisses im Dateisystem. Bei diesem Schritt werden gleichzeitig die Metadaten des Dokumentes in das zugehörige externe Repository übernommen. Umgekehrt wird ein Ressort aus dem externen Repository gelöscht, sobald das letzte Dokument aus diesem Repository nicht mehr extern sichtbar ist.

5.2.3 Dokumente

Dokumente auf Dateibasis stellen die Informationseinheiten dar, mit denen JDaphne arbeitet. Unterschieden nach ihrem Format werden die Dokumente entsprechenden Verarbeitungsschritten unterworfen. Die Verwaltung der Dokumente in JDaphne erfolgt anhand von eindeutigen Nummern. Über diese kann auf alle Dokumente im System eindeutig zugegriffen werden. Wie schon bei DAPHNE wurden die Nummern aus Zuordnungsgründen als Basisidentifikation der Dokumente gewählt. Alternativ kann die Kombination aus Ressort und Dokumentenname als eindeutiger Schlüssel verwendet werden. Da jedoch beide Namen im System an sich variabel gehalten sind, d.h. jederzeit modifizierbar sein sollen, stellt die Dokumentennummer die sicherere Identifikationsmerkmal dar. Weil die Nummern lediglich der internen Zuordnung dienen und sowohl dem Benutzer von JDaphne als auch dem Besucher der Web-Präsenz die gerade aktuellen Namen der Dokumente visualisiert werden, bringt ihre Verwendung keine Nachteile mit sich. Einzig bei der Vergabe von Namen für Dokumente besteht die Einschränkung, dass Dokumente nicht mit einer Ziffernkombination benannt werden dürfen. Vorteilhaft bei der Verwendung von Nummern ist, dass sie prinzipiell kürzer als die Dokumentennamen sind und deshalb einfacher z.B. für Hyperlinkverknüpfungen verwendet werden können. Im übrigen hat sich im Betrieb des Systems gezeigt, dass nur wenig Benutzer dazu tendieren, ihre Dateien mit sprechenden Namen zu versehen.

5.2.3.1 Repository

JDaphne legt die in das System eingespielten Dokumente in einem Dokumenten-Repository (s.o.) ab. Um auf die Dokumente entsprechend ihrem Zustand zugreifen zu können, werden mit JDaphne zustandsabhängige Dokumenten-Sub-Repositories eingesetzt. Jedes Dokument kann über die Angabe seiner Dokumentennummer und dem gewünschten Zustand über das Repository aus dem entsprechenden Sub-Repository abgefragt werden. Damit ist parallelisierter Zugriff bezüglich des Dokumenten-Zustandes für die Inhaber verschiedener Rollen im System möglich.

Die Dokumentenversionen in den einzelnen Repositories unterscheiden sich zum einen durch den Zustand ihrer Hyperlinks, zum anderen bei größerem Verzug zwischen Bearbeitung und Publizierung auch in ihrem Inhalt. Die abgezeichnete Version, die der Webmaster gerade ins WWW publiziert, kann einen anderen, älteren Inhalt haben als die gerade von dem Autor neu fertig gestellte, überarbeitete Version, die jetzt auf die Autorisierung durch den Freischaltungsberechtigten wartet.

Wie in der Konzeption erwähnt, steigen dadurch die Anforderungen an das Hyperlink-Management. Hyperlinks können nur für die referenzierten Dokumente aktiviert werden, die in dem entsprechenden Zustand ebenfalls verfügbar sind. Aber auch für die Metadaten der Dokumente ergibt sich eine nicht unerhebliche Aufwandssteigerung, da für jedes Dokument in jedem Zustand die passenden Metadaten vorgehalten werden müssen.

5.2.4 Metadaten

Um Dokumente im World Wide Web zu erschließen und über Suchmaschinen gezielt mit Hilfe bestimmter Kategorien auffindbar zu machen, werden sog. Metadatenelemente verwendet. Für den offenen Datenaustausch im Internet von großer Bedeutung ist die Interoperabilität der Metadatenelemente. Aktuell eingesetzt werden Metadatenformate wie beispielsweise *Dublin Core* [WKLW98] (Dublin Core, Warwick Framework – benannt nach den Workshops in Dublin, Ohio 1995 und Warwick, U.K. 1996) oder *Ressource Description Framework* (RDF)

[LS99]. Einen umfassenden Überblick über verschiedene Formate bietet beispielsweise Rusch-Feja [RF97].

Mit JDaphne werden Metadaten zu zwei verschiedenen Zwecken verwaltet. Zum einen, um innerhalb des Systems die Dokumente verwalten zu können, zum anderen, um sie für die Publikation im Internet einsetzen zu können. Während die Daten für den internen Gebrauch in einer Datenbank abgelegt werden, integriert sie der Publikationsprozess in das HTML-Dokument im *Dublin Core* und *RDF*-Format.

Wie bereits erwähnt, erlaubt JDaphne parallelen Zugriff auf Dokumente in unterschiedlichem Zustand. Deshalb wird für jedes Dokument in jedem Sub-Repository ein vollständiger Satz von Metadaten bereitgehalten. Dieser wird bei dem Zustandsübergang eines Dokumentes in den neuen Zustand adaptiert.

Zu den Metadaten, die in JDaphne verwaltet werden, gehören Basisinformationen wie Autor, Titel, Untertitel, Sprache, Erstellungs-, Änderungs- und Erscheinungsdatum, Schlagworte und eine Kurzbeschreibung des Inhalts. Um jedes Dokument bei der Publikation mit einer vollständigen, die gesamte Web-Präsenz wiedergebenden Beschreibung versehen zu können, ermöglicht JDaphne es zusätzlich zu den Dokumenten, auch die Ressorts mit Metadaten auszustatten. Diese werden ergänzend zu den für das Dokument verfügbaren Metadaten bei der Publikation im WWW mit in die Dokumentenbeschreibung eingebracht. Je nach Bedarf können auch die Metadaten der in der Hierarchie übergeordneten Ressorts für diese generalisierten Metadaten eines Dokumentes herangezogen werden. Mit JDaphne wird somit eine umfangreiche, immer auch die gesamte Web-Präsenz reflektierende Beschreibung der Dokumente sichergestellt. Dies zahlt sich gerade dann aus, wenn auf die Web-Präsenz über Suchmaschinen zugegriffen wird. Der Suchende erhält neben der speziellen Beschreibung des Dokumenteninhalts immer auch gleich einen kurzen Überblick über die Inhalte der gesamten Web-Präsenz. Dies hilft während der Suche, die Relevanz eines „Fundes“ einzuschätzen.

Die Vergabe von Metadaten geschieht in JDaphne zu einem großen Teil automatisiert. Alle Daten, die aus dem Dokument bzw. dessen Handhabung verfügbar sind, werden direkt übernommen. Zusätzliche Informationen, wie z.B. dedizierte Schlagworte oder eine kurze Beschreibung des Inhaltes lassen sich nicht ohne weiteres automatisieren. Hier sind die Autoren gefragt. Sie werden bei der Fertigstellung eines Dokumentes vom System aufgefordert, das Dokument mit weiteren Metadaten näher zu charakterisieren.

Neben dem externen Nutzen der Metadaten in den publizierten Dokumenten sind diese auch für die internen Mechanismen von Bedeutung. Sie werden zum einen für die Auffindung von Dokumenten, z.B. im Rahmen einer Stichwort-Suche, eingesetzt, die im übrigen auch auf der externen Web-Präsenz verwendet wird. Zum anderen stellen sie die Basis für die Algorithmen dar, die Hyperlinkvorschläge für die Verknüpfung der Dokumente liefern. Dieses geschieht entweder als Ausgangslage für das Case Based Reasoning (CBR) [HRHM00] oder direkt über die Suche nach passenden Dokumenteninhalten anhand von Schlagworten aus Titel oder Dokument. Die Qualität der Metadaten ist ausschlaggebend für die Resultate der Linkvorschlagkomponente.

5.3 JDaphne's Architektur

Zur Umsetzung der oben beschriebenen Konzeption setzt JDaphne eine verteilte System-Architektur ein. Das Gesamt-System (Abb. 5.6) besteht aus untereinander kommunizierenden Komponenten auf der Server-Seite und web-basierten Benutzerschnittstellen auf der Client-Seite. Bei der Architektur ist zwischen internen, d.h. im Intranet betriebenen Komponenten, und externen, d.h. auf dem Web-Server der Web-Präsenz arbeitenden Komponenten, zu unterscheiden. Während erstere dem Aufbau und der Verwaltung der Web-Präsenz dienen, stellen

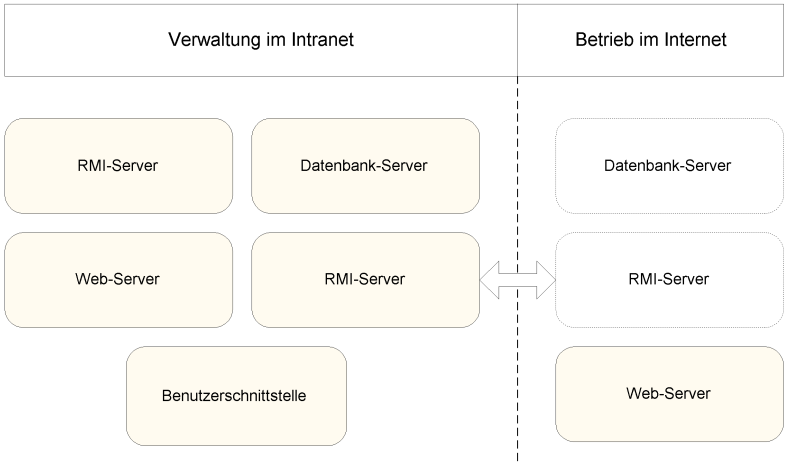

Abbildung 5.6: Komponenten im Intra- und Internet (schematisch).

letztere die Basis für den Betrieb der Web-Präsenz dar. Welche Komponenten genau dabei auf der Internet-Seite des Systems zum Einsatz kommen, hängt von der gewählten Export-Variante, „statischer“ oder „dynamischer“ Export, ab. Analog zu den System-Komponenten verteilt sich auch der Datenbestand (Abb. 5.7) auf den intern von JDaphne genutzten und den extern für den Betrieb der Web-Präsenz benötigten. Im folgenden werden zuerst die

Abbildung 5.7: Der Datenbestand von JDaphne aufgesplittet nach internem Datenbestand und für den Betrieb auf dem externen Web-Server benötigten Daten bei statischem und dynamischem Export.

zu der internen Architektur gehörenden Komponenten inklusive der Benutzerschnittstellen von JDaphne beschrieben. Im Anschluss daran folgt die Betrachtung der auf der System-Architektur auf dem externen Rechner (im WWW) zum Einsatz kommenden Komponenten. Alle im Rahmen von JDaphne entwickelten Komponenten sind in Java in der Version 1.2 bzw. 1.3 entwickelt worden. Neben der weitreichenden Plattformunabhängigkeit trägt insbesondere auch die hochentwickelte Netzwerktauglichkeit von Java zu der verteilten System-Architektur bei. Ohne weiteres können alle im folgenden beschriebenen Server-Komponenten

auf separaten Rechnern installiert werden. Bei geeigneter Netzwerkbandbreite lässt sich so die Performance des Systems in einem weiten Bereich an die jeweiligen Bedürfnisse anpassen.

5.3.1 System-Architektur im internen Netz

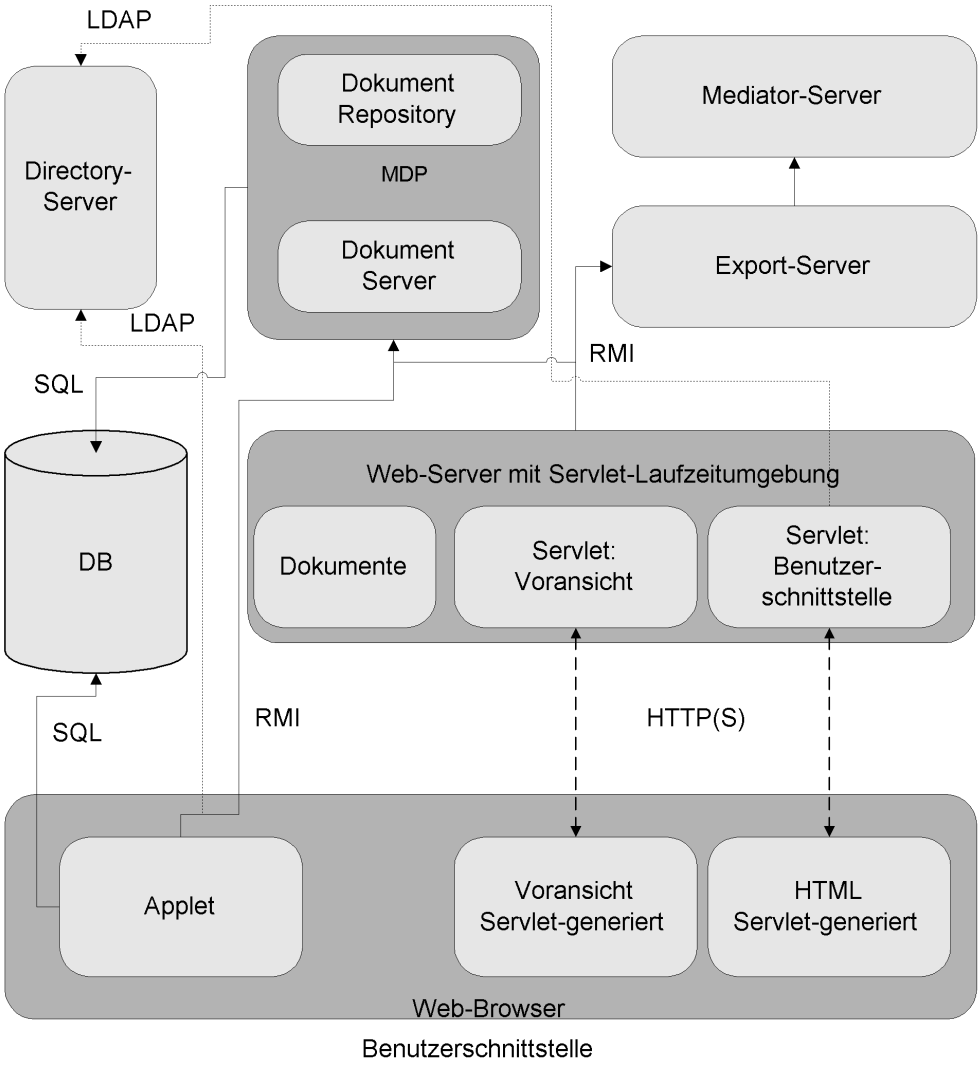

Abbildung 5.8: Die Architektur von JDaphne im internen Netz.

Die interne Architektur (Abb.5.8) von JDaphne kann in die Benutzerschnittstelle und die beteiligten Server-Komponenten unterteilt werden. Die Benutzerschnittstelle des Systems wird durch den Web-Server bereitgestellt. Er bietet zum einen die Laufzeitumgebung für die Servlets und liefert zum anderen die zum System gehörenden Objekte an den Web-Browser des Benutzers. Aus der Perspektive von JDaphne ist der Dokumenten-Server die zentrale Komponente; er verwaltet den Datenbestand des Systems verteilt auf das Dateisystem und eine relationale Datenbank. Die Aufbereitung der Dokumente, d.h. die Zusammenführung von Inhalt, Layout und Navigation, übernimmt der Export-Server. Der optional einsetzbare Directory-Server kann alternativ zu der Datenbank für die Benutzerverwaltung zum Einsatz kommen. Die Kommunikation zwischen den einzelnen Komponenten arbeit mit den Protokollen HTTP [BLFF96, GMM+99] und RMI [Far98]. Die Datenbank wird mittels SQL

angesprochen. Für die Einbindung des Directory-Servers wird LDAP eingesetzt. Zu allen genannten Protokollen existieren auf dem Secure Socket Layer (SSL) oder ähnlichen Tunnel-Mechanismen basierende sichere Alternativen. Werden diese statt der Standard-Protokolle eingesetzt, kann auf JDaphne auch außerhalb des Intranets von im Internet lokalisierten Rechnern ohne Sicherheitsbedenken zugegriffen werden.

5.3.1.1 Web-Server

Der Web-Server dient als Server-Komponente der Bereitstellung der Benutzerschnittstelle von JDaphne für den Zugang mittels Web-Browser. Dabei erfüllt der Web-Server zwei Funktionen: zum einen erlaubt er den Web-Browsern den Download der interaktiven Benutzerschnittstelle in Form des unten beschriebenen Java-Applets sowie zu dem System gehöriger Dokumente wie Übersichten, Anleitungen und Grafiken.

Zum anderen ist Aufgabe des Web-Servers der Betrieb der Laufzeitumgebung der für die Benutzerschnittstelle benötigten Java-Servlets. Hierzu bindet der Web-Server eine den aktuellen Servlet-Spezifikationen 2.1 [HC00] entsprechende „Servlet-Engine“, den aus dem Jakarta-Projekt stammenden Tomcat [McL00], ein.

Als Web-Server kann ein beliebiger eingesetzt werden, der die Einbindung einer den Spezifikationen genügenden Servlet-Engine erlaubt. Ferner wird für die Integration in eine PKI bzw. den sicheren authentisierten Zugriff auf den Server eine SSL-Erweiterung zum Aufbau von HTTPS-Verbindungen benötigt.

In der prototypischen Implementation von JDaphne kommt als Web-Server ein Apache 1.3.19 [LL99] zum Einsatz. Für diesen existieren zum einen unter den gängigen Betriebssystemen entsprechende Module zur Einbindung von Servlets, zum anderen lassen sich damit bei Einsatz des SSL-Moduls auch HTTPS-Verbindungen und somit auch in offenen Netzen eine sichere Kommunikation mit Client- und Server-Authentisierung realisieren.

5.3.1.2 Datenbank

Eine weitere wichtige, ebenso wie der Web-Server von JDaphne integrierte Fremd-Komponente ist die für die Verwaltung der Metadaten eingesetzte Datenbank. Als relationale Datenbank mittels SQL eingebunden stellt sie die Datenbasis für die Dokumenten- und Hyperlink-Management-Fähigkeiten von JDaphne dar. Sofern die Benutzerverwaltung nicht an einen Directory-Server ausgelagert wird, werden hier auch Benutzerinformationen sowie Gruppen- und Rollenzugehörigkeiten gespeichert.

Bei der in dem Prototypen eingesetzten relationalen Datenbank handelt es sich um eine Shareware Datenbank, die MySQL-Datenbank in der Version 1.22 [KK01]. Mit der flexiblen Datenbankschnittstelle JDBC [Ree97], welche bei JDaphne zum Einsatz kommt, kann davon ausgegangen werden, dass ein schneller Wechsel des Datenbanksystems möglich ist. Da alle Zugriffe anderer Komponenten auf die Datenbank zudem über eine gemeinsame Datenbankklasse erfolgen, sind Modifikationen der zugrundeliegenden SQL-Abfragen an zentraler Stelle möglich. Die aktuelle Implementation der Datenbankklasse setzt weitestgehend Standard-SQL [DD97] (ANSI Level 2) ein. Nur in sehr zeitkritischen Prozeduren oder mangels Standardbefehlen wurden native Abfragemöglichkeiten der MySQL-Datenbank ausgenutzt. Auf diese beschränkt sich dementsprechend bei einer Migration auf eine andere Datenbank der zu leistende Umsetzungsaufwand.

Da die MySQL-Datenbank für die meisten Betriebssysteme verfügbar ist, ergeben sich durch sie kaum Einschränkungen bei der Wahl einer Systemplattform. Als Bestandteil des Verteilungs-Konzeptes von JDaphne ist es jederzeit möglich, die Datenbank gegebenenfalls

auf einem separaten Rechner zu installieren, da alle anderen Komponenten „remote“ auf die Datenbank zugreifen.

5.3.1.3 Dokumenten-Server

Um über eine optimale Möglichkeit zum Lastausgleich zu verfügen, sind die Server-seitigen Komponenten von JDaphne als eigenständige Server-Programme implementiert. Ihre Funktionalitäten werden von den Servern untereinander genauso wie von den Benutzer-Schnittstellen mittels RMI abgerufen.

Von der zentralen Komponente im Konzept von JDaphne, dem Dokumenten-Server, wird die Konsistenz von Datenbank und Dokumenten-Repository gewährleistet. Er stellt sämtliche Funktionalitäten bereit, die auf die Dokumente und Ressorts angewendet werden können. Zu seinen Aufgaben gehört der Import von Dokumenten, das Parsen von Metadaten, das Umsetzen von Hyperlinks und die Bereitstellung des Dokumenten-Workflow.

Alle Funktionalitäten sind speziell für den Aufruf aus den Benutzerschnittstellen optimiert. Durch ausführliche Rückgabewerte wird die Benutzerschnittstelle in die Lage versetzt, dem Benutzer detailliert Auskunft über die durchgeführte Aktion zu geben. Um die Aufrufe netzwerktauglich zu halten, zeichnen sich die Methodenaufrufe durch geringste Datenvolumina aus. Sofern nicht ein Dokument selbst übertragen werden muss, z.B. beim Import, werden Methoden lediglich mit den Basisinformationen, im Allgemeinen den Dokumentennummern der zu bearbeitenden Dokumente, aufgerufen. Auf die in JDaphne implementierten Funktionalitäten, die zum größten Teil von dieser Komponente bereitgestellt werden, geht Abschnitt 5.4 in diesem Kapitel ein.

5.3.1.4 Export-Server

Der Export-Server ist die für die Zusammenführung von Inhalten mit Layout und Navigation in HTML-Dokumenten zuständige RMI-Komponente. Sie verfügt über die Methoden zum Exportieren, d.h. insbesondere zum Generieren von Dokumenten. Bei dem Generierungsprozess werden die unformatierten Dokumente in die Templates mit den Layout-Vorschriften eingebaut. Dabei werden weiterhin alle in den Templates gesetzten Navigationselemente mit Werten belegt. In Abhängigkeit von dem angestrebten Verwendungszweck, für interne Voransicht, externen statischen oder dynamischen Export, werden die hinter den Navigationselementen liegenden Hyperlinks gesetzt.

In enger Zusammenarbeit mit dem Export-Server stehen das für die interne Voransicht zuständige Servlet und der gegebenenfalls auf dem externen Web-Server betriebene, ebenfalls über ein Servlet realisierte dynamische Export. Beide rufen über RMI die Generierungsfunktionalität des Exportservers für die von den Web-Browsern angeforderten Dokumente ab. Die Voransicht verwendet dafür die Dokumente aus den internen Sub-Repositories von JDaphne, der dynamische Export generiert seine Dokumente auf der Basis der publizierten Dokumente.

Innerhalb der System-Architektur ist der Export-Server im internen Netz positioniert. Seine Export-Funktionalitäten können bei geeigneter Konfiguration der das interne von dem externen Netz trennenden Firewall ohne weiteres für dynamische Exporte von dem externen Web-Server aus aufgerufen werden.

5.3.1.5 Mediator-Server

Die letzte der drei RMI-Server-Komponenten von JDaphne ist der Mediator-Server. Seine zentrale Aufgabe liegt in der Einbindung und Integration externer Datenquellen [Wie92, ABS00] in die Web-Präsenz. Bei JDaphne wird er aber auch für die Generierung von Übersichten

über den publizierten Dokumentenbestand zu bestimmten Themen oder Ressorts eingesetzt. Parametrisiert über in JDaphne als „aktive Dokumente“ abgelegte HTML-Dokumente mit Datenquellen und Aktionsbeschreibung führt diese Komponente die Abfrage der spezifizierten Datenquellen aus. Abgefragt werden können neben SQL-Datenbanken (JDBC), z.B. der internen Datenbank, auch beliebige URIs. Das Ergebnis der Abfrage wird an die Stelle der Parametersätze in die HTML-Seite integriert. Während für die Abbildung komplexer Prozesse nach wie vor andere Konzepte, wie z.B. Java-Server-Pages zum Einsatz kommen sollten, bietet der Mediator-Server die Möglichkeit, auf einfache Art und Weise externe und interne Daten in die Web-Präsenz zu integrieren. Die Mediatorkomponente kommt sowohl für den statischen Export, als auch beim dynamischen Export für die interne Voransicht und die externe Web-Präsenz zum Einsatz.

5.3.1.6 Directory-Server

Für den Betrieb von JDaphne ist kein Directory-Server (Verzeichnis-Dienst) notwendig. Existiert in einem Unternehmen bereits eine in einem Directory-Server abgelegte Benutzerverwaltung lässt sich diese auszugsweise für JDaphne nutzen. Benutzer und Rollen können dann aus dem Directory-Server abgefragt werden, um die Authentisierung durchzuführen und die erlaubten Rollen zuzuordnen. Ein Directory-Server wird insbesondere dann benötigt, wenn JDaphne in eine bestehende PKI eingebunden werden soll. In diesem Fall ist zum einen die Umsetzung der internen Kennungen auf die in den Zertifikaten genannten IDs durchzuführen, zum anderen muss der Zugang zu dem Directory-Server konfiguriert werden.

5.3.1.7 Benutzer-Schnittstelle

Die Benutzerschnittstellen von JDaphne sind entsprechend der Konzeption webbasiert. JDaphne bietet für den Web-Browser zwei verschiedene Schnittstellen mit unterschiedlichen Funktionalitäten an. Als minimale Schnittstelle, gerade für den Gebrauch auch außerhalb des Intranets über vergleichsweise langsame Modem-Leitungen wurde die von einem Servlet generierte HTML-Schnittstelle konzipiert. Im Gegensatz dazu, mit hoher Funktionalität, aber auch hohem Ressourcenbedarf ausgestattet, bietet die als Java-Applet implementierte Java-Schnittstelle die von den Benutzern an ihrem Arbeitsplatz gewohnte Handhabung mit großer Interaktivität bei hohem Benutzerkomfort. Beide Varianten zeichnen sich durch eine weitreichende Unabhängigkeit von der auf dem Arbeitsplatz des Benutzers eingesetzten Web-Browser-Version aus. Die HTML-Schnittstelle vermeidet konsequent die Ausnutzung Web-Browser-spezifischer Eigenschaften und verzichtet für diese Unabhängigkeit gegebenenfalls auf Benutzerkomfort. Bei Bedarf kann durch beträchtlichen Entwicklungsaufwand eine Optimierung auf einen Standard-Web-Browser stattfinden. Da jedoch mit der Java-Variante eine komfortablere Schnittstelle bereitsteht, kann auf diesen Aufwand verzichtet werden.

Der Web-Browser wird von JDaphne in beiden Varianten als Anzeigekomponente für die Voransicht von Dokumenten eingesetzt. Deshalb sollte, wenn möglich, für JDaphne derjenige Web-Browser eingesetzt werden, für den die mit JDaphne erstellte Web-Präsenz konzeptioniert wurde. Dann lassen sich die eingespielten Dokumente in dem korrekten Umfeld bezüglich ihres Designs und ihrer Funktionalität überprüfen.

5.3.1.7.1 Java-Applet

Die als Java-Applet implementierte Benutzerschnittstelle von JDaphne bietet aufgrund des an Windows angelehnten „Look & Feels“ den Benutzern einen hohen Bedienkomfort. Beim Start als eigenständiges Fenster geöffnet, stellt das Applet neben den anhand des

Dokumenten-Repositories und der Datenbank aufgebauten Dokumentenübersichten die Bedienelemente zum Ausführen von allen Methoden des Dokumenten-Servers über RMI bereit. In Abhängigkeit von der Rolle des sich anmeldenden Benutzers wird die Funktionalität der Benutzerschnittstelle konfiguriert. So wird jedem Benutzer entsprechend seiner Rolle eine passende Funktionalität geboten. Weiterhin ergänzt dieser Mechanismus die Rechte-Überprüfungen vor dem Ausführen von durch den Benutzer initiierten Aktionen. Der hohe Bedienkomfort und die aufwendige Visualisierung, die das Java-Applet bietet, haben ihren Preis. Da es sich bei Java um eine Interpreter-Sprache handelt, ist auf dem ausführenden Client-Rechner der Performancebedarf hoch und laut *SUN* [Eck00] mindestens ein Pentium-Prozessor 166 MHz und 48 MB Hauptspeicher erforderlich.

Ein Vorteil des Java-Plug-ins liegt in der einheitlichen Sicherheitsarchitektur für das Ausführen von vertrauenswürdigem Code. Dies ist bei JDaphne insbesondere für Aktionen notwendig, die die Sandbox öffnen müssen, z.B. für den Zugriff auf das Datei-System beim Import von Dateien. Da die Java-Benutzerschnittstelle von JDaphne die Berechtigung benötigt, Dateien im Dateisystem zu lesen und zu schreiben, muss der Applet-Code signiert werden. Das Java-Plug-in von *SUN* bietet zwei verschiedene Mechanismen für die Ausführung von vertrauenswürdigem, d.h. signiertem Java-Code an. Wie das Plug-in den signierten Code behandelt, hängt von dem Algorithmus ab, mit dem das Java-Archiv signiert wird. Der Signatur-Algorithmus wiederum ist abhängig von dem Algorithmus, mit dem das Schlüsselpaar erzeugt wurde, dessen privater Schlüssel zum Signieren des Archivs eingesetzt wurde. Zur Verfügung für das digitale Signieren stehen der DSA- und der RSA-Algorithmus. Bei mit dem DSA-Algorithmus signierten Archiven greift das Java-Plug-in auf die fein granular einstellbare Sicherheits-Policy zurück, die im Dateisystem des Web-Browsers für den Signatur-Ersteller hinterlegt werden muss. Um auf die Verteilung solcher Policies verzichten zu können, setzt JDaphne den RSA-Algorithmus zum Code-Signieren ein. Dies hat zur Folge, dass vor der Ausführung des Codes auf der Client-Seite eine globale Sicherheitsabfrage (Abb. 5.9) stattfindet. Der Benutzer wird gefragt, ob er global die Berechtigung zum Zugriff auf die Systemressourcen geben möchte. Eine solche wird benötigt, um z.B. Verbindungen zu

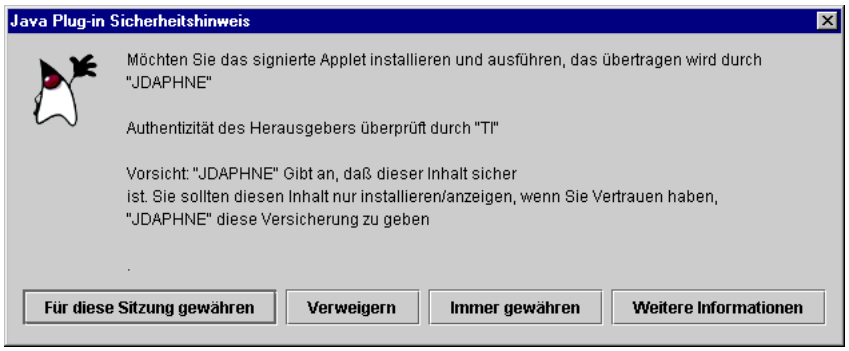

Abbildung 5.9: Der Sicherheitshinweis des Java-Plug-ins von SUN vor der Ausführung von signiertem Code.

den RMI-Servern (siehe Seite 117) aufzubauen und wie bereits erwähnt, auf Dateien in dem lokalen Datei-System zuzugreifen.

5.3.1.7.2 HTML-Servlet

Alternativ zu dem Java Plug-in von *SUN* mit seinem Windows ähnlichem „Look & Feel" bietet JDaphne eine HTML-Variante als Benutzerschnittstelle an. Ihr Ressourcenbedarf auf

der Client-Seite ist im Vergleich zum Applet gering. Von einem Servlet, das die Anbindung an die Datenbank und die Dokumenten-Server-Komponente verwaltet, generierter HTML-Code visualisiert – im Web-Browser angezeigt – die Bedienelemente von JDaphne. Das Servlet übernimmt dabei die Ausführung der Methoden des Dokumenten-Servers über RMI. Der Umfang der Funktionalität ist an den des Java-Applets angelehnt, erreicht diesen aber nicht vollständig. Auf der Server-Seite steigt durch das Servlet der Ressourcenbedarf spürbar. Durch die geeignete Verteilung der Benutzer auf die beiden Schnittstellen bietet die Architektur zusätzliche Möglichkeiten der Lastverteilung. Während die Applet-Variante der Benutzer-Schnittstelle für den Betrieb im Intranet konzipiert ist und deshalb auf die Verwendung von sicheren Datenübertragungsprotokollen verzichtet, ist die HTML-Variante primär für den Einsatz über das Internet gedacht. In diesem Fall lässt sich die Nutzung der SSL-Fähigkeiten des Web-Servers für sichere Übertragungen über HTTPS mit der sicheren Authentisierung durch die Einbindung in eine PKI kombinieren.

5.3.1.7.3 Editor-Anwendungen

Konzeptionelle Entscheidung für die Bearbeitung von Dokumenten beliebiger Formate ist die Integration externer Editor-Anwendungen. Dementsprechend stellen die Eingabekomponenten zwar keinen direkten Bestandteil der System-Architektur von JDaphne dar, ihre Existenz ist jedoch essentiell für die Arbeit mit JDaphne. Voraussetzung für die Integrierbarkeit einer externen Anwendung zum Bearbeiten von Dokumenten stellt deren Ausführbarkeit parametrisiert mit dem Dateinamen des zu bearbeitenden Dokumentes als eigenständiger Prozess des Betriebssystems dar. JDaphne setzt zur Bearbeitung eines Dokumentes den Betriebssystem-Aufruf der für das entsprechende Format konfigurierten Anwendung ab. Parametrisiert mit dem Dateinamen startet die Anwendung mit dem zur Bearbeitung geöffneten Dokument. Nach der Bearbeitung wird das Dokument- durch den Benutzer angestoßen, wieder zu dem Server transferiert.

5.3.2 System-Architektur im externen Netz

Die für die Bereitstellung der Web-Präsenz im Internet benötigte Architektur unterscheidet sich aufgrund anderer benötigter Funktionalitäten von der intern zum Einsatz kommenden. Im Wesentlichen wird ein Web-Server benötigt, um die Dokumente an die Web-Browser im

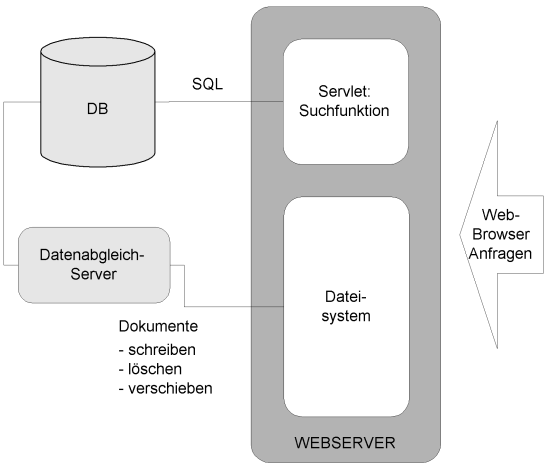

Abbildung 5.10: Visualisierung der Architektur bei statischem Export.

WWW zu distribuieren. Der in einem Dateisystem abgelegte Datenbestand wird, sofern dieser Prozess nicht manuell oder durch einen Datei-Transfer mit FTP durchgeführt wird, von einem als RMI-Komponente realisierten Datenabgleich-Server mit dem internen Datenbestand abgeglichen. Für die Bereitstellung von Suchfunktionalitäten und bei dynamischem Export von Dokumenten wird zusätzlich zu dem Web-Server noch eine Laufzeitumgebung für Servlets sowie eine Datenbank benötigt. Je nach Konfiguration kommt extern ein eigenständiger Export-Server zum Einsatz oder die interne Komponente wird auch von extern aufgerufen. Die Abbildungen 5.10 und 5.11 zeigen, dass für die Realisierung des dynamischen Exports eine

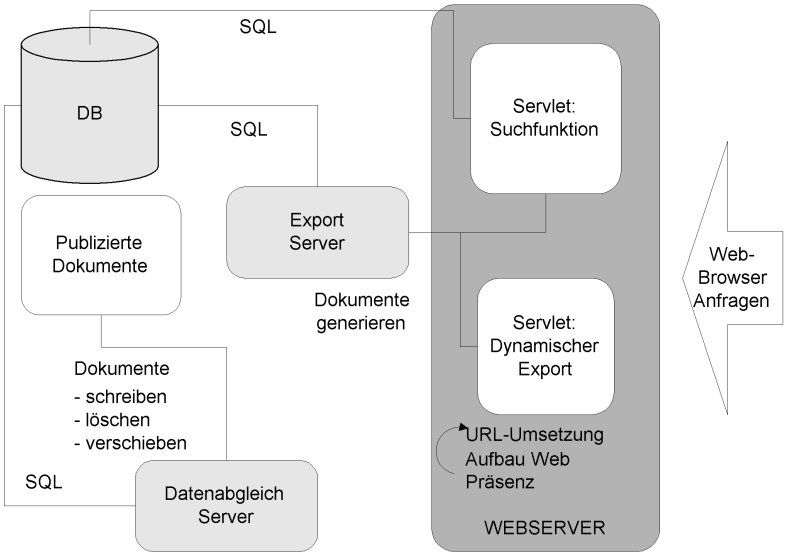

Abbildung 5.11: Visualisierung der Architektur bei dynamischem Export.

deutlich umfangreichere Architektur benötigt wird als für den statischen Export. Aufgrund der größeren Möglichkeiten der Anpassung dynamischer Dokumente, z.B. bei der Personalisierung von Dokumenten, ist ihr Nutzen jedoch unbestritten.

5.3.2.1 Web-Server

Analog zu der Web-Server-Komponente bei der Erläuterung der internen Struktur kann auch der extern zum Einsatz kommende Web-Server prinzipiell frei gewählt werden. Eingeschränkt wird die Wahl des Servers allerdings dann, wenn dynamische Komponenten auf der Web-Präsenz zum Einsatz kommen sollen. Sowohl das für die Suche durch Benutzer der Web-Präsenz konzipierte Such-Servlet als auch die dynamische Export-Möglichkeit von JDaphne benötigen eine Laufzeitumgebung für Servlets.

5.3.2.2 Datenabgleich-Server

Der Datenabgleich-Server dient dem Abgleich von internem und externem Datenbestand. Von intern über RMI aufgerufen macht diese Komponente den aktuell publizierten Datenbestand für die Web-Präsenz zugänglich. Vorrangig werden die Dokumente in dem Dateisystem des Web-Servers abgelegt und nicht mehr notwendige Dokumente entfernt. Zusätzlich findet, sofern extern vorhanden, eine Aktualisierung der Datenbank statt. Die für die aktuell publizierten Dokumente verfügbaren Metadaten werden mit dem internen Bestand abgeglichen.

5.3.2.3 Export-Server

Der bei entsprechender Konfiguration des Systems extern zum Einsatz kommende Export-Server ist identisch mit dem intern eingesetzten.

5.3.2.4 Datenbank

Bei der Datenbank auf dem externen Web-Server handelt es sich um einen Auszug der intern eingesetzten Datenbank. Ein zu dem internen Datenbank-Server analoger Server speichert die für Suchfunktionen und dynamischen Export notwendigen Metadaten zu den publizierten Dokumenten. Bei dem Prototypen kommt die gleiche Datenbank wie intern (eine MySQL-Datenbank) zum Einsatz. Da auch diese Datenbank von den anderen Komponenten über JDBC mit SQL angesprochen wird, steht dem Austausch gegen ein anderes Fabrikat wenig entgegen.

5.4 In JDaphne implementierte Funktionalitäten

Die Konzeption eines Redaktionssystems ohne eigene Eingabekomponente stellt besondere Herausforderungen an die Implementation eines an sich schon komplexen Systems. Im Rahmen dieses Abschnittes werden die für die Betrachtung der Konzeptrealisierung mit JDaphne relevanten Mechanismen und Algorithmen erläutert. Dabei handelt es sich weniger um eine detaillierte Auflistung und Beschreibung der Vielzahl bei der Implementation eingesetzter Algorithmen, als vielmehr um eine globalere Sicht auf die implementierten Techniken (vgl. Anhang A, B) mit dem Fokus auf die zentralen technischen Aspekte von JDaphne. Zu diesen gehören der Dokumentenimport an sich, d.h. die Filter- und Parse-Mechanismen, die Hyperlink-Management-Funktionalität und schließlich die Export-Mechanismen von JDaphne.

5.4.1 Dokumentenimport/-bearbeitung

Ein wichtiger Aspekt bei JDaphne ist die Integration externer Eingabekomponenten. Dies wirkt sich insbesondere auf die Handhabung von Dokumenten in dem System aus. Zum einen werden deshalb Dateien zu den zentralen Informationseinheiten, mit denen das System arbeitet, zum anderen gliedert sich dadurch die Bearbeitung eines Dokumentes in mehrere Stufen, visualisiert über den Zustand *"in Bearbeitung"* in Abb. 5.12. Die erste Stufe beinhaltet den Datentransfer von dem Server auf den Client und vis versa. Die zweite Stufe bereitet die Dokumente durch Filterung auf. Die dritte Stufe schließlich macht die importierten Dokumente durch Parse-Vorgänge, bei denen die notwendigen Informationen aus den Dateien extrahiert werden, für das System zugänglich. Die so importierten Dokumente werden dem von JDaphne vorgegebenen Workflow unterworfen und können abschließend auf der Web-Präsenz veröffentlicht werden. Die hier kurz skizzierten Mechanismen werden im folgenden detailliert besprochen.

5.4.1.1 Einbindung – Editor-Anwendung

Die Integration beliebiger Eingabekomponenten durch JDaphne funktioniert analog zu dem in DAPHNE implementierten Mechanismus der Hilfsapplikationen (siehe S. 80). Der Vorteil dieses Mechanismus' gegenüber der „Bearbeiten-Funktion", den z.B. gängige Web-Browser bieten, liegt in der Browser-Unabhängigkeit. Nach der Einstellung der entsprechenden Mime-Types startet der Web-Browser parametrisiert die Hilfsapplikationen, die die Bearbeitung der

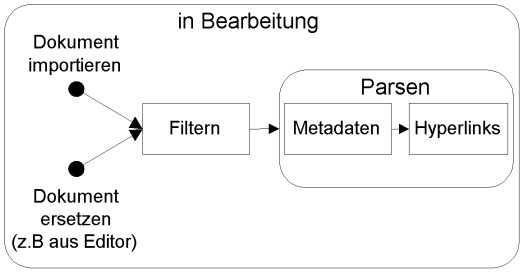

Abbildung 5.12: Durch den Neu-Import initial oder nach einem Bearbeitungsvorgang kommt ein Dokument in den Zustand „in Bearbeitung“. Dort wird es im Falle des HTML-Formats über Parse-Mechanismen dem System zugänglich gemacht. Die für die Verwaltung des Dokumentes notwendigen Metadaten werden ermittelt und in der Datenbank abgelegt.

Dokumente überwachen. Im Gegensatz zu dem Upload der Dokumente über den Web-Browser direkt, muss sich der Benutzer nicht mehr darum kümmern, wo die lokale Kopie des Dokumentes abgelegt wird und diese beim Wiedereinspielen suchen. Dieser für den Benutzer unnötige Aufwand wird ihm durch die Benutzerschnittstelle von JDaphne abgenommen. Konkret sieht der Bearbeitungsvorgang (Abb. 5.13) eines Dokumentes bei JDaphne wie folgt aus: Selektiert

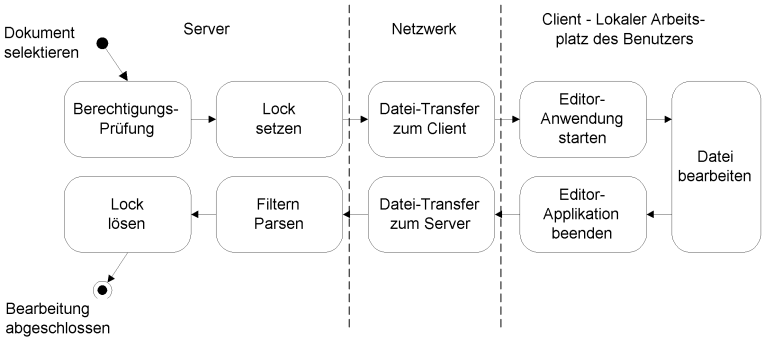

Abbildung 5.13: Der Bearbeitungsvorgang eines Dokumentes durch einen Autor in Einzelschritten.

der Benutzer in der Benutzerschnittstelle von JDaphne ein Dokument für die Bearbeitung, so erfolgt in einem ersten Schritt eine Berechtigungsüberprüfung. Sofern das Dokument nicht von einem anderen Benutzer bearbeitet wird, setzt das System eine Bearbeitungssperre (Lock) für dieses Dokument. Das nun für andere Benutzer nur noch lesend zu öffnende Dokument wird auf den lokalen Arbeitsplatz transferiert. Handelt es sich bei der Benutzerschnittstelle um das interaktive Java-Applet, so führt dieses den Transfer durch. Andernfalls wird der Transfer über den Web-Browser gesteuert und das Dokument mitsamt einem Satz Parameter, die das Dokument spezifizieren, an eine der oben beschriebenen Hilfsapplikationen übergeben. Dieser bzw. dem Applet obliegt es, die eigentliche Editor-Anwendung zu starten. Dazu muss für jeden Benutzer konfiguriert werden, mit welcher Applikation welches Dokumentenformat bearbeitet werden soll. Diese Daten, die auch von dem Benutzer selbst modifiziert werden können, werden, inklusive ihrer Arbeitsplatzabhängigkeit, mit den Benutzerdaten auf dem Server gespeichert.

Entsprechend der Konfiguration wird nun die eingestellte Applikation, parametrisiert mit

der lokal abgelegten Kopie des Dokumentes, gestartet. Für diesen Schritt gibt es zwei Optionen: entweder die Datei wird nach dem Schließen der Anwendung durch die Hilfsanwendung bzw. das Applet nach Rückfrage automatisch auf den Server zurückgespielt oder es wird eine eigenständige Übersicht in Form eines „In Bearbeitung“-Fensters von den Dokumenten erzeugt, die die Hilfsanwendung momentan lokal zum Bearbeiten abgelegt hat. Diese können dann bei Bedarf jederzeit nach dem Speichern durch die Editor-Anwendung zum Server übertragen und zurück nach JDaphne gespielt werden. Ist die Bearbeitung des Dokumentes bzw. der Dokumente beendet, so werden diese auf Knopfdruck durch den Benutzer zurück nach JDaphne gespielt. In diesem Schritt wird die Bearbeitungssperre für das Dokument gelöst. Das an den Server übertragene Dokument wird bei dem Einspielprozess entsprechend seinem Format weiterverarbeitet. Da HTML das interne Basisformat für Dokumente darstellt, werden insbesondere Dokumente dieses Formats einer umfangreichen Verarbeitung unterworfen.

5.4.1.2 Dokumentfilter

Besonders wichtig im Hinblick auf die Integration beliebiger Editoren, insbesondere von Anwendungen der Office-Pakete, ist die Filterung der importierten HTML-Dokumente. Sie dient speziell der Bereinigung der HTML-Dokumente von unerwünschten HTML-Tags. In Anbetracht der angestrebten Trennung von Inhalt und Layout sind Dokumente, die über eine nicht policy-konforme Formatierung verfügen, ohne eine solche Filterung nur unzureichend einsetzbar.

JDaphne setzt zu diesem Zweck zwei eigenentwickelte, in Java implementierte Streaming-Filter ein. Zum einen gibt es einen Tag-Filter, zum anderen einen Standard-Filter. Als Stream-Filter implementiert lassen sich beide Filter in einen Datenstrom einklinken und können so z.B. direkt beim Einlesen der Datei in das System angewendet werden, ohne dass ein zusätzlicher Lese- und Schreibvorgang der Daten anfällt. In Hinsicht auf die Performance-Probleme, die Java als interpretierte Programmiersprache mit sich bringt, stellt dies eine optimierende Strategie dar.

Der Standard-Filter dient der Bereinigung aller eingelesenen Dokumente von unerwünschten Ausdrücken und bietet so insbesondere die Möglichkeit der automatischen Konvertierung von Sonderzeichen. Beispielsweise können mit einem normalen Texteditor erstellte Umlaute direkt in das HTML-Format, z.B. *ü* nach *ü*, überführt werden. Die Parametrisierung dieses Filters erfolgt über die Datenbank in der die zu berücksichtigenden Ausdrücke abgelegt sind. Sie werden angepasst an das Einsatzgebiet von JDaphne konfiguriert. Der zusätzlich implementierte Tag-Filter ist speziell auf die Filterung von HTML-Dokumenten ausgelegt. Er dient der Bereinigung der importierten Dokumente von HTML-Layouts. Bei dem Tag-Filter handelt es sich um eine flexible Implementation, da er analog zu regulären Ausdrücken beliebig attributierte Tags verarbeiten kann. Er wird in Abhängigkeit von der Anwendung, mit der die zu filternden Dokumente bearbeitet werden, mit in der Datenbank abgelegten Parameterpaaren initialisiert. Bei diesen Parameterpaaren handelt es sich um HTML-Tag-Namen – jeweils der Name des Tags, der gefiltert werden soll, kombiniert mit dem Namen des Tags, durch den er gegebenenfalls ersetzt werden soll. Auf diese Weise wird ein flexibles Aufbereiten der Dokumente möglich. Erstens können bestimmte Tags, z.B. unerwünschte Formatierungen, komplett aus dem Dokument entfernt werden. Zweitens können attributierte Tags durch Tags ohne Attribute ersetzt werden, ein Prozess der der Vereinheitlichung der Dokumente dient. Drittens können attributierte Tags durch konfigurierte Tags ersetzt werden, z.B. lassen sich so Textzeilen in bestimmten Schriftgrößen automatisiert in Überschriften (Headings) umwandeln. Zusätzlich lassen sich in den Vorgang des Dokumentenimports nach der Filterung noch Validierungsschritte integrieren. Nach der Filterung der

importierten HTML-Dokumente werden diese dem nächsten Verarbeitungsschritt zugeführt. Dabei handelt es sich um Mechanismen, die das importierte Dokument im System greifbar machen.

5.4.1.3 Dokument-Parser

Analog zu den gerade beschriebenen Filtern wurden für den Einsatz in JDaphne spezielle Dokument-Parser implementiert, die in Abhängigkeit vom Format (siehe S. 42) das gezielte Extrahieren von Informationen aus den importierten Dokumenten zur Aufgabe haben. Anders als bei XML-Dokumenten, bei denen es eine Forderung nach Wohldefiniertheit gibt und die über ihre DTD spezifiziert sind, ist der Aufwand für das Parsen von HTML-Dokumenten erheblich. Die Spezifikation sieht zwar auch hier wohldefinierte Dokumente vor, jedoch haben proprietäre Tags den Eingang in viele Dokumente gefunden. Die für JDaphne entwickelten Parser zeichnen sich durch eine hohe Robustheit ihres Parse-Verhaltens aus. Selbst Dokumente, die mit auf dem Document Object Model (DOM) [HNW+01] basierenden Parsern nicht gefasst werden können, werden von JDaphne einwandfrei eingelesen. Zusammen mit den oben beschriebenen Filtern sind so sehr robuste Werkzeuge für den Import von HTML-Dokumenten entstanden, die als Streaming-Lösung realisiert in einem Schritt den vollständigen Import-Vorgang abarbeiten können. Die Informationen, die durch Parser aus den HTML-Dokumenten extrahiert werden, sind im Fall von JDaphne Hyperlinks, Metadaten und Schlagworte.

5.4.1.3.1 Hyperlinks

Für die Bereitstellung von Hyperlink-Management-Funktionen ist es unerlässlich, die Verknüpfungen der Dokumente untereinander zu kennen. Da JDaphne nicht das DOM verwendet, ist hier die Funktionalität eines leistungsfähigen Hyperlink-Parsers entstanden. Wiederum als Streaming-Lösung implementiert zeichnet sich dieser Filter durch seine hochwertigen Link-Erkennungsfähigkeiten auf der einen Seite und der direkt integrierten Ersetzungsfunktion auf der anderen Seite aus. Aufgrund der dateibasierten Verwaltung von Dokumenten in JDaphne werden gefundene Hyperlinks in der Datenbank abgelegt, aber nach wie vor in der Datei belassen. Dazu werden die Hyperlinks durch den Parser in den Dokumenten identifiziert, mit den in dem System bekannten Ressourcen abgeglichen und direkt durch die interne Referenz auf die verlinkte Ressource ersetzt. Im Streaming-Verfahren können somit Hyperlinks gefunden, von dem System interpretiert und gegebenenfalls modifiziert werden. Damit stellt dieser Parser eine wichtige Säule des Hyperlink-Managements dar. Auch hier ist größter Wert auf Robustheit gelegt worden. Die Hyperlink-Policy gibt zwar vor, wie Hyperlinks aussehen müssen, damit sie von dem System verstanden werden, der Parser an sich ist jedoch wiederum dafür ausgelegt, sämtliche Hyperlinks, die ein Web-Browser versteht, ebenfalls fehlerfrei aus einem Dokument herauszuparsen. Einschränkungen der Link-Erkennung des Parsers liegen in denjenigen Hyperlinks, die durch Skript-Sprachen in den Dokumenten selbst verwaltet werden. Hier steht die Komplexität der Auswertung zu dem erzielbaren Resultat in keinem Verhältnis.

5.4.1.3.2 Metadaten

Der Dokument-Parser von JDaphne extrahiert aus den HTML-Dokumenten die im *Dublin Core* Format eingegebenen Metadaten. Diese werden dann von JDaphne mit den systemeigenen Metadaten abgeglichen. Im speziellen werden große Teile der Metadaten von JDaphne automatisiert vergeben und überschreiben die von den Autoren innerhalb des Dokumentes vergebenen.

5.4.1.3.3 Schlagworte

Ergänzt werden die bislang aufgeführten Filter und Parser durch eine analog implementierte Schlagwort-Parser-Variante. Sie liefert die für die Kategorisierung und inhaltliche Interpretation von HTML-Dokumenten – diese ist für sinnvolle Hyperlink-Vorschläge notwendig – erforderliche Grundlage. Als Resultat des Parse-Vorgangs liegen alle in dem Dokument auftretenden Schlagworte mitsamt ihrer Häufigkeit vor. Zusätzlich sind spezielle HTML-Tags, z.B. die Header-Tags, gesondert von dem Parser abfragbar. Dies erlaubt in Addition zum Vorkommen eines Wortes auch über seine Platzierung im Dokument Rückschlüsse auf seine Relevanz für die Beschreibung des Dokumentes.

Alle beim Import eines Dokumentes zum Zuge kommenden Filter und Parser angewendet auf den HTML-Text-Strom liefern JDaphne eine umfassende Kenntnis über das Dokument. Mit wachsendem Umfang des Dokumentes kommt es dabei zu spürbaren Performanceengpässen im System. Für den produktiven Einsatz stellen diese jedoch kein Hindernis dar, da die webgerechte Aufbereitung von HTML-Dokumenten für eine Web-Präsenz nach kurzen, auf eine Bildschirmseite passenden Dokumenten verlangt.

5.4.2 Aktionen und Zustandsänderung von Dokumenten

Bei der Erläuterung der relevanten Verwaltungsstrukturen von JDaphne wurden vier verschiedene Sub-Repositories zur Versionierung von Dokumenten bezüglich ihres Zustands eingeführt. Dieser Abschnitt beschreibt die in JDaphne implementierten Aktionen (Abb. 5.14), die die Modifikationen eines Dokumentes in einem Zustand, den Übergang in ein neues Sub-Repository, bzw. das Entfernen eines Dokumentes aus einem Sub-Repository bewirken.

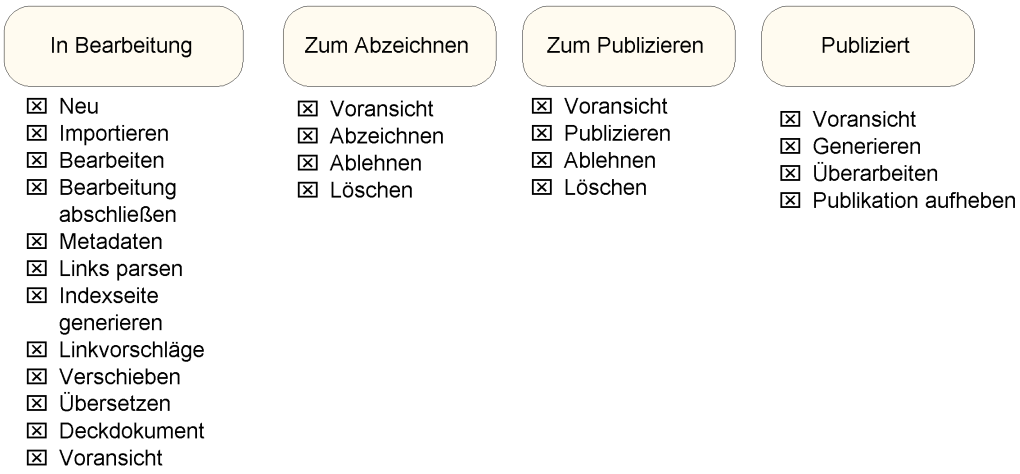

Abbildung 5.14: Eine Übersicht über Dokumentzustände und verfügbare Aktionen.

- *Neues Dokument erstellen*
  Um ein neues Dokument zu erstellen, selektiert der Autor ein Ressort. Hat er die entsprechenden Berechtigungen, kann er in diesem ein neues Dokument erstellen. Dazu wählt er aus den von der Administration konfigurierten Vorlagen eine für sein Vorhaben geeignete aus. Anhand dieses Templates wird ein neues Dokument in JDaphne erstellt. Dieses wird zum einen in der Datenbank referenziert, zum anderen wird die Datei physikalisch in das „In Bearbeitung“ - Sub-Repository von JDaphne aufgenommen.

Das bis auf die Template-Informationen leere Dokument steht nun für die Bearbeitung zur Verfügung.

- *Importieren/Bearbeiten*
  Für die Sicht auf das Repository stellen der „Bearbeiten"-Vorgang und der Import eines Dokumentes einen analogen Prozess dar (Abb. 5.12).

- *Metadaten bearbeiten*
  Nach dem Bearbeiten eines Dokumentes müssen vom Autor seine Metadaten nachgeführt werden. Über die Metadaten-Aktion wird für das selektierte Dokument ein Editor geöffnet, der das Bearbeiten der für das Dokument aus dem Parse-Prozess bekannten Metadaten erlaubt.

- *Hyperlinks-Parsen*
  Damit ein neuer Dokumentenbestand in seiner Hyperlink-Verknüpfungsstruktur erfasst wird, kann manuell die Hyperlink-Umsetzung für diese Dokumente angestoßen werden. Die Aktion „Hyperlinks-Parsen" startet den Hyperlink-Umsetzungs-Mechanismus von JDaphne basierend auf dem Zustand „In Bearbeitung".

- *"Bearbeitung abschließen", bzw. „Zum Abzeichnen" weiterreichen*
  Dokumente im Zustand „In Bearbeitung" stellen die Datenbasis von JDaphne dar. Hier werden Dokumente erzeugt und modifiziert. Alle Dokumente, die von den Autoren als in ihrer Bearbeitung abgeschlossen betrachtet werden, werden bei JDaphne entsprechend dem Mehr-Augen-Prinzip „Zum Abzeichnen" weitergeleitet. Mit dieser Weiterleitungsaktion verbunden ist neben einem erneuten Abgleich mit der Datenbank und der Ausführung diverser Prüfmechanismen, die u.a. im Rahmen der Erläuterung des Hyperlink-Managements beschrieben sind, der Transfer des Dokumentes in das Sub-Repository der Dokumente, deren Bearbeitung abgeschlossen ist, d.h. die „Zum Abzeichnen" bereitliegen. Der Transfer wird im Dateisystem von JDaphne über das Anlegen einer Kopie realisiert. Das Original wird auch nach dem Anlegen der Kopie beibehalten, es stellt die Ausgangsbasis für weitere Modifikationen des Dokumentes durch den Autor dar. Durch den Übergang des Dokumentes in den neuen Zustand wird zeitgleich ein neuer Task generiert, der das Abzeichnen des Dokumentes beinhaltet.

- *Abzeichnen*
  Unter Abzeichnen eines Dokumentes wird im Rahmen von JDaphne die zur Kenntnisnahme und Autorisierung eines Dokumentes durch den berechtigten Inhaber der Rolle *Freigabeberechtigter* bezeichnet. Das Abzeichnen eines Dokumentes kann dabei als mehrstufiger Prozess konfiguriert werden. Es muss dann von einer eingestellten Anzahl von Inhabern der Rolle *Freigabeberechtigter* erfolgen. Haben die entsprechenden Personen das Dokument zur Kenntnis genommen und für in Ordnung befunden, erfolgt dessen Transfer in den Zustand „Zum Publizieren". Damit ist erneut ein Abgleich der Datenbank verbunden. Nachdem der „Abzeichnen"-Task gelöscht wurde, wird ein neuer „Publizieren"-Task aufgebaut, der den Webmaster zum Publizieren des Dokumentes auffordert.

- *Ablehnen*
  Alternativ zum Abzeichnen eines Dokumentes kann der *Freigabeberechtigte* dieses auch als unzureichend erkennen und das Abzeichnen ablehnen. Diese Aktion ist nicht mit dem Transfer eines Dokumentes innerhalb des Repositories verbunden. Bei ihr wird lediglich datenbankseitig vermerkt, dass und warum das Abzeichnen abgelehnt wurde.

Der „Abzeichnen“-Task wird gelöscht und ein „Wiedervorlegen“-Task für den Autor generiert.

- *Publizieren*
  Alle Dokumente, die sich in dem Zustand „Zum Publizieren“ befinden, sind dank des Hyperlink-Managements bezüglich ihrer Hyperlinks und ihrer Inhalte konsistent. Dem Webmaster als der Person mit der Kenntnis über und der Verantwortung für die Web-Präsenz obliegt es, diese Dokumente zu veröffentlichen. Die über den „Zum Publizieren“-Task kenntlich gemachten Dokumente, die sich neu in dem Zustand befinden, werden von dem Webmaster gesichtet und publiziert. Stellt der Webmaster ein Problem mit dem Dokument fest, so kann er die Publikation ablehnen, eine Aktion, die der soeben beschriebenen „Ablehnen-Aktion“ entspricht.

  Die Publikation eines Dokumentes ist mit dem Anlegen einer Kopie verbunden. Das abgezeichnete Dokument wird gelesen und in das Sub-Repository für die publizierten Dokumente kopiert. Dort ist es Teil der Basis des auf der Web-Präsenz veröffentlichten Dokumentenbestandes im Roh-Format. Abgesehen von dem bereits bekannten Datenabgleich mit der Datenbank wird bei der Publizierung gleichzeitig die eigentliche Generierung des Dokumentes, der Export, angestoßen. Wie im nächsten Abschnitt, der sich mit dem Export auseinandersetzt, beschrieben, wird dabei auf der im WWW sichtbaren Web-Präsenz entweder statisch oder dynamisch der Inhalt des Dokumentes mit dem Layout und der Navigationsstruktur zusammengeführt.

- *Publikation aufheben*
  Ist ein Dokument nicht mehr für die Web-Präsenz aktuell oder relevant, wird es aus der Web-Präsenz entfernt. Dieser Prozess wird durch die „Publikation aufheben“-Aktion in die Wege geleitet. Initialisiert durch den *Webmaster* wird ein bei der Betrachtung der Hyperlink-Konsistenzhaltung beschriebener Prozess in Gang gesetzt. Das Dokument selbst wird nach dem Lösen aller Verknüpfungen aus der Web-Präsenz entfernt. Anschließend wird die Datei aus dem Sub-Repository der publizierten Dokumente gelöscht und in das Archiv übernommen.

- *Löschen*
  Das Löschen eines Dokumentes ist nur möglich, wenn es nicht publiziert ist. Ein in dem Zustand „In Bearbeitung“ befindliches Dokument kann von dem *Autor* gelöscht werden. Vor dem Löschen werden die Hyperlink-Management-Funktionalitäten, die die Link-Konsistenz wahren, aufgerufen. Nach einer Archivierung werden sämtliche in der Datenbank vorgehaltenen Informationen über das Dokument gelöscht. Weiterhin werden alle Sub-Repositories, in die das Dokument transferiert wurde, von dem Dokument bereinigt.

- *Generieren*
  Vergleichbar mit der Publikations-Aktion betrifft die Generierungs-Aktion den extern auf der Web-Präsenz sichtbaren Dokumentenbestand. Das Generieren wird von dem *Webmaster* entweder manuell angestoßen oder zeitgesteuert ausgeführt, wenn sich Modifikationen in der Navigationsstruktur oder dem Layout, die sich intern in der Voransicht bewährt haben, auch auf die externe Web-Präsenz auswirken sollen. Dazu werden alle Dokumente aus dem „Publiziert“-Zustand erneut exportiert. Sofern nicht ohnehin dynamisch bei jedem Export generiert, werden so die aktuellen Einstellungen auf der Web-Präsenz sichtbar.

- *Überarbeitung anordnen*
  Sowohl der *Webmaster* als auch der *Freigabeberechtigte* können die Überarbeitung eines Dokumentes anordnen. Bei diesem Vorgang wird für das selektierte Dokument ein neuer Task erstellt. Versehen mit einem Kommentar durch den Auftraggeber wird dieser Überarbeitungs-Task beim *Autor* sichtbar und kann von diesem bis zu dem vorgegebenen Zeitpunkt abgearbeitet werden.

- *Voransicht/Lesen*
  Da JDaphne über keine eigene Eingabekomponente verfügt, stellt die Voransicht der Dokumente eine zentrale Funktionalität dar. Selektiert der Benutzer ein Dokument für die Voransicht, so wird dieses vom System aus dem passenden Zustand geholt und in der Voransicht-Komponente wie für den externen Export mit Layout und Navigation versehen.

- *Verschieben*
  Das Verschieben eines Dokumentes oder Verzeichnisses ist eine zustandübergreifende Aktion. Ihre Durchführung wird im folgenden bei der Beschreibung des Hyperlink-Managements dargestellt. Die Aktion „Verschieben“ wirkt auf das gesamte Repository. Damit müssen die sich daraus ergebenden Modifikationen am Dokumentenbestand und der Verknüpfungsstruktur in allen Zuständen berücksichtigt werden.

- *Deckdokument definieren*
  Aufgrund ihrer exponierten Stellung in einem Ressort kommt den Deckdokumenten in JDaphne eine besondere Bedeutung zu. Aus Datenbanksicht bedeutet das Auswählen eines (neuen) Deckdokumentes lediglich das (Um-)Setzen eines entsprechenden Flags. In der dynamischen Voransicht werden damit die Hyperlinks automatisch richtig transponiert. Für die Verwaltung der statischen Web-Präsenz sind die Auswirkungen deutlich größer, da hier durch diese Modifikation eine erneute Generierung des betroffenen Dokumentes in Gang gesetzt und zudem die Hyperlink-Management-Komponente aktiv wird.

- *Deckdokument erstellen*
  Alternativ zu der manuellen Erstellung eines Deckdokumentes und späteren Selektion als solches verfügt JDaphne über eine Automatik zur Generierung eines solchen Dokumentes. Das Ergebnis der Aktion „Deckdokument erstellen“ basiert auf dem Dokumentenbestand im „In Bearbeitung“-Zustand. Das Deckdokument erhält eine Verknüpfung zu jedem in dem Ressort verfügbaren Dokument. Durch Konfiguration dieser Aktion kann festgelegt werden, ob lediglich HTML-Dokumente oder auch andere Dokumentformate berücksichtigt werden sollen. Dokumente werden in der Übersicht anhand ihres Titels und Untertitels gelistet. Auch andere Meta-Informationen können in die Liste aufgenommen werden. Der Aufruf dieser Aktion erzeugt ein neues Dokument im „In Bearbeitung“-Zustand, das für die weitere Modifikation durch den *Autor* zur Verfügung steht.

- *Dokument übersetzen (Sprachversion erstellen)*
  JDaphne verwaltet die Sprachversionen von Dokumenten in der Form: Dokument *D'* ist eine Übersetzung in Sprache *S* von Dokument *D*. Damit diese Beziehung automatisiert aufgebaut wird und nicht manuell nachgeführt werden muß, kann für jedes Dokument die Aktion „Dokument übersetzen“ aufgerufen werden. Diese führt jedoch nicht wie der Name vermuten lässt eine Übersetzung des Dokumentes durch. Sie legt in JDaphne eine Kopie des Dokumentes an und setzt dessen Eigenschaften auf die neue Sprache um,

z.B. in dem Sprachkürzel für den Dateinamen. Weiterhin generiert der Aufruf dieser Aktion einen neuen Task, einen „Übersetzen"-Task. Das kopierte Dokument liegt für einen zur Übersetzung kompetenten *Autor* in dem „In Bearbeitung"-Zustand vor. In der Datenbank wird die Beziehung zwischen den Dokumenten bezüglich ihres Ursprungs abgelegt.

- *Linkvorschläge einholen*
  Die Qualität einer Web-Präsenz steigt mit der Qualität der Verknüpfungen der Dokumente untereinander. JDaphne bietet über die Hyperlink-Management-Komponente die Möglichkeit, Hyperlinks in den Dokumenten zu generieren. Dieser Prozess wird durch die Aktion „Linkvorschläge einholen" gestartet. Je nach selektierter Option werden dabei die bereits von dem Autor entsprechend der Hyperlink-Syntax in JDaphne vorgegebenen Hyperlinks gesetzt oder alternativ eine Liste möglicher Hyperlinks in einem separaten Fenster angezeigt. Im ersten Fall werden die vorgeschlagenen Hyperlinks in das Dokument in dem „In Bearbeitung"-Zustand geschrieben und müssen von dem *Autor* in einem normalen Bearbeitungsprozess in der externen Editor-Anwendung weiterverarbeitet werden. Im anderen Fall können die vorgeschlagenen Links aus dem separaten Fenster in die Eingabekomponente, mit der das Dokument gerade bearbeitet wird, kopiert werden.

5.4.3 Hyperlink-Management

Für die mit JDaphne aufgebauten Web-Präsenzen ist Hyperlink-Konsistenz eine der Basisanforderungen. JDaphne integriert für die Hyperlink-Management-Fähigkeit zum einen die auf Seite 82 beschriebene Hyperlink-Management-Komponente, zum anderen sind eigenständige Algorithmen entwickelt worden, die bezüglich der Behandlung der Verknüpfungen zustandsübergreifend speziell an den internen Dokumenten-Workflow angepasst sind. Die bereits beschriebene Hyperlink-Management-Komponente wird von JDaphne eingesetzt, um bei Dokument-Verschiebungen die Konsistenz zu erhalten. Weiterhin können die auf Case-Based-Reasoning beruhenden Funktionalitäten der Hyperlink-Vorschläge genutzt werden. Die speziell für JDaphne entwickelten Mechanismen zur Hyperlink-Verwaltung werden im folgenden Abschnitt vorgestellt. Dazu wird die interne Hyperlink-Syntax erläutert und die Speicherung der Hyperlinks in der Datenbank erklärt.

5.4.3.1 Auswerten neu eingespielter Dokumente

JDaphne ist von seiner Konzeption her so angelegt, dass es alle Hyperlinks aus den eingespielten Dokumenten extrahiert (s.o.) und auf die Verlinkung der Web-Präsenz anpasst. Dazu werden die in dem Parse-Prozess gefundenen Hyperlinks analysiert und in interne, d.h. Verknüpfungen innerhalb der Web-Präsenz, und externe Hyperlinks, d.h. Verweise auf andere Ressourcen im Internet, unterschieden. Die externen Links werden lediglich auf ihre Funktionstüchtigkeit überprüft. Links auf interne Ressourcen werden entsprechend der internen Syntax umgesetzt. Innerhalb des Systems hat jede Ressource eine eindeutige Kennung (Dokumentennummer). Beim Bearbeiten und beim Importieren sind die Hyperlinks über Dateinamen realisiert. Deren automatisierte Umsetzung auf Dokumentennummern und Ressortkürzel führt JDaphne nach den in der Link-Policy abgelegten Regeln durch. Die Notwendigkeit einer solchen Policy erwächst aus der dynamischen Konzeption des Systems. Sowohl Dokumentennamen als auch Verzeichnisnamen sind bei JDaphne nur temporären Charakters und können jederzeit modifiziert werden. Die über die Policy vorgegebenen Regeln erleichtern den Autoren den Einsatz von Hyperlinks.

1. In Daphne werden die Dokumente anhand von Dokumentennummern verwaltet. Daher kann ein Link einfach aus einer Dokumentennummer bestehen, z.B. `<a href="342.html">`. Der Suffix kann, muss aber nicht, angegeben werden.

2. Die Namen der Dateien in einem Ressort sind eindeutig. Besteht ein Link nur aus einem Dokumentennamen, z.B. `<a href="internet.html">`, so ist dieser Link korrekt, wenn es in dem Ressort, in das diese Datei eingespielt wird, ein Dokument mit diesem Namen gibt. Ist der Dateiname global eindeutig, so wird der Link auf die zugehörige Datei umgesetzt.

3. Das System unterstützt relativen Links nur in einer formalen Art und Weise. Alle Links müssen, sofern nicht der in 1. beschriebene Weg gewählt wird, mit dem Dokumentennamen und dem Ressortkürzel eingegeben werden. Ein Beispiel dafür wäre: `<a href="imag/bild1.jpg">`, wobei das Ressortkürzel „imag“ ist.

4. Links auf Deckseiten (index.html) werden in der Form „Ressortkürzel“ + „/“ erwartet. Beispielsweise wäre der Hyperlink auf das Verzeichnis mit dem Ressortkürzel „demo“ in der Form `<a href="demo/">` zu schreiben.

5. Der Hyperlink auf das Home-Verzeichnis, d.h. auf die Homepage, kann alternativ zu 4. auch in der Form „/“ geschrieben werden.

5.4.3.2 Interne Hyperlink-Syntax

Nachdem ein Dokument in das System einspielt wurde, parst der Link-Erkennungs-Mechanismus das Dokument und modifiziert es gegebenenfalls, um seine Hyperlinks in die interne Link-Syntax zu übertragen. Diese unterscheidet sich nur marginal von der normalen Hyperlinksyntax. Konkret werden alle Hyperlinks für den internen Gebrauch innerhalb der nicht-exportierten Versionen der Dokumente in die Form „Dokumentennummer plus spezielle Hyperlink-Information für den Autor“ umgesetzt. Beispielsweise wird so aus dem Hyperlink `<A HREF="demo/Mein Dokument.html">Ein Hyperlink auf mein Dokument</A>` der folgende interne Hyperlink zusammengesetzt:

```
<A HREF="732.html" |JDAPHNEINFO:ORIGINALLINK="demo/Mein Dokument.html"|>
Ein Hyperlink auf mein Dokument</A>
```

Das Dokument mit dem Namen „Mein Dokument.html“ in dem Verzeichnis mit dem Kürzel „demo“, auf das der Link verweist, hat beim Einspielen in JDaphne die interne Dokumentennummer 732 bekommen. Durch diese ist es innerhalb von JDaphne eindeutig referenzierbar. Um die Lesbarkeit der Link-Ersetzungen zu gewährleisten, wurde das JDAPHNEINFO-Element konzipiert, welches für den internen Dokumentenbestand in den Anker-Tag eingebaut wird. Dieses enthält weiterhin die für den Autor lesbare Information, ob der Link richtig umgesetzt werden konnte. Falls nicht, wird der Link als „inkonsistent“, d.h. broken gewertet und das JDAPHNEINFO-Element enthält den Text „Broken-Link“. Analog werden zu den Verweisen auf Dokumente anhand der Dokumentennummern Verweise auf Verzeichnisse über die Angabe des Ressortkürzels realisiert.

5.4.3.3 Speicherung von Hyperlinks in der Datenbank

Zusätzlich zu der Referenz in den Dokumenten selbst, die nach dem Parsen erhalten bleibt, werden alle von JDaphne erkannten internen Hyperlinks in der internen Datenbank des Systems abgelegt. Diese Hyperlink-Informationen stellen die Basis für das Hyperlink-Management von JDaphne dar. Sie haben die Attribute (Abb. 5.15) Ausgangspunkt und

Zielpunkt des Hyperlinks durch Dokumentennummern bzw. Ressortkürzel identifiziert. Vervollständigt werden die Datensätze durch den Status, d.h. das Sub-Repository des geparsten Dokumentes, einen Zeitstempel und den Link-Typ, d.h. die Angabe, ob es sich um einen Anker oder eine Source handelt. Hyperlinks, die von JDaphne keiner internen Ressource

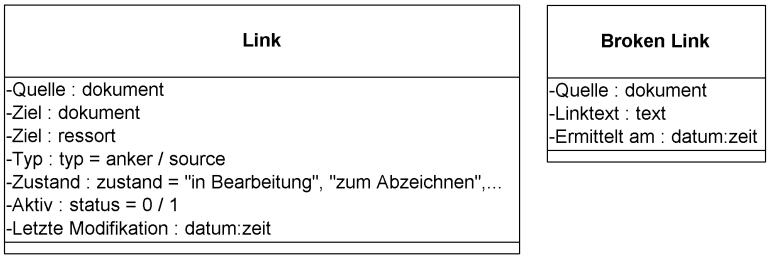

Abbildung 5.15: Die Attribute eines Hyperlinks. Rechts, die im Fall eines Broken-Links verwaltete Information.

zugeordnet werden können, werden separat verwaltet. In diesem Fall wird statt der Zielangabe der nicht zuzuordnende Hyperlink im Klartext abgelegt. Auf diese Weise kann dem Benutzer jederzeit unabhängig von den in den Dokumenten niedergelegten Daten visualisiert werden, welche Hyperlinks nicht zugeordnet werden konnten. Bei jeder Statusänderung eines Dokumentes müssen die in der Datenbank abgelegten Hyperlinkinformationen aktualisiert werden. Dieser Prozess ist zwar aufwendig, stellt aber sicher, dass der Datenbestand in der Datenbank konsistent mit dem Dokumentenbestand in dem Repository ist. Daher kann anhand der Informationen in der Datenbank jederzeit bestimmt werden, wie die Verknüpfungsstruktur der Dokumente in den Zuständen aussieht und auf diese Weise können die Hyperlink-Verwaltungs-Fähigkeiten von JDaphne gewährleistet werden.

5.4.3.4 Hyperlink-(De-)Aktivierung

Zentrale Hyperlink-Management-Aufgabe von JDaphne ist die Konsistenzhaltung der Verknüpfungsstruktur von der Bearbeitung der Dokumente über die Qualitätssicherung, den Publikationsprozess bis hin zum Entfernen der Inhalte aus der Web-Präsenz. Die Herausforderung bei dieser Aufgabe liegt darin, dass nicht alle Dokumente zugleich Zustandsübergänge durchführen (Abb. 5.16). Während ein Dokument auf die Abzeichnung wartet, ist das dieses Dokument referenzierende Dokument vielleicht schon im Internet sichtbar gemacht – eine Situation, die ohne ein angepasstes Hyperlink-Management direkt zu einem Broken-Link führt. Wie in der Konzeption erläutert, wird bei JDaphne dieses dokumentenstatus-bedingte Konsistenz-Problem durch die Aktivierung und Deaktivierung von Hyperlinks gelöst. Ausgangsforderung (siehe S. 109) für die Aktivierungstechnologie ist die Hyperlink-Konsistenz im „in Bearbeitung“-Zustand, d.h. ein Dokument, das diesen Zustand verlässt, enthält keine Broken-Links mehr. Dieser Forderung genügen die in JDaphne möglichen Aktionen, da vor jedem Zustandsübergang die Dokumente auf Broken-Links überprüft werden. Folglich kann davon ausgegangen werden, dass ein Dokument, das von dem Zustand „In Bearbeitung“ in den Status „Zum Abzeichnen“ überführt wird, nur Referenzen auf Dokumente enthält, die in JDaphne bekannt sind. Weiterhin ist damit sicher, dass die referenzierten Dokumente mindestens in dem für die Dokumente „In Bearbeitung“ zuständigen Sub-Repository verfügbar sind.
Dementsprechend werden bei dem Übergang eines Dokumentes *D* von einem Zustand *Z1* in

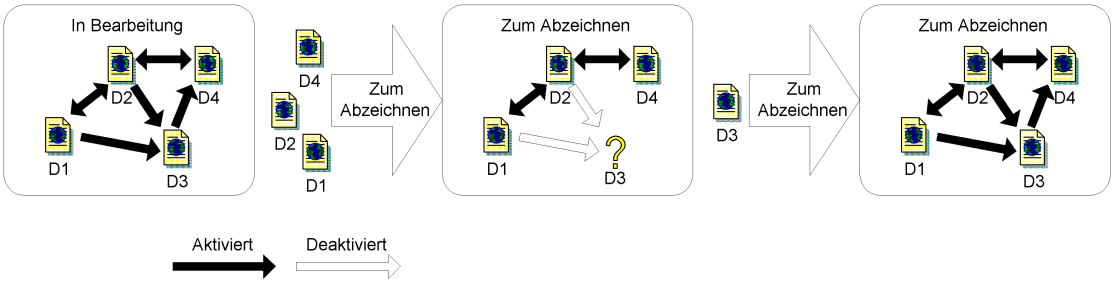

Abbildung 5.16: Visualisierung der Hyperlink-Aktivierungs-Problematik am Beispiel des „In Bearbeitung“- und des „Zum Abzeichnen“-Zustandes. Dokument D3 wird erst später nachgezogen. Für die Dauer, in der D3 nicht im entsprechenden Zustand ist, werden die Hyperlinks deaktiviert.

einen anderen Zustand *Z2*: $D(Z1) \Rightarrow D(Z2)$, in einem ersten Schritt alle von diesem Vorgang durch von D ausgehende Hyperlinks betroffenen Dokumente ermittelt:

- $R(Z1, D(Z1))$: Menge aller Dokumente im Zustand *Z1*, die von $D(Z1)$ referenziert werden.
- $R(Z2, D(Z1))$ Menge aller Dokumente im Zustand *Z2*, die von $D(Z1)$ referenziert werden, aber bereits im Zustand *Z2* sind.

Dies ist anhand der in der Datenbank abgelegten Dokument-Zustands- und Hyperlink-Informationen für *Z1* und *Z2* über eine Abfrage aller referenzierenden und referenzierten Dokumente (vgl. Abb. 5.15) in den Link-Tabellen möglich. Im nächsten Schritt wird überprüft, ob der Zustandsübergang des Dokumentes aus Hyperlink-Sicht erlaubt ist. Dies ist er dann, wenn keine Source-Verknüpfungen von einer Deaktivierung in dem Dokument betroffen sind. Diese als Bestandteile des Dokumentes gewerteten Referenzen verhindern den Zustandsübergang, welcher im Falle seines Eintretens wertlose Dokumente entstehen ließe.

- $S(Z1, D(Z1))$: Menge aller Dokumente im Zustand *Z1*, die von $D(Z1)$ als Source eingebunden werden und $S(Z1, D(Z1)) \subseteq R(Z1, D(Z1))$.
- $S(Z2, D(Z1))$: Menge aller Dokumente im Zustand *Z2*, die von $D(Z1)$ als Source eingebunden werden, aber bereits im Zustand *Z2* sind, wobei wieder gilt $S(Z2, D(Z1)) \subseteq R(Z2, D(Z1))$.

Die Abbruchbedingung für den Zustandsübergang $D(Z1) \Rightarrow D(Z2)$ lautet damit:

$$S(Z1, D(Z1)) \neq S(Z2, D(Z1)) \tag{5.1}$$

Erlaubt ist der Übergang somit für ein HTML-Dokument, wenn alle referenzierten Sourcen bereits in dem Zustand *Z2* verfügbar sind:

$$S(Z1, D(Z1)) \equiv S(Z2, D(Z1)) \tag{5.2}$$

Ist die Zustandsänderung erlaubt, geht D in den Zustand *Z2* über, wobei alle von $D(Z1)$ ausgehenden Hyperlinks für $D(Z2)$ übernommen werden und den Status „aktiv“ erhalten. Im nächsten Schritt erfolgt die Aktivierung der durch diesen Zustandsübergang $D(Z1) \Rightarrow D(Z2)$ betroffenen Hyperlinks. Dies ist die Menge aller Hyperlinks, die im Zustand *Z2* auf $D(Z2)$

verweisen und bislang deaktiviert waren. Abschließend wird die Menge der von $D(Z2)$ über Hyperlinks referenzierten Dokumente $R(Z2, D(Z2))$ mit der Menge $M(Z2)$ der Dokumente im Zustand $Z2$ verglichen. Ermittelt wird die Menge $H(Z2) = R(Z2, D(Z2)) \cap M(Z2)$. Alle von $D(Z2)$ ausgehenden Hyperlinks, die auf $H(Z2)$ verweisen, werden deaktiviert.

Betrifft der Zustandsübergang eines Dokumentes dessen Entfernung aus einem Sub-Repository (z.B. durch Löschen), so ändert sich der Algorithmus insofern, als dass nun eine andere Dokumentenmenge von der Hyperlink-Aktivierung betroffen ist. Alle diejenigen Hyperlinks, die auf das „gelöschte“ Dokument in dem Sub-Repository aus dem das Dokument entfernt wird, zeigen, werden deaktiviert. Sind alle Referenzen auf das Dokument deaktiviert, so kann es ohne Auswirkungen auf die Hyperlink-Konsistenz entfernt werden.

Aus implementatorischer Sicht erfolgt die (De-)Aktivierung von Hyperlinks im internen Bereich virtuell. Lediglich die Datenbank wird aktualisert, die Dokumente selbst werden nicht modifiziert. Bei der Voransicht werden dynamisch aktuell in einem Zustand deaktivierte Hyperlinks durch Markierungen visualisiert (Abb. 5.17), die dem Benutzer das Fehlen eines Dokumentes anzeigen. Anders arbeitet die Hyperlink-(De-)Aktivierung auf der extern

Abbildung 5.17: Screenshot: Deaktivierte Hyperlinks in der dynamischen Voransicht.

sichtbaren Dokumentenbasis der Web-Präsenz. Dort werden deaktivierte Hyperlinks bei dem Generierungsprozess vollständig aus den Dokumenten eliminiert, indem wahlweise der komplette Anker-Tag mit dem dazwischen stehenden Text oder aber nur der Anker entfernt wird. Die Änderung des Aktivierungsstatus' eines publizierten Dokumentes in der statischen Web-Präsenz wird über die Neu-Generierung dieses Dokumentes für die Web-Präsenz umgesetzt.

5.4.3.5 Verschieben von Dokumenten und Ressorts

Eine für das Hyperlink-Management im Normalfall sehr aufwendige Aufgabe ist das Verschieben von Dokumenten oder Ressorts. Für JDaphne sind hier über die nummernbasierte Referenzierung von Dokumenten und die eindeutigen Ressortkürzel Abhilfen geschaffen worden, die die Verschiebung sehr simplifizieren. Bei dynamischen Exporten wird abgesehen von dem Sub-Repository übergreifenden Verschiebevorgang keinerlei Aktivität notwendig. Bei statischen Exporten muss nach der Verschiebung eines Dokumentes oder Ressorts beim Publizieren der Modifikation ein Abgleich des Dokumentenbestandes der Web-Präsenz durchgeführt werden. Auf der Basis der in JDaphne's interner Datenbank gespeicherten Verknüpfungsinformationen werden nach der Neuerzeugung der verschobenen Datei(en) alle diese referenzierenden Dokumente mit der aktualisierten Link-Information neu generiert. Dies ist ein Vorgang, der bei dem Verschieben von Ressorts, je nach deren Position in der Struktur, auf eine vollständige Neugenerierung der Web-Präsenz hinauslaufen kann. Nach dem Generierungsprozess wird das durch Neuerzeugung an der Zielposition bereits verfügbare Dokument von seiner ursprünglichen Position gelöscht. So wird gewährleistet, dass zu keiner Zeit während des Verschiebe-Vorgangs Referenzen „ins Leere weisen".

5.4.4 Dokument-Export

Beim Dokument-Export werden Content, Layout und Navigation (siehe S. 56) zusammengeführt. Die mit neutralem Layout vorgehaltenen Contents werden gelayoutet und mit entsprechender Navigation versehen publiziert. Der Export bzw. der Publikationsvorgang kann bei JDaphne „statisch" oder „dynamisch" erfolgen. Im ersten Fall findet dieser Vorgang einmalig im Rahmen einer Dokument-Generierung bei einer Änderung des Dokumentenbestandes statt. Das Dokument liegt danach fertig zur Auslieferung durch einen Web-Server vor. Im zweiten Fall, bei einem dynamischen Export, ist der Web-Server in den Export-Vorgang involviert. Fordert ein Browser ein bestimmtes Dokument an, so generiert der Web-Server mittels des auf den Export-Server zugreifenden Servlets die gewünschte Datei aus den oben genannten Komponenten. Dynamischer Export wird im Allgemeinen eingesetzt, wenn die Dokumente sich mit hoher Frequenz inhaltlich ändern oder wenn eine Personalisierung der Seite für den Benutzer stattfinden soll. Ein dynamischer Export belastet die Performanz eines Web-Servers gegebenenfalls stark, auf jeden Fall aber stärker als statisch vorliegende Dokumente. Um z.B. temporäre Einblendungen in allen statischen Dokumenten sichtbar zu machen, müssen diese vollständig neu generiert werden.

5.4.4.1 Templates

JDaphne kennt zwei Typen von Templates: zum einen gibt es die konventionellen Templates, die den Autoren werden als Vorlage, in die sie ihre Texte eintragen können, angeboten. Zum anderen gibt es sogenannte Export-Templates; in ihnen ist definiert, wie das Dokument beim Export aus den Bausteinen Content, Layout und Navigation zusammengebaut werden soll.

5.4.4.1.1 Standard Templates

Bei den Standard Templates handelt es sich um Dokumentvorlagen, die von den Autoren als Grundgerüst für neu zu erstellende Dokumente einsetzt werden. Hier ist eine Auswahl verschiedener Templates verfügbar. Von diesen am häufigsten genutzt wird das Standard Template, welches lediglich das HTML-Grundgerüst (siehe S. 43) sowie einen ersten Überschrift-Tag enthält. Mit dem Template wird innerhalb von JDaphne festgelegt, um welchen Dokumenten-

typ es sich handelt, eine Eigenschaft, die für dynamisch generierte Dokumentenübersichten auf der Web-Präsenz ausgenutzt wird.

5.4.4.1.2 Export Templates

Für jedes Ressort bzw. jeden Dokumententypen von JDaphne kann ein eigenes Template entworfen werden, das den Aufbau der Dokumente beschreibt. In dem Template stehen Formatierungsanweisungen, die definieren, wo der Inhalt des Dokumentes platziert wird und nach welchen Vorschriften die Navigationsstruktur aufgebaut werden soll. So kann bei der Generierung der statischen Dokumente für die externe Web-Präsenz die Verzeichnisstruktur in geeigneter Form als Navigationselement in jede Seite eingebunden werden. Auch die Integration von anderen Informationen zur Generierungszeit, z.B. das Generierungsdatum, ist auf diese Weise möglich. Bei den Templates in JDaphne handelt es sich somit um Formatierungsschablonen mit selbst definierten Datenfeldern, welche beim Exportvorgang mit Daten aufgefüllt werden. Entsprechende Daten sind z.B. das Dokument (der Content) selbst, der Link auf die Einstiegsseite oder das aktuelle Ressort. Diese Datenfelder, eine Übersicht folgt am Ende dieses Abschnitts, ermöglichen das freie Platzieren von dem System bekannten Informationen innerhalb der zur Verfügung stehenden Dokumentenebene. Für alle Datenfelder können Formatierungsvorschriften vergeben werden. So ist eine freie Layoutkreation durch den Designer, die zudem durch Cascading Style Sheets (vgl. Seite 44) ergänzt werden kann, problemlos möglich.

Grundsätzlich handelt es sich bei den Templates um HTML-Dokument ohne Inhalt. Lediglich die Aufteilung des Dokumentes in Navigation, Inhalt und Sonstiges (z.B. Disclaimer) wird vorgegeben. Die Datenfelder, die beim Exportvorgang mit Daten gefüllt werden, haben folgendes Format:

```
<JDAPHNETAG>[Formatierungsanweisungen]<DATA>
[Formatierungsanweisungen-Ende]<JDAPHNETAG-Ende>
```

Dabei wird der DATA-Tag durch die eigentlichen Daten, z.B. den Link auf das Home-Verzeichnis, ersetzt. Diese Form der Formatierung bewährt sich insbesondere bei der Ausgabe von Listen, wie z.B. der Liste der Unterverzeichnisse. In diesem Fall werden die zwischen den JDAPHNE-Tags stehenden Formatierungsanweisungen für jedes Listenelement separat vergeben. Dies ermöglicht z.B. die Verwendung von Ikons vor den Verzeichnisnamen etc. Tabelle 5.2 gibt eine Übersicht über verfügbare Datenfelder.

Tag	Bedeutung	Bemerkung
`<JDAPHNE_DOC>`	An dieser Stelle wird das Dokument platziert.	Dient der Einbindung des Contents in Tabellenstrukturen.
`<JDAPHNE_HOME>`	An dieser Stelle einen Link auf das Home-Verzeichnis setzen.	Dient der Navigation innerhalb der Web-Präsenz.
`<JDAPHNE_RESSORT>`	An dieser Stelle einen Link auf das aktuelle Ressort setzen.	Wird benötigt, um innerhalb des Verzeichnisses jederzeit auf das Deckdokument wechseln zu können.

Tag	Bedeutung	Bemerkung
`<JDAPHNE_SUB_RESSORT>`	An diesre Stelle eine Liste mit den Unterressorts des aktuellen Ressorts setzen.	Die Formatierung wird für jeden Eintrag übernommen. Diese Liste ist für die Navigation wichtig.
`<JDAPHNE_TOP_RESSORT>`	An dieser Stelle eine Liste mit Ressorts aus der obersten Ebene setzen.	Kann für die Top-Navigationsleiste eingesetzt werden.
`<JDAPHNE_TOP_RESSORT_SELECTED>`	An dieser Stelle das Ressort auf oberster Ebene, das in dem Zweig des aktuell selektierten Ressorts ist setzen.	Dient unterschiedlichen Darstellungen selektierter Zweige.
`<JDAPHNE_TOP_RESSORT_BEFORE_SELECTED>`	An dieser Stelle eine Liste aller Ressorts auf oberster Ebene, die vor dem obersten Ressort stehen, das in dem Zweig des aktuell selektierten Ressorts ist setzen.	Dient unterschiedlichen Darstellungen selektierter Zweige.
`<JDAPHNE_TOP_RESSORT_AFTER_SELECTED>`	An dieser Stelle eine Liste aller Ressorts auf oberster Ebene, die nach dem obersten Ressort stehen, das in dem Zweig des aktuell selektierten Ressorts ist setzen.	Dient unterschiedlichen Darstellungen selektierter Zweige.
`<JDAPHNE_PATH>`	An diese Stelle eine Liste aller Ressorts in dem Zweig des selektierten Ressorts, die zwischen der obersten und der aktuellen Ebene liegen setzen.	Dient der Präsentation des aktuellen Zweiges.
`<JDAPHNE_SIBLING_BEFORE_SELECTED>`	Benachbarte Ressorts auf der gleichen Ebene wie das selektierte, die in der Reihenfolge vor dem selektierten dargestellt werden müssen, setzen.	Dient der Präsentation der aktuellen Ressortebene innerhalb des Zweiges.

Tag	Bedeutung	Bemerkung
<JDAPHNE_SIBLING_AFTER_SELECTED>	Benachbarte Ressorts auf der gleichen Ebene wie das selektierte, die in der Reihenfolge nach dem selektierten dargestellt werden müssen, setzen.	Dient der Präsentation der aktuellen Ressortebene innerhalb des Zweiges.
<JDAPHNE__TIME>	Die aktuelle Zeit.	Vorrangig für die dynamische Generierung.
<JDAPHNE__DATE>	Das aktuelle Datum.	Vorrangig für die dynamische Generierung.
<JDAPHNE_MODIFICATION_DATE>	Das Datum der letzten Änderung des Inhalts.	
<JDAPHNE_GENERATION_DATE>	Das Datum der letzten Generierung des Dokumentes.	

Tabelle 5.2: Datenfelder in JDaphne's Export-Templates.

5.4.4.2 Tag-Ersetzung

Um das Layout von HTML-Dokumenten zu beeinflussen, hat sich die Ersetzung von bereits in den Dokumenten vorhandenen Formatierungs- und Strukturierungstags (siehe S. 43) bewährt. Die Tag-Ersetzung war einer der ersten Mechanismen, der zum Layouten von Dokumenten im Redaktionssystem eingesetzt wurde. Bei der Tag-Ersetzung in JDaphne gibt es für jedes Ressort eine eigenständige Liste von Textfragmenten z.B. Tags mit entsprechenden Ersetzungen. Beispielsweise kann in dieser Liste die Ersetzung der Formatierung von Überschriften der Kategorie „H1“ durch eine bestimmte Schriftart, z.B. der Größe 3 (Font size=3), definiert sein. In Listen können anstelle von den Standard-Symbolen Grafiken verwendet werden etc. Sinnvoll ist der Tag-Ersetzungsmechanismus insbesondere in Kombination mit Cascading Style Sheets (siehe S. 44). Dann lassen sich z.B. Paragraphen-Markierungen „<p>“ durch die mit der entsprechenden Layoutklasse versehenen Anweisungen „<p id=afett>“ ersetzen und ermöglichen so einheitliche Formatierungen. Basierend auf dieser Technik können Layouts nach Belieben noch nach der Produktion der Dokumente modifiziert werden. Kombiniert mit regulären Ausdrücken ermöglicht die Tag-Ersetzung ergänzende Layoutarbeiten an den Dokumenten. Insbesondere können auf diese Weise auch im Nachinein von den Autoren verwendete, im Layout aber unerwünschte Tags eliminiert werden, sofern sie nicht bereits von Importfiltern substituiert wurden.

5.4.4.3 Dokumentlayout

JDaphne unterstützt die beiden beim Dokumentlayout einer Web-Präsenz (siehe S. 59) beschriebenen Möglichkeiten: Beim Export können sowohl Tabellenstrukturen als auch Frame-Sets generiert werden. Im ersten Fall werden die Navigationsleisten zu der eigentlichen Inhalts- bzw. der Übersichtsseite hinzugefügt, indem sie in einer Tabelle neben dem Dokument platziert werden. Wird ein Frame-Set benötigt, so generiert JDaphne für jedes exportierte Dokument ein eigenständiges. Auf diese Weise können die Navigationsleisten in eigenständigen Frames untergebracht werden.

5.4.5 Statischer Export

JDaphne bietet sowohl die Option des statischen als auch des dynamischen Exports. Beim statischen Export werden die mit Layout und Navigation integrierten Inhalte in das Dateisystem des Web-Servers gespeichert. Versehen mit den entsprechenden Berechtigungen können sie von dort mit Web-Browsern abgefragt werden. Der Übersichtlichkeit halber und um die Web-Statistiken informativer zu gestalten, erfolgt der Export in eine Verzeichnishierarchie. Der statische Export ist diejenige Variante, die mit deutlich weniger Aufwand auf der Seite des Web-Servers bereitgestellt werden kann als dies beim dynamischen Export möglich ist. Der Rechner, der die Web-Präsenz im Internet verfügbar macht, muss lediglich über einen Web-Server verfügen. Sofern auf die von JDaphne gebotene Suche verzichtet wird, sind keine weiteren Installationen für ein Back-End notwendig. Damit die Dokumente im Dateisystem abgelegt werden können, müssen sie zuvor generiert werden. Änderungen im Dokumentenbestand haben aufgrund der Hyperlinks zwischen den Dokumenten Inkonsistenzen (Broken-Links) zur Folge. Im Normalfall wird dieses Problem durch das „Staging“ gelöst: Nach einer Modifikation im Dokumentenbestand werden sämtliche Dokumente neu generiert. Dieser Ansatz hat, sofern der Generierungsprozess das Entstehen von Broken-Links verhindert, eine konsistente Web-Präsenz zur Folge. Mit JDaphne ist das komplette Neugenerieren der Web-Präsenz jederzeit möglich. Ab einer bestimmten Dokumentenzahl dauert der Vorgang der Neugenerierung so lange, dass die Zahl der Neugenerierungen pro Zeiteinheit so weit beschränkt ist, dass eine zeitnahe Publikation von Dokumenten verhindert wird. Deshalb bietet JDaphne einen mehrstufigen Generierungsprozess, der es ermöglicht, einzelne Dokumente oder Dokument-Mengen zu publizieren, ohne die vollständige Web-Präsenz neu generieren zu müssen. JDaphne generiert zusätzlich zu der Menge der publizierenden Dokumente die im Umfeld von dieser Modifikation betroffenen Dokumente (Hyperlink-Aktivierung, vgl. S. 136) auch neu. Das Verfahren, das JDaphne zu diesem Zweck einsetzt, ist dreistufig. Es basiert auf dem Sub-Repository der publizierten Dokumente und der Hyperlink-Informationen der Dokumente in diesem Zustand und in generierter Form.

Sei M die Menge der vom *Webmaster* neu publizierten Dokumente. Alle Dokumente aus M werden einem virtuellen Generierungsprozess unter Nutzung der Export-Templates ausgesetzt. Dieser Prozess ermittelt die durch Hyperlinks aus Content und Navigation der Dokumente aus M erwachsene Menge $R(M)$ der von den neu publizierten Dokumenten referenzierten Dokumente. Ein Abgleich mit der Menge P der Dokumente im Zustand „Publiziert“ liefert die Dokumente, die von den zu generierenden Dokumenten nicht referenziert werden dürfen, da sie nicht publiziert sind. Die Menge N der Dokumente, auf die zeigende Hyperlinks deaktiviert werden müssen, ist die Menge aller Dokumente D für die gilt $D \in R(M)$, $D \notin R(M) \cap P$. Des weiteren ergibt ein Vergleich der noch deaktivierten Hyperlinks in bereits generierten Dokumenten mit der M die Menge $A(M)$ der aufgrund zu aktivierender Hyperlinks auf Dokumente aus M neu zu generierenden Dokumente. Somit bildet sich die Menge G der zu generierenden Dokumente als $G = M \cup A(M)$. Diese werden von JDaphne neu generiert und auf dem Web-Server platziert.

Analog arbeitet das hier beschriebene Verfahren, wenn die Publizierung von Dokumenten aufgehoben wird. Sei L die Menge der Dokumente, die von der Web-Präsenz genommen werden sollen. In einem ersten Schritt wird die Menge X der Dokumente aus P bestimmt, die auf Dokumente aus L verweisen. Die Hyperlinks der Dokumente aus X, die auf Dokumente aus L zeigen, werden deaktiviert. Unter Nutzung der aktualisierten Hyperlink-Aktivierungs-Informationen werden die Dokumente aus X neu generiert. Die danach nicht mehr referenzierten Dokumente aus L werden aus dem Zustand „Publiziert“ entfernt, vom Web-Server gelöscht und in das Archiv übernommen, ohne das Inkonsistenzen zu Tage getreten wären.

Bei einer Modifikation der Navigationsstruktur in der Web-Präsenz oder in den Export-

Templates ist das hier beschriebene Prozedere der partiellen Neugenerierung nur bedingt einsetzbar, da in einem solchen Fall eine virtuelle Neugenerierung sämtlicher publizierter Dokumente durchgeführt werden muss, um Änderungen in der Hyperlink-Aktivierung zu ermitteln. Hier bleibt es in der Regel dem *Webmaster* überlassen, die Neugenerierung einzelner Ressorts mitsamt deren Unterressorts anzustoßen, wenn auf eine vollständige Neugenerierung verzichtet werden soll.

5.4.6 Dynamischer Export

Der dynamische Export von JDaphne setzt voraus, dass die notwendige Datenbasis extern verfügbar ist. Dies bedeutet, dass Zugriff auf die Datenbank, die publizierten Inhalte und die Export-Templates bestehen muss. Ferner muss der Web-Server in der Lage sein, das Programm, welches die dynamische Generierung der Dokumente leistet, auszuführen. Bei der aktuellen Implementation von JDaphne findet der Export über ein Servlet statt. Der Web-Server benötigt daher ein entsprechendes Modul für die Bereitstellung der Servlet-Laufzeitumgebung. Die Anfragen werden von ihm an das Servlet weitergeleitet. Dieses wertet die Parameter, z.B. die angeforderte URI, das von dem Benutzer selektierte Template etc. aus und konstruiert das angeforderte Dokument aus Navigation, Layout und Inhalt. Da für die Generierung jedes Dokumentes der Inhalt selbst und das Template aus dem Dateisystem gelesen werden müssen, die Datenbank mit einer Vielzahl von Abfragen okkupiert wird und schließlich der Parse- und Zusammenbau-Vorgang auch einige Zeit in Anspruch nimmt, ist die Antwortzeit dieser dynamischen Variante deutlich größer als bei der statischen Variante. Der konkrete Faktor hängt dabei stark von dem Template und der speziellen Eigenart des Dokumentes ab. Trotz dieser Geschwindigkeitsprobleme kommt diese Variante mehr und mehr zum Einsatz, da ihr Vorteil in der sofortigen Übernahme eines neuen Publizierungsstandes der Dokumentenbasis liegt.

5.4.6.1 Einsatz von Mediatoren

5.4.6.1.1 Aktuelle Übersichten des publizierten Dokumentenbestandes

Für die Erstellung von dynamischen Übersichtsdokumenten bietet JDaphne die Möglichkeit, Anfragen an die interne Datenbasis in Dokumente einzubetten. Bei der Voransicht und dem Export (dynamisch oder statisch) werden diese Anfragen auf SQL-Basis ausgeführt und das Ergebnis in Form von Hyperlinks auf die gefundenen Dokumente dargestellt. Darüber lassen sich neben einfachen Übersichten beispielsweise in der Form „die drei aktuellsten Dokumente aus dem Ressort Pressemitteilungen“ auch komplexe Übersichten in der Form „alle Dokumente, die Herr Meier oder Frau Müller zu dem Thema 'Mobile Commerce' in den letzten zwei Jahren verfasst haben“ erstellen. Neben dem Einsatz von vorgefertigten einfachen Abfragen, die über vorkonfigurierte Templates von JDaphne bereitgestellt werden, können von mit dem Datenmodell vertrauten Autoren auch komplexere Abfragen realisiert werden. Die dynamischen Übersichtsdokumente sind aus Sicht von JDaphne normale HTML-Dokumente. Sie enthalten lediglich, wie dies auch für die Einbindung anderer Datenquellen gehandhabt wird, Absätze mit Abfrage-Parametern in der Form:

```
<dynamische Daten>
# Als Datenquelle die interne Datenbank von JDaphne verwenden
datenquelle=internal
# Die sql-Abfrage, die ausgeführt werden soll
abfrage=select Titel, dokument from Meta where
        ressort='ressortkürzel' and status='publiziert'
        order by erstellungsdatum desc limit 5
# Bei der zeilenweisen Darstellung soll dokument in den Anker-Tag
  des Hyperlinks eingesetzt werden
anker=dokument
```

```
# Die Darstellung soll in Form einer Liste erfolgen.
  Alternativ ist eine Tabelle (table) möglich.
typ=list
</dynamische Daten>
```

Beim Abruf solcher dynamischer Übersichten wird die Abfrage von dem System ausgeführt. Das Ergebnis ersetzt den Parameterblock. Der „fertige“ Inhalt wird den normalen Hyperlink- und der Layout-Umsetzung unterworfen. Für den statischen Export werden die dynamischen Übersichten in einem definierbaren Zyklus neu generiert, um auf diesem Wege quasi-dynamische Übersichten zu erhalten.

5.4.6.1.2 Einbindung anderer Datenquellen

Eine Web-Präsenz lebt durch die Interaktion, die sie den Benutzern bietet. Interaktion auf einer Web-Präsenz wird im Allgemeinen durch Skripte oder Programme erreicht, die als Schnittstellen zu Datenquellen dienen. JDaphne verwaltet die Einbindung solcher vom System aus betrachteten externen Datenquellen auf der Basis von Dokumenten. Entsprechende Dokumente enthalten neben normalen textuellen Elementen zusätzlich datenquellenspezifische Angaben. *Autoren* spielen zur Einbindung einer neuen Datenquelle ein dafür parametrisiertes Dokument als Repräsentant der einzubindenden Datenquelle in das System ein. Dieses Dokument durchläuft den Standard-Workflow für Dokumente, es unterscheidet sich jedoch bezüglich seiner Behandlung auf der Web-Präsenz von anderen Dokumenten. Als Repräsentant einer Datenquelle kann es über zwei verschiedene Mechanismen für die Interaktion mit dem Benutzer bereitgestellt werden.

Der erste Mechanismus setzt den auf die Web-Präsenz publizierten Repräsentanten als Template ein. Dieses stellt den Rahmen für das die Schnittstelle zu der Datenquelle repräsentierende Programm dar. Das Programm liest vor jeder Aktion das Template, wertet die Parametrisierung aus und baut seine Benutzerschnittstelle, die Eingabe-Maske und/oder die Datenausgabe in den von JDaphne publizierten Rahmen hinein. Das so zusammengebaute Dokument wird an den Web-Browser des Benutzers zurückgesandt. Es unterscheidet sich nur dadurch von anderen Dokumenten, dass die visualisierten Daten durch die Interaktion mit dem Benutzer konstruiert und aufbereitet wurden.

Der zweite Mechanismus arbeitet nach dem Prinzip des dynamischen Exports. Vor dem Ausliefern eines Dokumentes an den Web-Browser baut die dynamische Export-Komponente nicht nur das Dokument aus Inhalt, Navigation und Layout zusammen. Zusätzlich wird auch noch der Inhalt selbst geparst. Handelt es sich bei dem Dokument um einen Repräsentanten einer externen Datenquelle, so führt die Export-Komponente über den Mediator-Server die in dem Repräsentanten parametrisierte Aktion aus. Dabei handelt es sich normalerweise um den Aufruf einer der oben erwähnten interaktiven Komponenten, z.B. eines Skriptes. Ist das Ergebnis der Abfrage ein HTML-Dokument, so wird dieses analog zu einem aus dem Dateisystem gelesenen Dokument von der Export-Komponente aufbereitet. Ist das Ergebnis ein anderes Dokumenten-Format, gibt die Export-Komponente dieses ohne weitere Verarbeitung zurück. Für die Abfrage relationaler Datenbanken steht die auch für die dynamischen Übersichtsseiten genutzte Basis-Lösung zur Verfügung. Sie erlaubt es, sofern entsprechende Datenbanktreiber vorhanden sind, die in dem Repräsentanten spezifizierte Abfrage auszuführen und das Ergebnis der Abfrage an den Benutzer zurückzuliefern. Ebenfalls durch den Repräsentanten können die Bedienelemente für die Abfrage spezifiziert werden. Unterstützt wird die Navigation durch die Datensätze, Sortier-Reihenfolgen und im Vorhinein spezifizierte Einschränkungen durch Abfrage-Kriterien.

Kapitel 6

Aufbau und Management einer Web-Präsenz mit JDaphne

Einen wichtigen Aspekt bei der Entwicklung von DAPHNE und JDaphne stellt deren produktiver Einsatz im Unternehmen dar. Durch die konkreten Aufgabenstellungen und die speziell auf die Unternehmenswünsche angepassten Implementationen und Konfigurationen der Systeme sind viele konzeptionelle Entscheidungen beeinflusst worden. Einerseits beschneidet die projektgebundene Arbeit dabei durch die Vorgaben und Randbedingungen der Projektpartner das Spektrum der Realisierungsmöglichkeiten, andererseits liefert gerade der Praxiseinsatz selbstentwickelter Systeme tiefe Einblicke in deren Schwächen und Stärken. Beschreibt das zweite Kapitel noch die Problemstellung prinzipiell und fragt nach einem Werkzeug, das die Verwaltung einer Web-Präsenz ermöglichen kann, so wird in diesem Kapitel exemplarisch betrachtet, wie der Aufbau und Betrieb einer Web-Präsenz, im speziellen wenn ein Redaktionssystem wie JDaphne oder DAPHNE eingesetzt wird, prototypischerweise abläuft. Dazu werden verschiedene, mit der Integration und dem Betrieb eines solchen Systems verbundene, Punkte angesprochen. Um die Vorgänge zu illustrieren, betrachtet dieses Kapitel den Lebenszyklus einer Web-Präsenz wie in Abschnitt 3.2.2 dargestellt. Trotz des exemplarischen Charakters wird dabei versucht, allgemeingültige Mechanismen zu skizzieren, auch wenn insbesondere die Auswirkung auf das bzw. der Einfluss des Redaktionssystems in den Vordergrund gestellt wird.

Aus der Sicht des Redaktionssystems ergeben sich drei eng miteinander verwobene Phasen (Abb. 6.1) im Leben einer Web-Präsenz. Die erste Phase beinhaltet alle die Konzeption und den prototypischen Aufbau der Web-Präsenz betreffenden Schritte entsprechend den Phasen 2 und 3 aus dem Lebenszyklus der Web-Präsenz. Hier werden die Anforderungen an die Web-Präsenz definiert und vorbereitende Arbeiten getätigt. Zuständigkeiten werden geklärt und Vorgaben für die technische Umsetzung erarbeitet.

Schließlich findet in der zweiten Phase (Phase 4 im Lebenszyklus der Web-Präsenz) die Installation und nach entsprechender Anpassung einzelner Komponenten der Einsatz des Redaktionssystems statt. Der produktive Einstieg des Systems benötigt als präparatorische Arbeiten die Prozessintegration im Unternehmen sowie darauf aufbauend die Schulung der Mitarbeiter.

In der dritten Phase wird vorrangig der Betrieb der Web-Präsenz mit kleineren strukturellen Modifikationen, Phase 5 und 6 im Lebenszyklus der Web-Präsenz, mit den dabei anfallenden Arbeiten zusammengefasst. Neben der regelmäßigen Aktualisierung des Dokumentenbestandes fallen insbesondere die statistische Auswertung der Zugriffe auf die Web-Präsenz, ihr Monitoring sowie Backupstrategien in diese Phase. Da sich eine Web-Präsenz aus Aktualitätsgründen immer zu einem gewissen Grad im Umbruch befindet, wird weiterhin

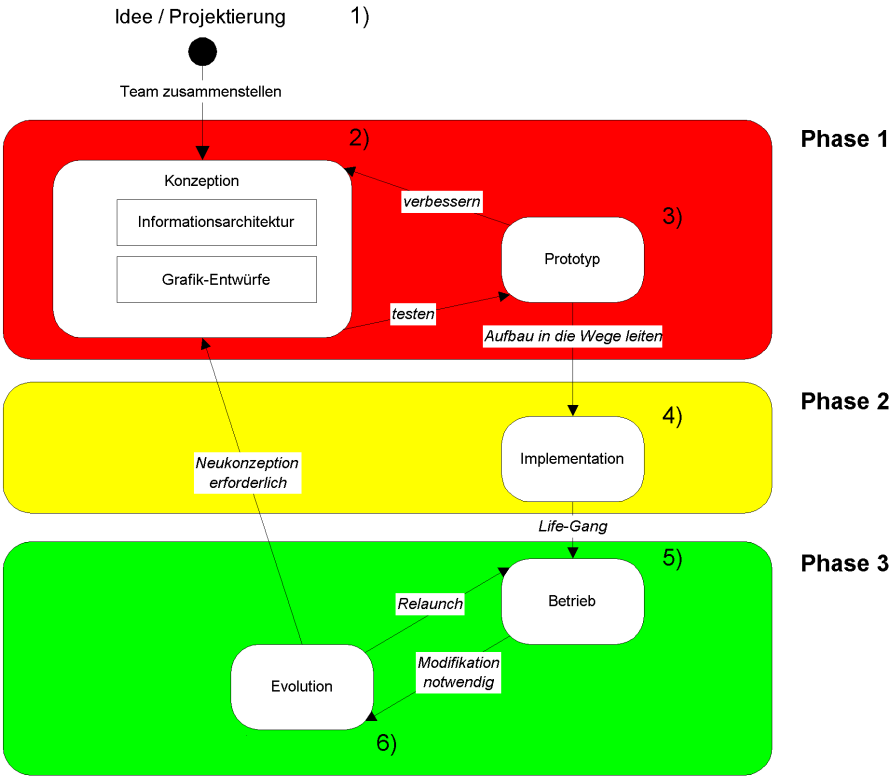

Abbildung 6.1: Die drei Phasen, in die sich unter Berücksichtigung des Redaktionssystems der Lebenszyklus einer Web-Präsenz unterteilen läßt (vgl Abb. 3.10).

an der konzeptionellen Ausrichtung und der Struktur gearbeitet werden. Der Dokumentenbestand wird modifiziert, erweitert und reduziert, je nach Bedarf. Deshalb lässt sich dieser Teil von Phase drei nur sehr schwer von den Aufbau-Arbeiten in Phase zwei trennen; allenfalls ein fließender Übergang kann konstatiert werden. Ein markanter Zeitpunkt, der zur Trennung von Phase zwei und Phase drei verwendet werden kann, d.h. die Implementation der Web-Präsenz abschließt, ist der Life-Gang (Übergang Phase 4 nach Phase 5 im Lebenszyklus der Web-Präsenz): die „fertige“ Web-Präsenz wird dem Publikum zugänglich gemacht. Zu diesem Zeitpunkt ist die Basis der Web-Präsenz geschaffen und Modifikationen sowie Evolutionen finden hauptsächlich in peripheren Bereichen der Web-Präsenz statt.

6.1 Konzeption der Web-Präsenz unter Einbeziehung des Redaktionssystems

Auch wenn seit dem Beginn des Web-Präsenz-Zeitalters schon geraume Zeit, insbesondere in Dimensionen der IT-Branche, vergangen ist und mittlerweile viele Erfahrungen in diesem Gebiet gesammelt werden konnten, muss auch heute noch ein Projekt „Unternehmens-Web-Präsenz“ mit großer Sorgfalt konzeptioniert werden. Sehr anschaulich wird beispielsweise in [Jac96] diskutiert, wie weit in einem Unternehmen der Soll-Zustand in einem solchen Projekt von dem Ist-Zustand abweichen kann und welche Probleme der Aufbau einer Web-Präsenz dann mit sich bringt. Ihr Erstellung stellt für ein Unternehmen aus mehreren Gründen auch heute noch eine große Herausforderung dar. Zum einen ist der finanzielle Aufwand [Sie98] für

das Unternehmen nicht unerheblich. Damit sich die Web-Präsenz im internationalen Umfeld behaupten kann, sind umfassende Investitionen zu tätigen. Diese betreffen nicht nur den initialen Aufbau, sondern insbesondere ihren Betrieb. Für die Konzeption der Web-Präsenz sind die finanziellen Mittel der das Projekt „Web-Präsenz“ limitierende Faktor. Von ihm hängt ab, welche Ressourcen zur Verfügung stehen, dies sowohl aus technischer Sicht als auch hinsichtlich des möglichen Personaleinsatzes. Da sich bislang der Mehrwert einer Web-Präsenz für ein Unternehmen nur im Fall einer transaktionellen Ausrichtung exakt messen lässt, fehlt es oft an verlässlichen Abschätzungen für einen angemessenen Ressourcen-Aufwand. Um so wichtiger wird es, das Projekt mit einem tragfähigen Konzept zu versehen. Unter der Prämisse, dass ein Redaktionssystem zum Einsatz kommt, sind innerhalb der Konzeption neben den inhaltlichen Aspekten auch vielfältige technische Aspekte, z.B. das Design betreffend, relevant, die Einschränkungen nach sich ziehen.

Die professionelle Konzeption einer Web-Präsenz ist, wie in [NL00] beschrieben, eine interdisziplinäre Angelegenheit (Abb. 6.2), die zu großen Teilen noch mit sehr informellen Mitteln stattfindet. Zum Einsatz kommen dabei vor allem die altbewährten Skizzen. Auf Papier oder mit dem im wissenschaftlichen Umfeld entstandenen Tool ”DENIM”[LNHL00] in elektronischer Form werden Site-Maps, Abläufe und Gliederungen entworfen und diskutiert. Der Konzeptionsprozess lässt sich in drei Themenfelder differenzieren: das Informations-Design, das Navigations-Design und das grafische bzw. visuelle Design. Ein neu im Zusammenhang des Web-Präsenz-Designs sich herausbildender Begriff ist die Informationsarchitektur [RM98], die sich aus der Kombination von Informations-Design und Navigations-Design ergibt. Üblicherweise findet das visuelle Design erst nach der Festlegung der Informationsarchitektur statt. Sofern keine expliziten Fachleute im Unternehmen vorhanden sind bzw. ein-

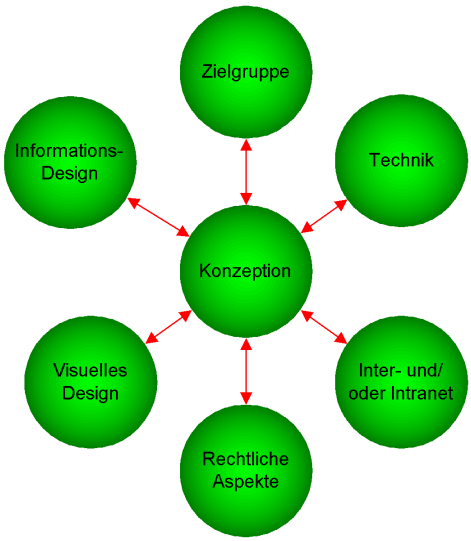

Abbildung 6.2: Aspekte der Konzeption einer Web-Präsenz.

gekauft werden, wirken an der Informationsarchitektur Teams aus Marketing-Mitarbeitern, Technikern und Juristen sowie in großem Maße gerade in inhaltlicher Hinsicht die Fachabteilungen des Unternehmens mit. In welcher Reihenfolge und in welchem Umfang die Mitwirkenden in die Konzeption eingebunden werden, variiert in Abhängigkeit von der geplanten Ausrichtung der Web-Präsenz. Die Erfahrung hat gezeigt, dass für eine tragfähige Konzeption einer Unternehmens-Web-Präsenz die Diskussion und Berücksichtigung der im folgenden diskutierten Aspekte gute Ausgangspunkte sind. Insbesondere wenn die Web-Präsenz mit

einem Redaktionssystem aufgebaut und verwaltet werden soll, muss dies in die Konzeption einfließen. Im Normalfall sind durch den hohen Automatisierungsgrad, der mit dem Redaktionssystem verbunden ist, Einschränkungen auferlegt. Ist die Entscheidung bereits gefallen, mit welchem System die Web-Präsenz verwaltet werden soll, kann die Konzeption die Vorgaben dieses speziellen Systems aufgreifen. Anderenfalls muss die Konzeption die Automatisierungstauglichkeit ohne die Kenntnis des konkreten Systems in allgemeiner Breite berücksichtigen.

6.1.1 Kenntnis der Zielgruppe

Die oberste Vorgabe bei der Konzeption einer Web-Präsenz stellt die Beantwortung der Frage nach der Zielgruppe [Riz00], für die sie entwickelt wird, dar. Nur wenn die angesprochenen Benutzerkreise bekannt sind, kann eine zielgerichtete Konzeption [BB00] von Design, Struktur, Interaktivität und Inhalten erfolgen und für die Benutzer eine entsprechende Nutzbarkeit [Zib00] gewährleistet werden. Eine Basis für die Ermittlung einer Benutzergruppe und deren bevorzugte Hardware-Ausstattung bieten Statistiken über die Nutzung des Internets [CC01]. Aufgabe der Marketing-Abteilung des Unternehmens ist es, aus diesen Benutzerkreisen die für das Unternehmen relevanten Gruppen zu identifizieren und damit die Vorgabe für die Konzeption der Web-Präsenz zu liefern. Von technischer Seite betrachtet ist insbesondere die durchschnittliche Hardware-Ausstattung der Benutzergruppen von großem Interesse. Sie bestimmt maßgeblich die Ausgestaltung der Web-Präsenz bezüglich der Bildschirmauflösung und der Komplexität dynamischer Komponenten.

6.1.2 Ausrichtung der Web-Präsenz – Ziele

Gemeinsam mit der Spezifikation der Zielgruppen erfolgt die Festlegung der Ausrichtung der Web-Präsenz. Hier müssen die Fragen beantwortet werden, welchem Zweck sie dienen soll und welche Intention mit ihrem Aufbau aus der Sicht des Unternehmens verbunden ist. Aus der Beantwortung dieser Fragen folgen die Vorgaben für die mit der Web-Präsenz zu realisierenden Funktionalitäten. Diese wiederum beeinflussen maßgeblich die an das Redaktionssystem und gegebenenfalls weitere für die Bereitstellung der Funktionalitäten notwendige Komponenten zu stellenden Anforderungen. Aus der Positionierung der Web-Präsenz bezüglich der geforderten technischen Funktionalitäten lassen sich die für den Betrieb notwendige Hardware auf der einen Seite und der Implementierungsaufwand auf der anderen Seite ableiten. Die genaue Spezifikation der erforderlichen Funktionalitäten in Form eines Pflichtenheftes ist Voraussetzung für eine termingerechte Umsetzung durch die Entwickler. Ein kurzer Überblick über Ziele, die für ein Unternehmen mit dem Aufbau einer Web-Präsenz verbunden sein können, kann im Rahmen der Einleitung dieser Arbeit nachgelesen werden.

6.1.3 Informations-Design

Sobald die Ausrichtung der Web-Präsenz und ihre anzusprechende Zielgruppe bekannt ist, kann mit der sorgfältigen Konzeption der zu präsentierenden Inhalte begonnen werden. Nur wenn diese überzeugen, kann mit einer dauerhaften Akzeptanz durch das Zielgruppenpublikum gerechnet werden. Weil, wie die Praxis [Lin01] zeigt, eine Restrukturierung immer mit überproportionalem Aufwand verbunden ist, sollte vor der Implementation der Web-Präsenz zumindest eine weitreichende Strategie bezüglich der zu präsentierenden Inhalte vorhanden sein [Nie00]. Für die Akzeptanz auch innerhalb des Unternehmens setzt der Aufbau einer Web-Präsenz voraus, dass innerhalb des Unternehmens ein Konsens darüber herrscht, welche Informationen publiziert werden sollen. Diese Entscheidung wird neben der Geschäftsführung

im wesentlichen durch das Marketing getragen. Wichtig ist in jedem Fall, dass in diesem Zusammenhang bereits während der Konzeption geklärt wird, wer die zu publizierenden Inhalte bereitstellt. Für die intern vorhandenen Inhalte muss die Bereitschaft und Befähigung, die konzeptionierten Inhalte zu erstellen, vorhanden sein, damit die Web-Präsenz selbstständig gepflegt wird. Zusätzlich kann es aus konzeptionellen Überlegungen notwendig werden, nicht im Unternehmen vorhandene bzw. nicht im Besitz des Unternehmens befindliche Daten auf der Web-Präsenz zu veröffentlichen. Für die Bereitstellung und die Aufbereitung solcher Inhalte ist mit externen Quellen, im Normallfall Agenturen, rechtzeitig Kontakt aufzunehmen.

Ob die von einer Web-Präsenz bereitgestellten Informationen von dem Publikum der Zielgruppe abgerufen werden, hängt neben ihrer Qualität und Aktualität in hohem Maße von ihrer Auffindbarkeit [Fle98] ab. Dokumente werden auf einer Web-Präsenz durch zwei Mechanismen gezielt gefunden: entweder der Benutzer setzt die Suchmaschine der Web-Präsenz ein oder er folgt den durch die Navigationselemente vorgegebenen Pfaden. Sind diese einfach gangbar und ist ihre Struktur für das Publikum leicht nachvollziehbar, werden die Inhalte von den Benutzern problemlos gefunden. Grundsätzlich sollten alle Pfade das Publikum so schnell wie möglich ans Ziel führen. Nur wenige Internetnutzer sind bereit, in tiefere Strukturen einer Web-Präsenz einzudringen, solange sie nach allgemeinen Informationen suchen, erst bei sehr fachspezifischen Inhalten wächst diese Bereitschaft. Bezüglich der Tiefe und Breite [LC98] der inhaltlichen Struktur einer Web-Präsenz ist dementsprechend ein geeigneter Kompromiss zu finden. Zu den Inhalten von allgemeinem Interesse sollten kurze Wege führen, die der breiten Masse [Mil56] einfach zugänglich sind. Spezielle Inhalte können tiefer platziert werden und von der Interessensgruppe gezielt zugreifbar sein. Grundsätzlich sollte bei der Navigationsstruktur auf intuitive Nutzbarkeit geachtet werden, denn nur wenn die Navigation der Intuition folgend möglich ist, können die Benutzer die Web-Präsenz geeignet bedienen.

Bei allen konzeptionellen Entscheidungen bezüglich der Inhalte und ihrer Struktur auf der Web-Präsenz ist zu beachten, dass ihre Umsetzbarkeit mit dem zum Einsatz kommenden Redaktionssystem gegeben ist. Vorteilhaft bei den in dieser Arbeit beschriebenen Systemen wirkt sich die durch die spezielle Anpassung der Export-Komponente entstehende hohe Flexibilität aus. Auch bezüglich der Eingabekomponenten, die als beliebige Anwendung integriert werden können, stellen sich die Systeme als sehr offen dar. Die Berücksichtigung der sich aus dem Einsatz solcher Systeme folgenden Restriktionen hilft jedoch, die Einschränkungen von vornherein zu minimieren und damit die Umsetzung zu beschleunigen.

6.1.4 Visuelles Design

Über Geschmack lässt sich bekanntlich streiten. Dies gilt auch für das Design [Lan01] einer Web-Präsenz. Neben den technischen Einschränkungen, die in den folgenden Abschnitten beschrieben werden, ist hier insbesondere die Zielgruppe im Auge zu behalten. Abgesehen davon, dass grafik-lastige Web-Präsenzen durch die Geduld der Benutzer strapazierende Download-Zeiten unangenehm auffallen, werden sie von vielen Benutzern auch als „überladen“ empfunden. Die Anwendung von dynamischen Elementen belebt zwar eine Web-Präsenz, aber auf Grund von Inkompatibilitäten nicht auf jedem Web-Browser in gleicher Weise. Beim Design kommen dementsprechend eine Reihe von Randbedingungen zum Tragen, die dazu führen, dass ein gutes Design sich nicht nur in „gutem“ Aussehen wiederspiegelt. Vielmehr muss eine im Gesamtbild „funktionierende“ Web-Präsenz bei den Design-Aspekten im Vordergrund stehen. Es spricht vieles dafür, eine Web-Design-Agentur mit der grafischen Gestaltung zu beauftragen. Neben der Professionalität der vorgelegten Entwürfe hat dies den Vorteil gegenüber im Unternehmen entworfenen Designs, dass die Entwürfe von einer unabhängigen

Instanz stammen und damit außerhalb jeder internen Kritik stehen. Dabei sind der Agentur neben den technischen Vorgaben des Designs auch solche bezüglich der Gestaltung zu machen. Als sehr wichtig hat es sich erwiesen, dass nach ersten grundlegenden Entwürfen in Form von Grafiken, die späteren, insbesondere der finale Entwurf, bereits im Zielmedium, d.h. als HTML-Dokument mit den eingebetteten grafischen Gestaltungselementen, vorgelegt werden. Auf diese Weise ist gewährleistet, dass die Vorschläge der Design-Agentur sich in dem Zielmedium zumindest aus gestalterischer Sicht umsetzen lassen. Weiterhin kann so die für den Benutzer der Web-Präsenz anfallende Datenmenge direkt abgeschätzt werden und somit in die Entscheidung einfließen.

Neben der Startseite sind von der Agentur auch konzeptionelle Vorgaben für die Gestaltung der hierarchisch tieferliegenden Ebenen anzufordern, die in die Erstellung der Templates einfließen. Angepasst an die inhaltlichen Vorgaben müssen hier die Navigationsstrukturen „grafisch“ umgesetzt werden. Gegebenenfalls muss die Visualisierung der inhaltlichen Aspekte stattfinden. Dies kann durch Modifikation des Hintergrundes genauso geschehen wie durch andere Schriftfarben oder dem Beibehalten inhaltsabhängiger Motive. Um ein einheitliches Bild der Web-Präsenz zu garantieren, sollte streng nach dieser Konzeption vorgegangen werden. Der Einsatz des Redaktionssystems ist an dieser Stelle äusserst hilfreich, da den Autoren das Basis-Layout von ihm vorgegeben wird. Durch den mehrstufigen Workflow kann dafür gesorgt werden, dass nur design-konzeption-konforme Dokumente auf der Web-Präsenz publiziert werden. Zu den technischen Aspekten des Designs gibt es verschiedene Ansichten, aber grundsätzlich ist hier allein die Zielgruppe entscheidend. Wenn diese mit einem bestimmten Design erreicht werden kann, dafür aber alle sonstigen Besucher der Web-Präsenz mehr oder weniger ausgeschlossen sind, dann kann das Design trotzdem als gelungen betrachtet werden. Je breiter jedoch die Zielgruppe gefasst ist, desto mehr müssen sich die Designer den technischen Einschränkungen beugen und Kompromisse in Betracht ziehen. Im folgenden sind einige wichtige, sich aus technischen Aspekten ergebende, Einschränkungen erläutert.

6.1.4.1 Optimale Bildschirmauflösung

Eine häufig bei dem grafischen Design unterschätzte Problematik ist die mit vielen Designvorschlägen einhergehende Einschränkung auf eine feste Bildschirmauflösung. Das Design der Web-Präsenz wird für eine Bildschirmauflösung entwickelt; betrachtet der Benutzer die Web-Präsenz mit einer anderen, so stellt sie sich ihm u.U. anders, gegebenenfalls unbrauchbar dar. Wie stark die angestrebte Bildschirmauflösung in die Design-Konzeption einfließt, hängt stark von deren Grafik-Lastigkeit ab. Grafiken zeichnen sich durch eine feste Dimension, die in der Regel auch angestrebt ist, aus. Textuelle Elemente hingegen werden von den Web-Browsern automatisch umgebrochen und passen sich so an die Bildschirmauflösung an. Je mehr eine Web-Präsenz auf grafische Elemente verzichtet, desto flexibler ist sie bezüglich der beim Benutzer vorhandenen Bildschirmauflösung. Vom Design-Aspekt her kann dies als Herausforderung betrachtet werden, da die Web-Präsenz sich durch die unterschiedlichen Layout-Mechanismen der Web-Browser dann in ihrer Darstellung deutlich stärker „anpasst“ als dies bei einer grafik-basierten Gestaltung der Fall ist. Je grafik-lastiger eine Web-Präsenz, insbesondere ihre Navigation, ist, desto mehr sind ihre Dokumente in ihrer Breite und Höhe fixiert. Für die Erstellung von Entwürfen muss vorgegeben werden, wie breit gegebenenfalls eine horizontale Navigationsleiste sein darf. Angegeben wird diese Breite in Pixel. Von der beim Benutzer verwendeten Bildschirmauflösung hängt es ab, ob die gewählte Breite der Web-Präsenz bildschirmfüllend ist. Hat dieser eine identische horizontale Auflösung gewählt, bekommt er eine bildschirmfüllende Ansicht des Dokumentes dargestellt. Ist die von ihm benutzte Auflösung höher, so bleibt ein Randbereich, da die zur Verfügung stehende Breite nicht ausgeschöpft wird. Ist sie niedriger, so ragt die Ansicht über den Bildschirm hinaus

und zwingt den Benutzer zum vertikalen Verschieben der Ansicht beim Lesen. Wie gut die Bildschirmauflösung einer Web-Präsenz gewählt wurde, hängt somit zu einem sehr großen Teil von den Voraussetzungen auf der Hardware-Seite der Zielgruppe(n) ab.

6.1.4.2 Optimierung auf Browser

Richtig eingesetzt kann die Internet-Technologie und dabei insbesondere das WWW als plattformübergreifende Lösung der Rechner-Kommunikation betrachtet werden. Während dies in den frühen Phasen des WWW auch noch als gegeben angesehen werden konnte, haben sich mittlerweile mehr und mehr plattformabhängige Komponenten eingeschlichen. Neben der oft großen Unterschiede bei den auf den Betriebssystemen verfügbaren Schriftarten, welche per se zu einem anderen Aussehen der Web-Präsenz führen, sind oft auch die aktuellen Web-Browser-Versionen nur auf dem Microsoft-Betriebssystem verfügbar. Die Versionen für die anderen Betriebssysteme folgen meistens mit Verzug, da die Benutzergruppe hier deutlich kleiner ist. Hinzu kommt, dass sich, wie bereits erwähnt, die gängigen Web-Browser stark in den von ihnen unterstützten Funktionalitäten differenzieren. Nicht nur, dass einige Funktionen nur von einigen Web-Browsern unterstützt werden, auch die Art und Weise, wie diese Funktionen unterstützt werden, variiert in der Web-Browser-Welt. Dies konfrontiert den Web-Designer mit einer durch einfache Mittel nicht zu lösende Problematik. Es können immer nur Annäherungen, in der Form, dass möglichst viele Web-Browser-Typen unterstützt werden, erzielt werden. Um dies zu gewährleisten, ist es notwendig, eine Test-Umgebung bestehend aus einem oder mehreren Rechnern, auf denen alle zu testenden Web-Browser aufgespielt sind, zu schaffen. Mit diesen können die zu unterstützenden Funktionalitäten während der Entwicklungsphase getestet werden. Das Design muss so lange modifiziert werden, bis es mit den in den Spezifikationen geforderten Versionen zusammenarbeitet. Sehr empfehlenswert in diesem Kontext ist es, eine auf der angestrebten Zielgruppe basierende Vorgabe zu erarbeiten, die festlegt, welche minimalen Anforderungen von dem Design erfüllt sein müssen. Bei der Optimierung des Designs auf hohe Web-Browser-Kompatibilität fallen besonders unterschiedliche Implementationen der Layout-Vorschriften und innerhalb der Web-Browser-Welt inkompatible Skript-Sprachen-Unterstützung auf.

6.1.4.3 "Redaktionssystem-Tauglichkeit"

Genauso wie die inhaltliche Konzeption muss auch das grafische Design auf seine automatisierte Umsetzbarkeit mit einem Redaktionssystem überprüft und gegebenenfalls daraufhin optimiert werden. Die automatisierte Umsetzung von Design-Konzepten stößt schnell an ihre Grenzen, wenn grafik-lastige Entwürfe durch die Export-Komponente erstellt werden müssen. Prinzipiell lassen sich auf Textgestaltung aufbauende Konzepte leichter automatisiert umsetzen als grafik-orientierte Ansätze. Dies liegt daran, dass die automatische Generierung von grafischen Elementen, z.B. Navigations-Icons, nur schwer den Ansprüchen der Designer gerecht wird. Hier ist beträchtlicher Aufwand vonseiten des Systems notwendig, um auf manuelle Unterstützung verzichten zu können. Wird diese Problematik von vornherein bei den gestalterischen Entwürfen berücksichtigt, lassen sich unter Umständen kombinierte Lösungen finden. Vorschläge, bei denen die relevanten, sich selten ändernden Elemente als Grafiken umgesetzt und die tieferliegenden, sich häufiger ändernden Strukturen über textuelle Visualisierungen zugänglich gemacht werden, sind in diesem Fall tragfähig. Ist von Anfang an klar, welches System eingesetzt werden soll und wie dessen Funktionalitäten aussehen, so können entsprechende Anforderungen für das Pflichtenheft formuliert werden. Bei den mit DAPHNE bzw. JDaphne umgesetzten Designkonzepten handelt es sich um das vollständige Spektrum. Rein grafik-orientierte Konzepte wurden ebenso realisiert wie komplett textbasier-

te; auch eine Kombination aus beiden hat sich als funktionierende Lösung dieser Problematik erwiesen. In jedem Fall erfolgte bei DAPHNE und auch bei JDaphne eine Anpassung der Exportkomponente bezüglich der Navigationsdarstellung an die individuellen Anforderungen des Projektpartners. Wie erwartet wächst der manuell zu leistende Aufwand für die Erstellung geeigneter Grafiken mit dem Grad der Grafikabhängigkeit. Solange die Häufigkeit der Strukturmodifikationen gering ist (und dies sollte bei einer inhaltlich ausgereiften Konzeption der Fall sein), bleibt der mit der Erstellung von Grafiken verbundene Aufwand überschaubar.

6.1.5 Rechtliche Aspekte

Auch wenn es zu Beginn des Internets und insbesondere des WWW so schien, als böte dieses neue Medium einen rechtsfreien Raum, so hat mittlerweile die stark wachsende Benutzergruppe in Kombination mit einer kommerziellen Nutzung dieses Mediums dazu geführt, dass zumindest auf nationaler Ebene versucht wird, die aus den konventionellen Medien bekannte Rechtsprechung auf das Internet zu übertragen. Die Internationalität des WWW sorgt hier für derzeit ungelöste rechtliche Probleme, da verschiedene nationale Rechtsordnungen miteinander kollidieren. Derartige Kollisionen können im gesamten Spektrum des Marken-, Patent- und Urheberrechtschutzes, z.B. in Bezug auf Copy Rights, Patente, Warenzeichen und Marken resultieren.

Bei der Konzeption einer Web-Präsenz muss auf jeden Fall die Einbettung in den nationalen rechtlichen Raum gewährleistet sein, um sich zumindest auf dieser Ebene abzusichern. Dies bezieht sich beispielsweise auf den Schutz von allgemeinen Persönlichkeitsrechten und anderen potentiell über das Internet zu realisierenden Straftatbeständen – z.B. Verleumdung, Diffamierung und Beleidigung auf der einen Seite, Kinderpornographie, Rechtsradikalismus und Anstiftung zu Straftaten auf der anderen Seite.

Im Bereich des Urheberrechts ist beispielsweise zu beachten, dass viele Grafiken nicht frei nutzbar sind und daher für ihren kommerziellen Einsatz durch Lizenzen abgedeckt werden müssen. Gerade in größeren Unternehmen ist es wichtig, die Inhalte, die auf der Unternehmens-Web-Präsenz veröffentlicht werden sollen, im Vorfeld durch Juristen gegenprüfen zu lassen.

6.1.6 Intranet & Internet

Ein weiterer wichtiger Punkt bei der Konzeption der Web-Präsenz ist die Klärung der Frage, inwieweit für das Intranet zusammengestellte und aufbereitete Inhalte für die Internet-Präsenz übernommen werden können und sollen sowie vice versa. Verschiedene Konzepte sind hier denkbar [Hor99]. Die vollständige Separierung von Internet- und Intranet-Web-Präsenz ist das bislang am häufigsten realisierte Konzept. Dies hat seinen Grund zum einen in der einfachen Umsetzbarkeit dieses Konzeptes, da viele Abstimmungsarbeiten entfallen und Arbeitsabläufe für Dokumente vereinfacht werden. Die Arbeiten an den Dokumentenbeständen erfolgen durch disjunkte Mitarbeitergruppen. Synergie-Effekte, aber auch gegenseitige Behinderungen werden so ausgeschlossen. Zum anderen liegt die Separierung aus historischer Sicht betrachtet darin begründet, dass viele Web-Präsenzen im Internet nach den Vorgaben des Unternehmens durch eine Web-Design-Agentur erstellt und verwaltet wurden und sich so dem Zugriff der Mitarbeiter entzogen. Im Gegensatz dazu hat sich häufig die Intranet-Präsenz vor allem durch die Initiative einzelner von der WWW-Technik begeisterter Mitarbeiter vergleichsweise unkoordiniert auf einem „freien“ Rechner entwickelt und wurde erst in späterem Stadium mit offiziellem Charakter versehen und reglementiert. Auf diese Weise entstanden in den Unternehmen zumindest zwei vollkommen unabhängige Präsenzen. Für die Mitarbeiter stellt dies aus Informationszugangssicht keinen direkten Nachteil dar, da sie sowohl auf

die Internet- als auch die Intranet-Präsenz Zugriff haben. Über Hyperlinks können relevante Ressourcen der Web-Präsenz im Intranet referenziert und somit eine Quasi-Integration nachempfunden werden.

Aber auch bei der Neukonzeption, selbst bei Einsatz eines Redaktionssystems, macht es durchaus Sinn, eine strikte Trennung von Internet- und Intranet-Präsenz zu fordern. Die Art der Informationen und ihre Aufbereitung kann sich derart gravierend unterscheiden, dass eventuelle Synergie-Effekte so klein sind, dass eine Mehrfachnutzung nicht rentabel ist. Aufgabe der Konzeptionsphase ist es zu analysieren, ob Synergien vorhanden sind, wenn ja welche es gibt und wie sie sich sinnvoll nutzen lassen. Sind keine Synergien zu erwarten, so kann auf die initiale Integration des Datenbestandes von interner und externer Web-Präsenz verzichtet werden. In diesem Fall sollte aber diese Erkenntnis hinterfragt und gegebenenfalls ein Integrationsprozess angestoßen werden.

Sind Synergien vorhanden, so sollten diese identifiziert und genutzt werden. Beispielsweise ist es sinnvoll, das Reservoir der im Intranet verfügbaren Dokumente zu nutzen, um diese, gegebenenfalls in überarbeiteter Form, auch im Internet zu veröffentlichen. Dabei ist zwingend zu beachten, dass keine internen Informationen nach außen gelangen können. Durch die Einschränkung auf einzelne, für Kundenkontakt zuständige Abteilungen, z.B. die Marketing-Abteilung, kann eine entsprechende Filterfunktion für den Datenfluss aus dem Intranet auf die externe Web-Präsenz installiert werden. Bestimmte Dokumente, beispielsweise Teile von Produktspezifikationen, bieten sich für eine Mehrfachnutzung an.

Eine starke und zugleich einfache Integration von Internet- und Intranet-Präsenz kann erreicht werden, wenn beide in Teilen ihrer Struktur analog angelegt und aus dem gleichen Dokument-Reservoir bedient werden, d.h. einzelne Dokumente lediglich den „Stempel“ extern bekommen und deshalb im Internet veröffentlicht werden. Dies ist z.B. sinnvoll, wenn die einzelnen Abteilungen in Form von Profit-Centern organisiert sind und ihre Produkte nicht nur intern, sondern auch extern anbieten müssen.

Aus der Art und Weise einer Integration von Internet- und Intranet-Präsenz ergeben sich für die technische Basis der Web-Präsenz und insbesondere für die Organisation der Daten mit dem Redaktionssystem Konsequenzen. Werden die Dokumentenbestände von Internet- und Intranet-Präsenz als disjunkt betrachtet und unterscheidet sich zudem die Struktur der Präsenzen erheblich, spricht einiges dafür, die inhaltliche Trennung auch über das zugrundeliegende Redaktionssystem wiederzugeben. DAPHNE beispielsweise wurde durch umfassende Anpassungen modifiziert mit zwei in der Verwaltung getrennten Dokumentenbeständen für Internet und Intranet betrieben. Dabei wurde die Trennung von internem und externem Datenbestand durch einen separaten Zugang zu beiden Funktionalitäten des Systems verdeutlicht. Analoge Benutzerschnittstellen, die jeweils den aktuellen Bereich veranschaulichen, in dem der Benutzer aktiv ist, bieten an die speziellen Anforderungen jedes Bereichs angepasste Funktionalitäten.

Anders stellt sich die Ausprägung des Redaktionssystems dar, wenn der Dokumentenbestand für Internet und Intranet integriert verwaltet wird. Der Verwaltungsaufwand, der dem System abverlangt wird, steigt um ein Vielfaches, da es die Zuordnung jedes Dokumentes zu einem oder mehreren Publikationszielen mitsamt den damit verbundenen Implikationen verwalten muss. Insbesondere das Hyperlink-Management wird deutlich komplizierter, da die Referenzen zwischen Dokumenten von den Publikationszielen abhängen. Das gemeinsame Betreiben verschiedener Web-Präsenzen lässt sich beispielsweise durch die Hyperlink-(De-)Aktivierungsmechanismen von JDaphne realisieren. Im Intranet soll jedes Dokument jedes andere referenzieren dürfen. Für die externe Web-Präsenz muss gewährleistet werden, dass keine Hyperlinks in das interne Netz zeigen, da diese für die Benutzer der externen Web-Präsenz nicht zugänglich sind und als „Broken-Link“ erfahren werden.

Bereits diese knappe Diskussion der Internet-/Intranet-Problematik zeigt, dass mit der Erstellung und Umsetzung eines Integrationskonzeptes ein im Vergleich zu der nur im Internet bzw. nur im Intranet betriebenen Web-Präsenz eine deutliche Aufwandssteigerung verbunden ist. Nur wenn die Umsetzung einer solchen integrativen Konzeption erfolgreich verläuft, steht zu erwarten, dass sich mit dem Betrieb auf Dauer eine Reduzierung des Gesamtaufwandes realisieren lässt und sich somit der erhöhte Investitions- und Konzeptionsaufwand amortisiert.

6.1.7 Technische Aspekte (Server)

Die Betrachtung der technischen Auswirkungen von inhaltlicher, funktionaler und gestalterischer Konzeption ist ein weiterer unabdingbarer Bestandteil der Konzeption einer Web-Präsenz. Abgesehen von der für die Installation des Redaktionssystems benötigten Infrastruktur muss der Betrieb der Web-Präsenz vorbereitet werden. In Betracht zu siehen sind hierfür zwei verschiedene Konzepte: entweder betreibt das Unternehmen den Web-Server selbst, erwirbt nur die Internet-Anbindung vom Internet-Service-Provider und kann so bestimmen, welche Hard- und Software eingesetzt wird oder das Unternehmen überträgt das Hosting der Web-Präsenz vollständig an den Internet-Service-Provider. In letzterem Fall wird automatisch auf dessen Hard- und Software-Konfiguration zurückgegriffen.

Aus technischer Sicht kann heutzutage (2001) ohne weiteres ein handelsüblicher PC mit Internet-Zugang als Web-Server eingesetzt werden. Mit einer solchen Maschine lassen sich die in einem mittelständischen Unternehmen anfallenden Anfragen qunatitativ problemlos bewältigen. Auf entsprechend für den Web-Server-Betrieb optimierte Maschinen muss aber in jedem Fall ausgewichen werden, wenn die Zahl der anfallenden Anfragen an den Server zu groß wird und die Anfragen selbst noch durch server-seitig initialisierte Aktivitäten den Rechner belasten. Deshalb stellt sich bereits in der Konzeptionsphase die Frage, mit welchem Anfrageaufkommen für die Web-Präsenz zu rechnen ist. Dieses hängt neben transaktionellen Elementen insbesondere auch sehr stark von dem geplanten Design der Web-Präsenz ab. Herrschen multimediale Elemente vor, so werden beim Laden eines Dokumentes noch eine Vielzahl von weiteren Anfragen für diese integrierten Elemente generiert. Somit multipliziert sich die Anfrage eines Dokumentes durch einen Web-Browser mit der Zahl der eingebetteten Elemente. Sind sichere Verbindungen von diesem Web-Server zu den Benutzern der Web-Präsenz geplant, so vervielfacht sich die benötigte Rechenleistung des Servers erheblich. Eine gute Konzeption der technischen Basis der Web-Präsenz sieht auf Hardware-Seite eine angemessene Skalierbarkeit vor.

Weitere Überlegungen bei der Konzeption der Web-Präsenz bezüglich der einzusetzenden Hardware müssen auch Ausfallkonzepte bereitstellen. Wird eine Web-Präsenz für Transaktionen eingesetzt, kann ein Ausfall des Web-Servers zu empfindlichen Einbußen führen. Abgesehen davon wirkt sich ein Server-Ausfall auch negativ auf das Image des Web-Präsenz-Betreibers aus.

Für die Software auf einem Web-Server gibt es im Normalfall zwei Hauptkriterien – die Performance/Skalierbarkeit und die Sicherheit. Um eine große Anzahl von gleichzeitigen Anfragen bewältigen zu können, muss die Software sehr effizient arbeiten. Bei Bedarf sollte das System über die Hardware skalierbar sein, ohne dass die Software umgestellt werden muss. Betroffen sind somit sowohl das Betriebssystem als auch die eigentliche Web-Server-Software. Bei der Konzeption der Web-Präsenz sollte zum einen darüber befunden werden, ob mit kommerzieller oder „freier“ Software gearbeitet werden soll; beide Wege haben ihre Vorteile. Wenn es jedoch um Haftungsansprüche geht, bleibt nur die kommerzielle Lösung. Zum anderen hängt von dem Grad der Dynamik, den eine Web-Präsenz bieten soll, ab, welche Art von Web-Server eingesetzt werden kann. Gegebenenfalls müssen zusätzliche Module für den

gewünschten Einsatzzweck verfügbar sein und mit in das Software-Konzept aufgenommen werden.

6.2 Realisierung der Web-Präsenz mit dem Redaktionssystem

Das Ergebnis der Konzeptionsphase ist eine klare Vorstellung der zu realisierenden Web-Präsenz. Geklärt und gegebenenfalls in einem Pflichtenheft abgelegt sind Prozesse, Strukturen und Inhalte, die mit dem Redaktionssystem implementiert werden müssen.

Das Vorgehen zur Installation und Konfiguration des Redaktionssystems wird in einem Projektplan niedergelegt. Dieser definiert den Zeithorizont für die Realisierungsphase und ordnet den Beteiligten spezifische Aufgaben zu. Für ein Unternehmen ist es unproduktiv, viele verschiedene IV-Komponenten gleichzeitig zu betreiben. Aus diesem Grund gibt es strategische Entscheidungen für bestimmte Produkte. Nach der Konzeptionsphase besteht für die Implementationsphase Klarheit darüber, wie das Redaktionssystem in die IV-Landschaft des Unternehmens integriert werden soll, die Hardware- und Software-Plattform ist festgelegt. Es ist spezifiziert, welche Komponenten aus der IV-Landschaft genutzt werden können bzw. müssen und welche vollständig zu integrieren sind. Betroffene Komponenten sind z.B. der Web-Server, die strategische Datenbank-Lösung und auf Client-Seite der Web-Browser. Bei dem Einsatz der Unternehmens-Benutzerverwaltung kommt insbesondere die Anpassung an die von dieser genutzten Strukturen hinzu. Für das Redaktionssystem bedeuten diese Vorgaben trotz aller Portabilität und Plattformunabhängigkeit die Notwendigkeit von Anpassungen. Oft zeigt sich bei dem Integrationsprozess, dass Schnittstellen nicht flexibel genug ausgelegt sind, um durch einfache Konfiguration die ausgetauschte Komponente einbringen zu können. In solchen Fällen sind Modifikationen am Programm-Code die Folge. Für das Redaktionssystem bedeutet eine weitere Installation folglich immer einen Fortschritt in der Konfigurierbarkeit der Komponenten und damit ein weiteres erschlossenes Einsatzfeld

Dass nicht nur technologische, sondern in großem Maße auch organisatorische Probleme zu lösen sind, legen, aus dem praktischen Umfeld des Dokumentenmanagements im Aufbau von Web-Informations-Systemen (WIS) kommend, Balasubramanian und Bashian in „Document management and web-technologies: Alice marries the mad hatter.“ [BB98] dar. Um den Mehrwert eines solchen Systems im Unternehmen nutzen zu können, muss es von organisatorischer Seite in die bestehenden Prozesse integriert werden.

6.2.1 Exemplarische Installation und Integration in die IV-Landschaft

Auf die Anpassungsarbeiten folgt die Installation des Systems in dem Unternehmensumfeld. Dieser Schritt wird in konkreter Form exemplarisch für die Installation von JDaphne beim Institut für Telematik betrachtet.

6.2.1.1 Server

Das Produktiv-System JDaphne des Instituts für Telematik arbeitet server-seitig im internen Netz (vgl. Abb. 5.8) auf einer Solaris 2.6 Plattform, einer SUN Ultra 10 mit 196 MB Hauptspeicher und einem Prozessortakt von 300 MHz. Für den Betrieb des Web-Servers und der Datenbank kommt ein PC unter Linux mit 128 MB und einem Prozessortakt von 266 MHz zum Einsatz. Sowohl auf dem Solaris- als auch auf dem Linux-System steht eine aktuelle Version von *SUN*'s Java-Software-Development-Kit (JDK1.3) als Laufzeitumgebung für JDaphne's Server-Komponenten zur Verfügung. Weiterhin installiert wurden ein *MySQL*-Datenbank-Server sowie ein *Apache*-Web-Server mit dem *Jacarta-Tomcat* als Servlet-Laufzeitumgebung.

Die Installation von JDaphne findet auf einem mit diesen Komponenten ausgestatteten System statt. Die Klassen von JDaphne werden in einem eigenständigen Verzeichnis installiert. Dort liegen in entsprechenden Verzeichnissen die Java-Klassen, die Konfigurationsdateien und Start-Skripte sowie die Templates. Der Datenpool (Dokumenten-Repository) kann in einem beliebigen Verzeichnis abgelegt werden, da er in das Backup-Konzept eingebunden werden muss. Für die Benutzerschnittstelle wird in der Dokumenten-Root des Web-Servers ein Verzeichnis angelegt, das HTML-Dokumentation zu JDaphne, die Einstiegsseite von JDaphne mit dem Applet-Tag, das signierte Archiv mit den Applet-Klassen sowie eine Konfigurationsdatei für das Applet enthält. Ergänzt wird die Benutzerschnittstelle durch die Konfiguration der Servlets, die die Voransicht bzw. die HTML-Variante der Benutzerschnittstelle generieren. Die Konfiguration der Server-Komponenten erfolgt über Konfigurations-Dateien. Dort werden die Namen der Server-Komponenten spezifiziert und die Pfade im Dateisystem eingetragen. Das Datenmodell von JDaphne wird über ein Skript in der Datenbank generiert, nachdem die entsprechenden Benutzer angelegt und mit Berechtigungen versehen wurden.

6.2.1.2 Client

Auf der Client-Seite stellen vorrangig Rechner mit dem Windows-Betriebssystem, von der Hardware in der Pentium II bzw. III Klasse mit 266-800 MHz und 128 MB Hauptspeicher einzuordnen, die Einsatzplattformen von JDaphne dar. Als Client-Software kommen Web-Browser vom Typ *Netscape Navigator* und *Microsoft Internet-Explorer* zum Einsatz. Alternative Plattformen (Linux, Solaris) sind getestet, werden aber derzeit nicht produktiv eingesetzt. Die auf der Client-Seite notwendige Installation beinhaltet als Basisvoraussetzung, sofern die Applet-Variante der Benutzerschnittstelle zum Einsatz kommen soll, das Java-Plugin von *SUN*. Die für die Bearbeitung der Dokumente eingesetzten Produkte hängen von den Vorlieben der Benutzer ab. Neben hochwertigen HTML-Editoren kommen sowohl einfache Text-Editoren als auch Produkte aus der Office-Familie zum Einsatz. Für die Bearbeitung von Grafiken und Bildern stellen der *Photoshop* mit dem *Paintshop Pro* die am häufigsten eingesetzten Produkte dar. Die Konfiguration der von JDaphne für die Bearbeitung von Dokumenten bestimmter Formate einzusetzenden Anwendungen erfolgt durch die Benutzer selbst.

Bezüglich der Performance des Gesamtsystems zeigt der Produktiv-Betrieb, dass insbesondere die als interner Server eingesetzte *SUN*-Hardware als minimale Anforderung verstanden werden kann. Arbeiten mit dem System ist mehreren Benutzern gleichzeitig ohne Einschränkungen möglich, wobei die Responsezeiten bei aufwendigeren Aktionen im akzeptablen Rahmen bleiben. Auf der Client-Seite macht sich der ressourcenverschwendende Gebrauch der Java-Swing-Klassen für die Visualisierung der Übersichten und Aktionen deutlich bemerkbar. Zusammen mit dem gleichzeitig geöffneten Web-Browser und gegebenenfalls von JDaphne gestarteten Editor-Anwendungen kann mit allen oben beschriebenen Arbeitsplatz-Rechnern gearbeitet werden. Auf den kleineren und langsameren Systemen ist jedoch die untere Grenze hinsichtlich des Performance-Bedarfs auf Client-Seite erreicht. Für noch leistungsärmere Systeme bleibt dann der Einsatz der HTML-Benutzerschnittstelle als Alternative.

6.2.2 Einrichten des Systems - Rechte und Rollen

Zum Einrichten des Systems gehört das Anlegen von Benutzern und deren Zusammenfassung zu Gruppen. Weiterhin werden Rollen definiert und konfiguriert. Abschließend erfolgt die inhaltliche Zuordnung von Gruppen zu Ressorts.

Für diesen Integrationsprozess werden im Normallfall die Strukturen des Unternehmens genutzt. In vielen Fällen erfolgt eine Transformation von im Unternehmen bereits etablierter

Hierarchien auf die Vergabe von Rechten und Rollen im Online-Redaktionssystem. Verfügt das Unternehmen über eine PKI, so findet im Rahmen des Integrationsprozesses die Anbindung des Redaktionssystems statt. Die Abbildung der Benutzer innerhalb des Redaktionssystems auf die bereits im Verzeichnisdienst abgelegten Benutzerdaten muss hergestellt werden. Alternativ zu der Abbildung auf interne Mechanismen ist auch die Erweiterung der eingesetzten Zertifikate um redaktionssystemspezifische Attribute möglich. Diese können aus den Zertifikaten abgefragt und dann ohne eine Abbildungsvorschrift direkt mit dem System genutzt werden. Im Allgemeinen stellt der Umweg über den Verzeichnisdienst den geläufigeren Weg dar, da auf diese Weise die verschiedenen Attribute flexibel für unterschiedliche Anwendungen parallel vorgehalten werden können. Konkret kann wie im Institut für Telematik die inhaltliche Zuordnung nach den Fachgebieten der Mitarbeiter erfolgen. Sofern in der Konzeption nicht anders vorgesehen, muss jedoch beachtet werden, dass sich die internen Strukturen nicht in der Struktur der Web-Präsenz niederschlagen. Denn die Sicht, die sich von außen auf die Web-Präsenz ergeben soll, ist auf den Benutzer ausgerichtet (Zielgruppe) und nicht auf die Betreiber der Web-Präsenz, d.h. die interne Struktur des Unternehmens. Deshalb sind bei dem Integrationsvorgang auch geeignete Abbildungen von inhaltlichen Zuständigkeiten in die Struktur der Web-Präsenz zu finden.

6.2.3 Initialer Aufbau der Web-Präsenz

Der initiale Aufbau der Web-Präsenz beginnt mit der Erstellung der in der Konzeption festgelegten Verzeichnisse (Ressorts). Diese werden mitsamt ihren Meta-Informationen in der Datenbank angelegt. Dieser Prozess kann entweder direkt über eine Datenbank-Benutzerschnittstelle, über die Benutzerschnittstelle von JDaphne (Abb. 6.3) oder im Fall von DAPHNE mit der Ressortmanager-Applikation (siehe S. 79) geschehen. Die so entstehende Verzeichnisstruktur dient als Gerüst für den Aufbau der Web-Präsenz. Nachdem die Berechtigungen für die Verzeichnisse an die Gruppen vergeben sind, erfolgt das Bestücken der Web-Präsenz mit Dokumenten. Dazu verbinden sich die Benutzer über den Web-Browser mit der Benutzerschnittstelle des Systems. In Abhängigkeit von dem bereits verfügbaren Dokumentenbestand können jetzt bereits bestehende Dokumente in die entsprechenden Verzeichnisse importiert oder neue Dokumente in den Verzeichnissen erstellt werden. Bei dem Import von Dokumenten muss darauf geachtet werden, dass zusammengehörige Dokumente, d.h. HTML-Dokumente und die eingebundenen Grafiken, möglichst gemeinsam importiert werden. Alle fertiggestellten und korrekt importierten Dokumente werden im folgenden Schritt von den Autoren zum Abzeichnen weitergeleitet. Über die Voransicht können die Freigabeberechtigten die fertigen Dokumente überprüfen und gegebenenfalls an den Webmaster weiterleiten. Dokumente, die ohne Überarbeitung nicht für die Publikation geeignet sind, werden den Autoren erneut vorgelegt. Über die ständige Rückkopplung wird der neue Dokumentenbestand in dem System langsam an die erste zu veröffentlichende Version herangeführt. Der Abgleich des Dokumentenbestandes mit den Vorgaben des Konzepts liefert ein Bild über den Stand der Umsetzung.

6.2.4 Mitarbeiterschulung

Der Einsatz von Internet-Technik und damit insbesondere der Umgang mit Web-Browsern wird in den Unternehmen immer stärker forciert. Durch diese einheitliche Schnittstelle zu den Informationssystemen und anderen Anwendungen besteht oft schon ein Gefühl für die mit dem Medium WWW verbundenen Vor- und Nachteile. Durch gezielte Schulungen verbessert sich das Verständnis der webbasierten Anwendungen noch stark. Gerade für ein System wie JDaphne/DAPHNE ist es hilfreich, wenn die Mitarbeiter mit den dahinter liegenden Konzep-

ten sowie den Stärken und Schwächen der eingesetzten Technologie vertraut sind. Schulungen für JDaphne/DAPHNE teilen sich deshalb im Normalfall in zwei verschiedene Schienen auf: die erste betrifft die im Internet-Umfeld relevanten Techniken, Konzepte und Datenformate, die zweite beschäftigt sich mit der korrekten Anwendung der Systeme.

6.2.4.1 Internet-Umfeld

Ohne Kenntnis der Internet-Technologie und der im Umfeld des WWW benötigten Terminologie ist es den Mitarbeitern sehr schwer möglich, geeignete Inhalte für die Web-Präsenz zu liefern. In der ersten Schiene der Schulung müssen die Mitarbeiter deshalb mit den wichtigsten Konzepten der Internet-Technologie, insbesondere dem WWW, vertraut gemacht werden. Gerade weil es durch die WWW-Technik vor dem Benutzer versteckt wird, ist das Client-Server-Prinzip ein grundlegender Inhalt einer WWW-Schulung. Den Benutzern muss trotz der Einfachheit in der Benutzung klar sein, wann welche Aktionen auf dem lokalen Rechner, ihrem Arbeitsplatz, stattfinden und wann die Aktionen von einem anderen, gegebenenfalls sehr weit entfernten, Rechner ausgeführt werden. Damit entsteht zum einen ein Gefühl dafür, warum es zu Wartezeiten kommt. Zum anderen wird klar, dass es bei Webapplikationen, deren Logik auf der Server-Seite implementiert ist, immer notwendig ist, die Antwort des Servers abzuwarten, bevor eine andere Aktion durch einen erneuten „Klick“ auf einen anderen Menüpunkt gestartet werden kann. Eine Basiskenntnis ist damit auch das Verständnis des Zusammenspiels von Web-Server und Web-Browser. Ohne tiefer auf die technischen Details einzugehen, müssen die Mitarbeiter sich darüber im klaren sein, welche Prozesse in Gang gesetzt werden, wenn der Web-Browser Dokumente vom Web-Server anfordert. Im Rahmen dieser Erklärungen sollte auch besprochen werden, welche Protokolle im Internet zum Einsatz kommen und wie sie sich voneinander unterscheiden. Zu den erforderlichen Kenntnissen für einen erfolgreichen Einsatz von JDaphne/DAPHNE gehören weiterhin die im Internet relevanten Dokumentenformate. Die meisten Mitarbeiter sind mit den gängigen Office-Produkten vertraut. Obwohl die entsprechenden Dokumentenformate mittlerweile ohne Probleme vom Web-Server verwaltet und vom Web-Browser angezeigt werden können, muss den Mitarbeitern ein Gefühl für die im Internet sinnvollerweise einzusetzenden Dokumentenformate gegeben werden. Die Begründung für diese Haupt-Dokumentenformate, wie HTML/XML, JPG, GIF und zunehmend auch PDF für Printversionen, liegt in der erzielbaren Plattformunabhängigkeit und den, bei korrektem Einsatz, kleinen Datentransfervolumen. Selbstverständlich soll kein Mitarbeiter dazu gebracht werden, seine HTML-Dokumente im Source-Code zu editieren. Für den zweckgerichteten Einsatz ist bei dem JDaphne/DAPHNE zugrundeliegenden Konzept, welches nicht auf Eingabemasken, sondern vollständigen vom Benutzer zu erstellenden Dateien beruht, die Kenntnis der Grundelemente dieser Sprache notwendig. Nur wenn die Hauptstrukturelemente bekannt sind, kann erwartet werden, dass die (automatische) Konvertierung eines Dokumentes vom Office-Format in eines der genannten Formate erfolgreich verläuft und der anschließende Import in das System nicht fehlschlägt. Voraussetzung für eine funktionierende Konvertierung ist in jedem Fall ein korrekter Umgang mit den Formatvorlagen der Office-Programme. Ein häufig auftretender Fehler in diesem Zusammenhang ist die Verwendung von Schriftformatierungen für Überschriften. Statt das entsprechende Formatierungselement einzusetzen, werden die Überschriften von den Benutzern manuell formatiert. Beim Export bzw. bei der Konvertierung erkennt das Programm in diesem Fall nicht die strukturelle Formatierung und übernimmt nur die visuelle Formatierung. Damit wird ein solches Dokument schnell für das Internet und insbesondere aus Sicht des Content-Managements, das im Fall von JDaphne/DAPHNE wenigstens auf teilstrukturierte Dokumente angewiesen ist, unbrauchbar.

Eine der Haupteigenschaften des WWW ist seine Verknüpfungsstruktur. Dokumente werden untereinander durch Hyperlinks verknüpft. Diese müssen bei aller Unterstützung durch

JDaphne/DAPHNE über die Navigationsleisten letztendlich von den Mitarbeitern gesetzt werden. Deshalb ist es Teil einer Schulung, die Mitarbeiter über den Aufbau und den Einsatz von Hyperlinks zu informieren. Der Aufbau einer URI muss im Grundkonzept verstanden sein, damit die Mitarbeiter eigenständig ein Netz von Verknüpfungen aufbauen können. Insbesondere der Unterschied zwischen relativen und absoluten Links ist wichtig. Erklärt werden muss zudem, wie Hyperlinks innerhalb von JDaphne/DAPHNE mit der nummernbasierten Referenzierung gesetzt werden. Das Einhalten von Konventionen ist insbesondere wenn verschiedene Mitarbeiter mit den Dokumenten arbeiten sollen, beim Einsatz eines solchen Systems von besonderer Bedeutung.

Auch die Einbindung von Grafiken oder anderen multimedialen Elementen muss im Rahmen der Schulung besprochen werden, damit Probleme in diesem Bereich vermieden werden können. Die Import-Mechanismen mit der automatischen Umsetzung von Verknüpfungen eingebundener Grafiken funktioniert am sichersten, wenn diese sich im gleichen Verzeichnis wie das referenzierende Dokument befinden. Zu den notwendigen Kenntnissen im Internet-Umfeld gehört für die Mitarbeiter, die als Autoren für die Web-Präsenz arbeiten sollen, weiterhin auch ein Verständnis für die Zielgruppe, für die die Inhalte erstellt werden sollen. Je mehr über den Hintergrund der Nutzer einer Web-Präsenz bekannt ist, desto genauer können die Inhalte auf deren Interessen ausgerichtet werden. Dies betrifft nach K. Zibel [Zib00] neben den bereits hier diskutierten formalen Aspekten, beispielsweise sinnvollen Hyperlinks, die angemessen beschriftet sind, auch die Lesbarkeit der Dokumente. Im Rahmen der Schulung muss bei den Mitarbeitern ein Verständnis geweckt werden, für wen sie ihre Dokumente erstellen. Das steigert zugleich auch die Motivation für die inhaltliche Arbeit an der Web-Präsenz.

6.2.4.2 Redaktionssystem

Wenn die grundlegenden Techniken beherrscht werden, erfolgt die Schulung im Umgang mit dem Redaktionssystem. Üblicherweise schließt sich an eine theoretische Erläuterung des Systems direkt dessen praktische Anwendung an. Angepasst an die Rollen, die die Mitarbeiter im System übernehmen sollen, werden die Schulungseinheiten aufgebaut. Die einzelnen Arbeitsschritte werden erläutert und ausprobiert. Optimalerweise wird im Rahmen der Schulung bereits ein Grundstock an Dokumenten in das System eingespielt. Dies hat den Vorteil, dass den Mitarbeitern die Möglichkeit zu Rückfragen gegeben und gleichzeitig schon produktiv gearbeitet wird. Gerade in der Anfangsphase ist eine gute Betreuung wichtig, um die Akzeptanz des Systems zu steigern.

Aus Sicht der Entwickler muss die Schulung am System dazu genutzt werden, bis dato unbekannte Problempunkte zu identifizieren. Dies fällt besonders einfach, wenn der Umgang der zu schulenden Mitarbeiter mit dem System überwacht wird. So lässt sich unter Umständen durch kleine Modifikationen die Intuition beim Umgang mit dem System deutlich steigern. Da ein System wie JDaphne oder DAPHNE aus seiner Konzeption heraus von den Unternehmen in Eigenverwaltung betrieben wird, ist auch eine tiefergehende Schulung für Administratoren und Anwendungsentwickler notwendig da diese als Multiplikatoren auftreten. Sie werden über die Funktionsweise der einzelnen Komponenten aufgeklärt und in die Lage versetzt, auch am Programm-Code Anpassungen durchzuführen. Üblicherweise werden so spezielle Zusatzfunktionalitäten von der unternehmenseigenen Anwendungsentwicklung erstellt.

Schließlich erwartet ein System wie JDaphne/DAPHNE und die zugehörige Web-Präsenz ein Mindestmaß regelmäßiger Wartung. Auf diesen Punkt wird unten bei der Beschreibung des Betriebs einer Web-Präsenz näher eingegangen. Die kritischen Stellen müssen den Administratoren im Rahmen der Schulung bekannt gemacht werden, damit sie diese routinemäßig überwachen und bei Problemen sinnvoll eingreifen können.

6.2.5 Systemintegration

An die Installation des Redaktionssystems muss sich die Integration in das Unternehmen anschließen. Eine Systemintegration kann über eine Reihe von Merkmalen klassifiziert werden. Solche Merkmale sind nach Mertens [Mer00] beispielsweise die Integrationsrichtung (horizontal, vertikal), die Integrationsreichweite (Bereichsintegration, innerbetriebliche Integration, zwischenbetriebliche Integration), der Integrationsgegenstand (Datenintegration, Funktionsintegration, Prozessintegration, Methodenintegration, Programmintegration) und der Automationsgrad (Teilautomation, Vollautomation). Als internes Informationssystem unterstützt das Redaktionssystem die innerbetriebliche Aufgabenerfüllung durch die kooperative Bearbeitung des Datenbestandes einer Web-Präsenz. Das webbasierte Vorgehen hilft, auf horizontaler Ebene über die verschiedenen Fachabteilungen des Unternehmens hinweg teilautomatisiert Daten und Prozesse zu integrieren.

6.2.5.1 Datenintegration

Die Datenintegration umfasst beim Redaktionssystem die gemeinsame Nutzung von Datenbeständen in zwei Richtungen. Zum einen werden über Importe bereits von anderen Anwendungen erzeugte und genutzte Daten für die Web-Präsenz aufbereitet. Zum anderen werden die für die Web-Präsenz erstellten Dokumente und Inhalte so exportiert, dass sie von anderen Anwendungen weiterverarbeitet und in anderen Kontexten genutzt werden können. Ein wichtiger Bestandteil der Datenintegration ist ein in abstrakter Form beschreibbares gemeinsames Austauschformat für die Daten. Sofern dieses vorliegt, können die Anwendungen auf Informationsebene (Semantik) miteinander kommunizieren. Im entgegengesetzten Fall findet, wie bei JDaphne über den Mediator-Server, die Einbindung externer Datenquellen über jeweils für spezielle Datensätze erstellte Abfrage-Repräsentanten statt. Der Mediator ist für die Anfrage und Aufbereitung der Daten zuständig. Die aufbereiteten Daten werden entweder statisch in das Redaktionssystem übernommen oder dynamisch bei Benutzeranfragen weitergereicht, nachdem sie über die Layout-Komponente des Redaktionssystems web-präsenz-konform gestaltet wurden. Umgekehrt findet der Export von Dokumenten der Web-Präsenz für andere Anwendungen über eine Aufbereitungskomponente, den Export-Server, statt. Seine Aufgabe ist das Verpacken der Daten in ein Austauschformat. Solche Exporte können beispielsweise einzelne Dokumentbausteine, z.B. die Metadaten, die in einem System zu den Dokumenten verfügbar sind, betreffen. Sofern inhaltlich strukturierte Dokumente mit dem System verwaltet werden, erfolgt der Export auf der Basis dieser Strukturen wiederum mit der Möglichkeit zur Nutzung auf einer semantischen Ebene.

6.2.5.2 Prozessintegration

Neben der Datenintegration, die mit dem Redaktionssystem betrieben wird, stellt die Prozessintegration eine große Herausforderung dar. Die Arbeit mit JDaphne/DAPHNE muss in den betrieblichen Arbeitsablauf integriert werden (vgl. [Tho01]). Für eine erfolgreiche Integration ist die Analyse der Workflows von Dateien und Informationen notwendig. Auf deren Basis können neue Prozesse definiert werden, die unter Einbeziehung des Redaktionssystems Arbeitsabläufe erleichtern. Weiterhin müssen bestehende Prozesse erweitert werden, um das durch die Web-Präsenz gebotene Potential zu nutzen. So ist es beispielsweise zwingend notwendig, die über Agenturen veröffentlichten Pressemeldungen mit dem Redaktionssystem auch für die Web-Präsenz zu verwerten. Zu diesem Zweck kann entweder der Vorgang „Pressemitteilung herausgeben“ erweitert werden, so dass die Pressemitteilung automatisch auch in das Redaktionssystem eingespeist wird oder ein neuer Prozess entwickelt werden, der die

Erstellung der Pressemeldung in dem Redaktionssystem vorsieht und über dieses ihre Publizierung verwaltet. Unter Nutzung der Workflow-Mechanismen des Redaktionssystems kann so die Publikation in einen Nachrichten-Verteiler gleichzeitig mit der auf der Web-Präsenz erfolgen. Damit wird zugleich sichergestellt, dass die Agentur in die Lage versetzt ist, über die Web-Präsenz des Unternehmens noch zusätzliche Hintergrundinformationen zu der Pressemitteilung abzurufen, sofern diese beispielsweise auf verwandte Themen verweist.

6.3 Life-Going und Betrieb der Web-Präsenz

6.3.1 Das Life-Going

6.3.1.1 Test von externen Rechnern

In einem ersten Schritt geht die Web-Präsenz für eine Testphase online, d.h. sie wird durch einen Internet-Provider über das Internet zugänglich gemacht. Falls die Web-Präsenz während dieser Testphase noch nicht öffentlich zugänglich sein soll, kann sie z.B. mit einem passwortgeschützten Zugang versehen werden. Diese im Internet sichtbare Version der Web-Präsenz muss einer eingehenden Überprüfung der Funktionalität unterzogen werden. Am besten eignet sich eine vorab erstellte Checkliste, welche die durchzuführenden Tests enthält. Um den Benutzer an einem beliebigen Ort im Internet zu simulieren, sollte die Web-Präsenz nicht von einem internen Internetzugang aus überprüft, sondern ein anderer Zugang gewählt werden. Damit werden Probleme treten Probleme zu Tage, wie sie sich einem Besucher aus dem Internet offenbaren. Auch die Ladezeiten der Dokumente sollten bei diesen Tests gewisse Limits nicht überschreiten. Sind Ladezeiten zu lang, sollte eine Überarbeitung des Layouts in Erwägung gezogen werden. Weiterhin müssen eventuelle dynamische Komponenten, z.B. CGI-Skripte oder Servlets, auf ihre Funktionstüchtigkeit untersucht werden. Die Verlinkung der Dokumente der Web-Präsenz sollte bei dem Einsatz eines Redaktionssystems fehlerfrei sein. Gerade zu Beginn ist der Einsatz von Link-Checkern jedoch ein Muss. Solange bei den Tests Probleme auftreten, müssen diese behoben und erneut überprüft werden. Am Ende der Testphase sollten alle Punkte der Checkliste abgehakt sein. Sind alle Tests zur Zufriedenheit verlaufen, kann die Web-Präsenz der Öffentlichkeit zugänglich gemacht werden.

6.3.1.2 Bewerben der Web-Präsenz

Ist eine Web-Präsenz online gegangen, so ist noch lange nicht damit zu rechnen, dass sie eine gehobene Anzahl von Besuchern anzieht. Die Web-Präsenz muss erst bekannt gemacht werden (Promotion). Das Bekanntmachen einer Web-Präsenz kann und sollte auf vielfältigen Wegen geschehen. Dem Medium Internet angepasste Wege sind Newsgroups, Portale und Suchmaschinen, aber auch konventionelle Wege wie Zeitungsannoncen, Werbebriefe bzw. Aufnahme der URI in die Geschäftskorrespondenz können wertvolle Hilfe leisten.

6.3.2 Betrieb

Wenn die Konzeptionsphase und der Aufbau der Web-Präsenz abgeschlossen sind, beginnt die Betriebsphase. In dieser werden die Zugriffe auf die Web-Präsenz analysiert und die inhaltliche Konzeption verifiziert sowie gegebenenfalls modifiziert. Dazu gehört neben der Pflege und Neuerstellung von Dokumenten auch ihre inhaltliche Aktualisierung. Um bei Systemschäden schnell ein Ersatzsystem ins Netz stellen, zu können sind Backup-Strategien zu entwickeln. Der Wunsch nach erweiterten Funktionalitäten führt zu Modifikationen am System und gegebenenfalls zum Relaunch.

6.3.2.1 Arbeiten mit JDaphne – Usability

Um die implementierten Mechanismen, An- und Übersichten praxisrelevant mit Beispielen diskutieren zu können, werden sie anhand des exemplarischen Einsatzes von JDaphne für die Verwaltung der Web-Präsenz des Instituts für Telematik betrachtet. Damit stehen reale Einsichten in den produktiven Betrieb einer Web-Präsenz mit JDaphne zur Verfügung, die helfen, die Praktikabilität der Umsetzung genauso wie das Konzept von JDaphne selbst zu bewerten. Mit einem Dokumentenumfang von ca. 2000 Dokumenten, ca. 100 Ressorts bei rund 30 Benutzern des Systems kommt die Web-Präsenz des Instituts einer der bei der Konzeption zugrundegelegten Zielumgebung für den Einsatz eines solchen Systems sehr nahe. Da alle im Umfeld des Instituts für Telematik tätigen Mitarbeiter und Hilfskräfte in dem Umgang mit der Internet-Technik zwangsläufig gut geübt sind, kann von einem bereits qualifizierten Benutzerkreis ausgegangen werden, wie er sonst nur durch entsprechende Schulungen zu erreichen ist.

6.3.2.1.1 Anmeldung am System

Um mit JDaphne zu arbeiten, müssen die Benutzer mit ihrem Web-Browser die Einstiegsseite vom Web-Server des Produktiv-Rechners anfordern. Die HTML-Seite enthält die für die JDaphne-Benutzerschnittstelle notwendige Parametrisierung des Java-Plug-ins. Dieses wird von dem Web-Browser initialisiert und damit das Herunterladen des signierten Java-Archivs mit den für das Applet notwendigen Java-Klassen gestartet. Sobald der Ladevorgang abgeschlossen ist, wird die Signatur des Codes gegen das vom Benutzer importierte Zertifikat des Code-Signers verifiziert. Nach erfolgreicher Verifikation wird der Benutzer gefragt, ob er dem Code-Signer vertraut (Abb. 5.9). Wird dies von dem Benutzer bestätigt, so wird die Ausführung des Java-Applets gestartet. Andernfalls wird an dieser Stelle der Start des Applets abgebrochen, weil das Plug-in Zugriffsverletzungen feststellt.

Im folgenden Schritt wird dem Benutzer die Möglichkeit zum Login gegeben. Er muss sich mit Benutzerkennung und Passwort identifizieren. Alternativ zu dem Zugang zum Redaktionssystem über die Benutzerkennung/Passwort-Variante ist auch der Zugang mittels Zertifikaten konfigurierbar. Zu diesem Zweck setzt die Authentisierungskomponente von JDaphne einen am Arbeitsplatz installierten Smart-Card-Leser ein. Dieser liest das Zertifikat von der Smart-Card aus, das vom Redaktionssystem gegen den konfigurierten Verzeichnisdienst geprüft wird. Verläuft die Verifikation erfolgreich, so wird der Zugang zu dem System mit den entsprechenden Autorisierungen gewährt.

6.3.2.1.2 Übersichten und Ansichten

Mit der erfolgreichen Authentisierung des Benutzers geht die Feststellung der von ihm wahrgenommenen Rollen einher. Basierend auf diesen Rollen wird die Benutzerschnittstelle des Applets konfiguriert. In ihrem Aussehen an ein Dateimanager-Programm (Abb. 6.3) erinnernd, zeigt sie auf der linken Seite eine Baumdarstellung der Ressort-Hierarchie. In den Ressorts verfügbare Dokumente werden in den tabellarischen Ansichten auf der rechten Seite dargestellt. Sie zeigen den Reitern entsprechend, welche Dokumente in den einzelnen Sub-Repositories abgelegt sind. Welche Reiter dabei für den Benutzer angezeigt werden, hängt von dessen Rolle ab. Auf diese Weise wird sichergestellt, dass jeder Rolle nur die Dokumente präsentiert werden, mit denen sie Aktionen ausführen kann. Welche Aktionen das sind, wird anhand der oben beschriebenen Rollen-Aktions-Matrix (siehe S. 114) bestimmt. Basierend auf den erlaubten Aktionen werden die Menüs der Benutzerschnittstelle angepasst. Nur diejenigen Menüpunkte werden aktiviert, die dem Benutzer zugänglich sind. Alle übrigen

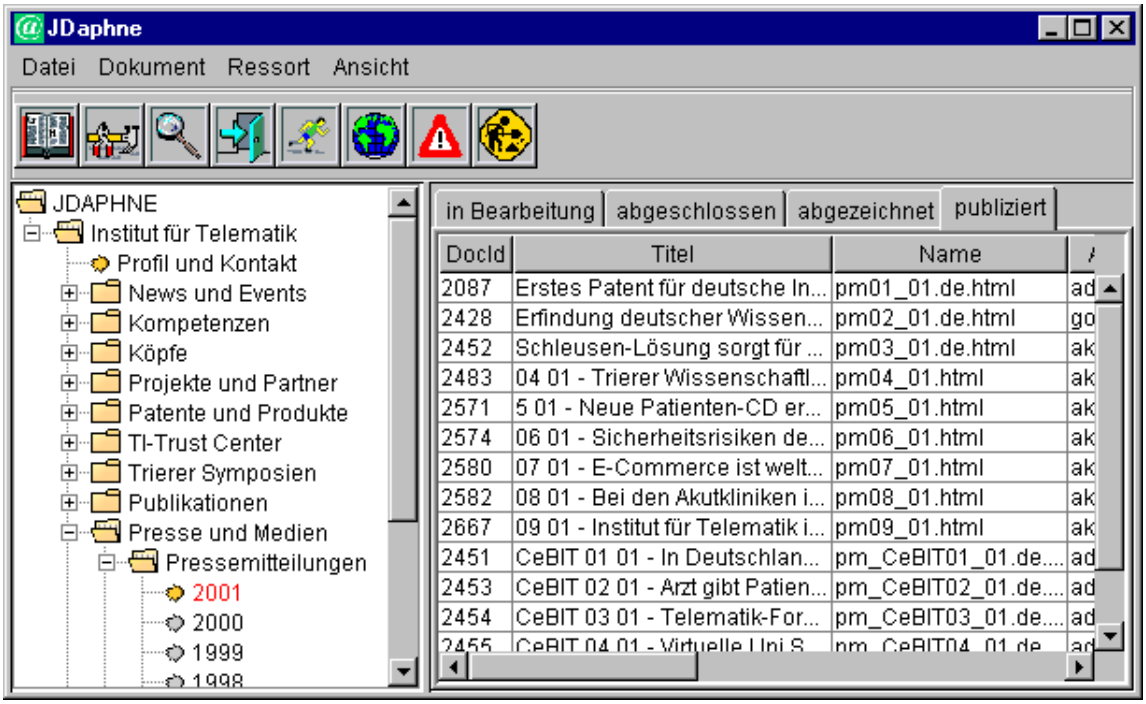

Abbildung 6.3: Ein Screen-Shot von dem Java-Applet als Benutzerschnittstelle. Der Benutzer ist in der Rolle als Administrator angemeldet und bekommt deshalb alle Repositories für Dokument-Zustände angezeigt.

Menüpunkte bleiben deaktiviert. Kontextsensitive Menüs, nach dem gleichen Mechanismus gesteuert, beschleunigen zusätzlich die Aktionen des Benutzers.

Innerhalb der Ansichten kann der Benutzer die Dokumente nach seinen Wünschen sortieren. Weiterhin wird die Übersichtlichkeit durch geeignete Filter stark verbessert. So wird die Einschränkung der in den Tabellen angezeigten Dokumente auf Dokumente im HTML-Format häufig ausgewählt. Dies insbesondere, weil so alle, den Zustand betreffende Aktionen, zugleich auch auf alle eingebundenen Dateien (Grafiken) wirken. Da die Benutzer über die Benutzerschnittstelle Aktionen mit Dokumenten starten, die serverseitig – bei der Selektion mehrerer Dokumente gleichzeitig – zum Teil recht zeitaufwendig abgearbeitet werden müssen, wurde bei der Entwicklung der Benutzerschnittstelle großer Wert darauf gelegt, dass der Benutzer jederzeit weitest mögliche Kenntnis über den Status der „remote“ ausgeführten Aktionen erhält. Zu diesem Zweck wird in der Benutzerschnittstelle die Ausführung der Aktionen in zwei Prozesse unterteilt: ein Prozess, der die Aktion „remote“ ausführt und ein anderer, der diesen Prozess überwacht und dessen Status anzeigt (Progressbar). Auf diese Weise wird dem Benutzer – wann immer möglich visualisiert – inwieweit die initiierten Aufgaben bereits erledigt wurden. Das Resultat seiner Aktionen wird dem Benutzer in einem protokollartigen Dialog (Abb. 6.4) aufgelistet. Somit kann er alle Aktionen in ihrer Ausführung bezüglich ihres Ergebnisses überprüfen. Während die bislang diskutierten tabellarischen Ansichten Auskunft darüber geben, welche Dokumente in den Sub-Repositories zur Verfügung stehen, ist die ToDo-Ansicht die in den meisten Fällen für die Arbeit mit dem System relevante Übersicht. Sie visualisiert dem Benutzer die nächsten im Workflow der Dokumente anstehenden Schritte in Form der Aktionen, die der Benutzer mit ihnen ausführen muß. Wieder anhand von Reitern kann sich der Benutzer die Dokumente mit ihren anstehenden Tasks in tabellarischer Form sortiert nach Aufgaben anzeigen lassen. Über ein kontextsensitives Menü werden die

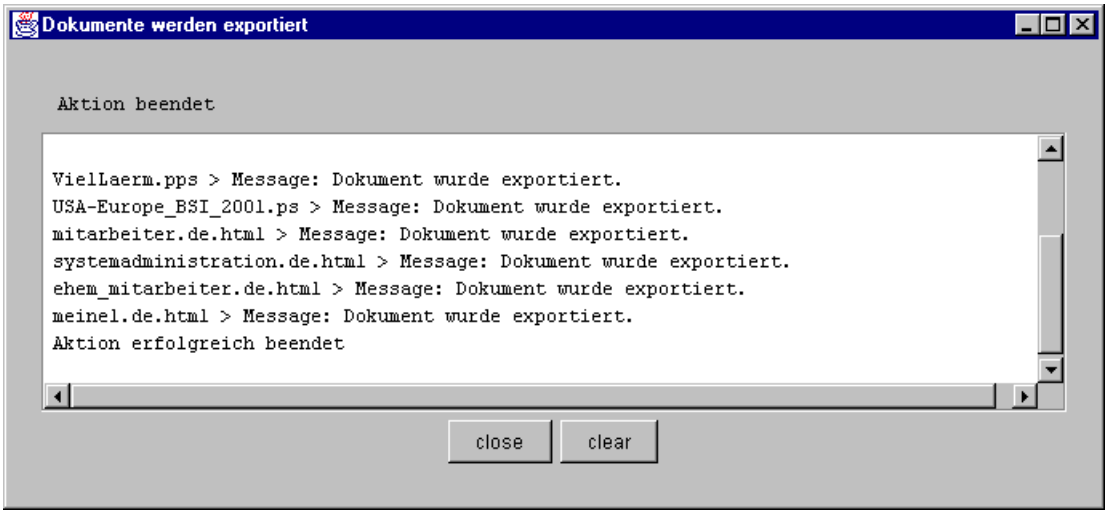

Abbildung 6.4: Der Informations-Dialog, der dem Benutzer mitteilt, wenn eine Aufgabe abgeschlossen wurde oder ob Probleme aufgetreten sind.

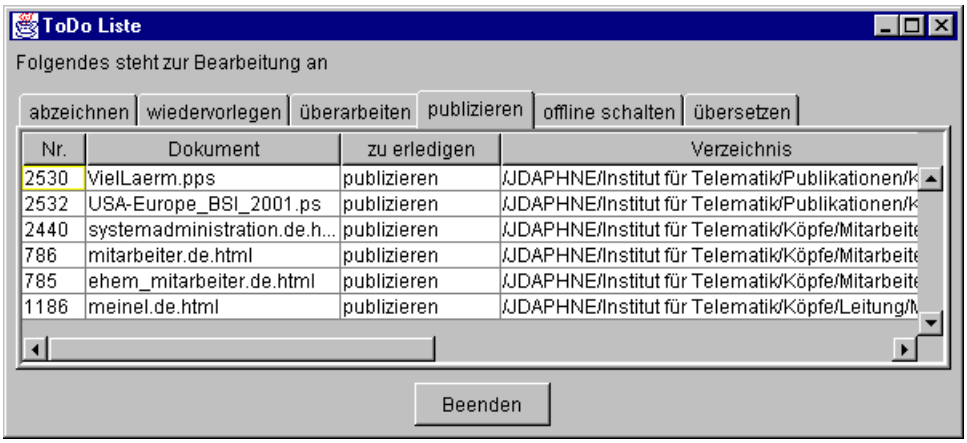

Abbildung 6.5: Ein Screen-Shot von der Aufgabenliste, die sämtliche anfallende Aufgaben darstellt, wie sie sich in der Administrationsübersicht präsentiert.

für die aktuell aufgelisteten Dokumente möglichen Aktionen angezeigt. Alle abgearbeiteten Tasks verschwinden aus der Übersicht.

Abgeschlossen werden die Dokumentansichten mit einer vollständigen Übersicht über alle einem Benutzer zugeordneten Dokumente. Diese werden ihm wieder anhand von Reitern selektierbar und nach Sub-Repositories geordnet in tabellarischer Ansicht angezeigt. Verschiedene weitere Dialoge und Fenster dienen der Darstellung und dem Editieren von Dokumentendaten. Sämtliche für ein Dokument relevante Hyperlinks lassen sich in tabellarischer Darstellung abrufen (Abb. 6.6). Über Reiter lassen sich die gewünschten Informationen selektieren: Broken-Links, Links, die in dem Dokument enthalten sind, und alle diejenigen Dokumente, die auf dieses Dokument verweisen. Dieses geschieht zwar ohne grafische Visualisierung, aber dafür in Abhängigkeit von den Sub-Repositories, in denen sich das Dokument befindet. Abgeschlossen wird diese Auflistung von Ansichten, Fenstern und Dialogen, mit denen sich die Ressorteigenschaften und bestimmte Metadaten für die Dokumente modifizie-

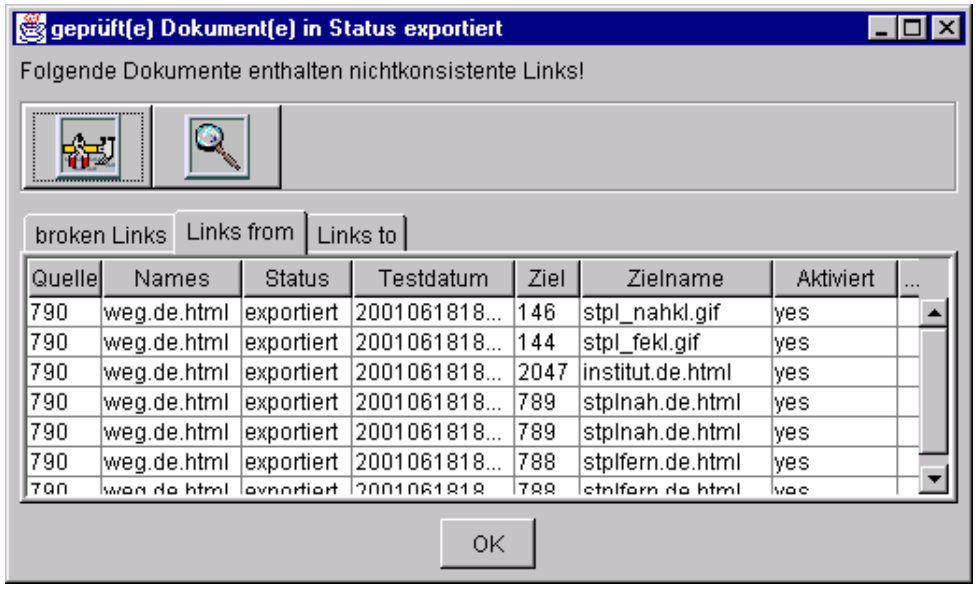

Abbildung 6.6: Tabellarische Ansicht der Hyperlinks, die von einem Dokument ausgehen. Die beiden anderen Reiter liefern analoge Übersichten auf Dokumente, die das aktuelle Dokument als Ziel haben. In der Ansicht „Broken-Links“ werden die Hyperlinks aus der Datei aufgelistet, die nicht zugeordnet werden konnten.

ren lassen. Mit ihnen können die Dokument- und Ressortdaten in Form von Textfeldern und Auswahllisten editiert werden.

6.3.2.1.3 Editieren eines Dokuments

Da der Bearbeitungsvorgang eines Dokuments bereits in Abschnitt 5.4.1.1 ausführlich beschrieben ist, wird hier lediglich auf die visuelle Umsetzung eingegangen. Nach der Überprüfung der Berechtigung des Benutzers und der Feststellung, dass das Dokument parallel von keinem anderen Benutzer bearbeitet wird, wird es aus dem „In Bearbeitung“-Sub-Repository gelesen. Das Applet fragt das Dokument von dem Dokumenten-Server ab und legt es unter dem aus der Datenbank abgefragten Original-Namen in dem konfigurierten Bearbeitungsverzeichnis auf dem lokalen Arbeitsplatz ab. Im gleichen Schritt wird eine Sperre (Lock) auf das Dokument gesetzt, die verhindert, dass ein anderer Benutzer ebenfalls dieses Dokument editiert. In einem nächsten Schritt wird festgestellt, mit welcher Anwendung der Benutzer Dokumente dieses Formats bearbeiten möchte. Sofern dies von vorhergehenden Bearbeitungsvorgängen bekannt ist, wird versucht, diese Anwendung parametrisiert mit der lokal auf dem Arbeitsplatz abgelegten Datei zu starten. Existiert die Anwendung auf diesem Arbeitsplatz nicht oder ist in einem anderen Verzeichnis installiert, so öffnet das Applet einen Dialog. Dieser fordert den Benutzer auf, diejenige Anwendung aus seinem Dateisystem zu selektieren, mit der er die Datei aus JDaphne bearbeiten möchte. Da die Benutzerpräferenzen in der Datenbank abgelegt werden, kann die Administration bei Bedarf diese Einstellungen an zentraler Stelle vornehmen und den Benutzer aus dem Konfigurationsprozess heraushalten. Nachdem sich die Editor-Anwendung mit dem Dokument aus JDaphne geöffnet hat, kann in der Folge das Dokument nach Gutdünken bearbeitet werden (Abb. 6.8). Weil es oft hilfreich ist, mehrere Dokumente gleichzeitig zu bearbeiten, dabei aber schnell die Übersicht über die zum Bearbeiten geöffneten Dokumente verloren geht, werden alle lokal abgelegten Dokumente in einem Dialog (Abb. 6.9) in einer Liste dargestellt.

Abbildung 6.7: Die Visualisierung von broken Hyperlinks (im Zustand „In Bearbeitung“) und nicht aktiven Hyperlinks (in allen anderen Zuständen) durch „Bömbchen“ mit der Voransicht eines Dokuments.

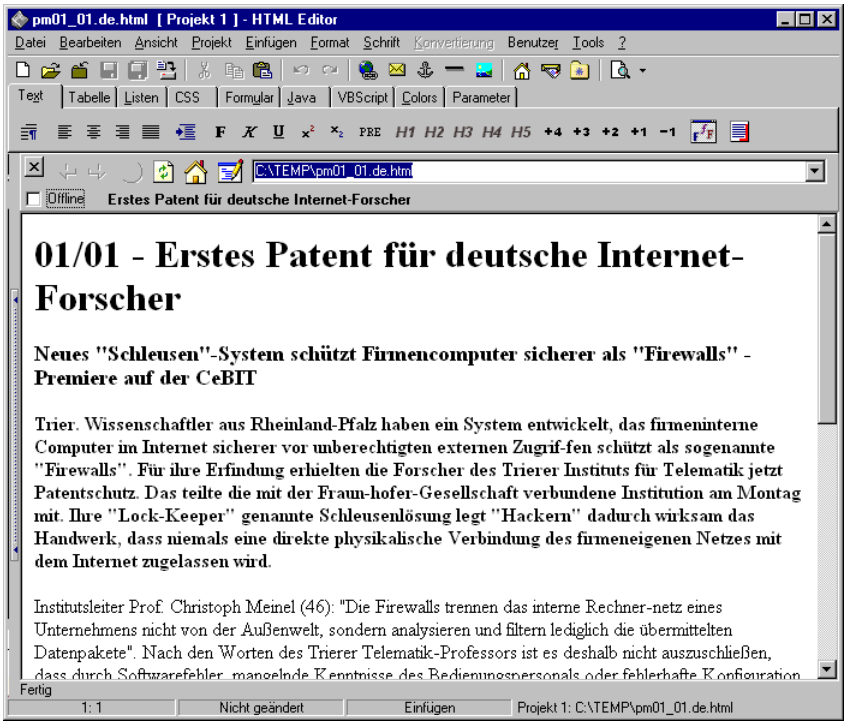

Abbildung 6.8: Das Dokument zum Bearbeiten im lokalen Editor.

Dieser Dialog hat weiterhin die Aufgabe, im Applet den Transfer des bearbeiteten Dokumentes zurück zu dem Dokumenten-Server in die Wege zu leiten. Über entsprechende Schaltflächen kann dieser Transfer auch ausgeführt werden, um das Zwischenresultat bereits in der Voransicht von JDaphne mit Layout und Navigation zu begutachten (Abb. 6.7). Diese visualisiert im Zustand „In Bearbeitung“ Hyperlinks, die noch nicht von JDaphne zugeordnet werden konnten. In den übrigen Zuständen symbolisiert eine analoge Voransicht den Deaktivierungsstatus eines Hyperlinks.

Parallel zur Voransicht kann das Dokument lokal weiter editiert werden, bis es den Anforderungen genügt. Ist dies der Fall, so verbleibt die lokale Kopie auf dem Arbeitsplatz des Benutzers. Das Dokument wird aus dem Bearbeitungsdialog entfernt und die Bearbeitungssperre für andere Benutzer aufgehoben. Sofern der Benutzer der Meinung ist, sein Dokument erfülle die Anforderungen der Kontrollinstanz, leitet er es zum Abzeichnen weiter. Werden im Laufe des Bearbeitungsvorgangs Grafiken in ein Dokument eingelagert, so müssen diese JDaphne über einen Importvorgang zugänglich gemacht werden. Anschließend müssen die Hyperlinks auf diese Dokumente umgesetzt werden.

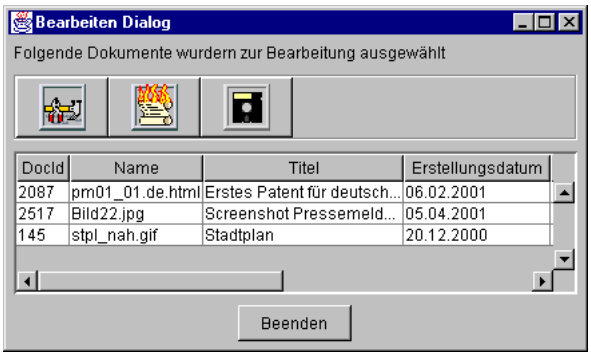

Abbildung 6.9: Eine Liste mit Dokumenten, die gerade lokal bearbeitet werden.

6.3.2.1.4 Setzen von Hyperlinks

Eine für die Benutzer arbeitsintensive Aufgabe bei der Erstellung und Bearbeitung eines Dokumentes ist dessen Einbindung in die Verknüpfungsstruktur der Web-Präsenz. Vorschläge für mögliche Hyperlink-Ziele können von dem System abgefragt werden. Letztendlich obliegt es jedoch dem Autor, während des Bearbeitungsvorgangs seine Dokumente mit Hyperlinks zu versehen. Einerseits ist mit JDaphne das Referenzieren von Dokumenten sehr einfach: lediglich die Nummer des Zieldokuments oder das Kürzel des Ressorts müssen angegeben werden, damit der Hyperlink eindeutig von JDaphne übernommen werden kann. Durch diese eindeutigen Dokument- und Ressortbezeichnungen, die außerdem noch prägnanter als sprechenden Namen sind, sinkt die Chance, dass der Benutzer Fehler macht. Andererseits kennt kaum ein Benutzer alle Dokumentennummern oder Ressortkürzel auswendig und ist deshalb zwingend auf die Übersichten und Visualisierung der Ressorthierarchie angewiesen, um die gewünschten Referenzen setzen zu können. Im Endeffekt bietet die Verwendung von Dateinamen in Hyperlinks, sofern die referenzierten Dokumente – dies ist häufig bei Grafiken der Fall – im gleichen Ressort liegen, einen hilfreichen Kompromiss. Insbesondere beim Import von Dokumenten mitsamt ihrer eingebundenen Grafiken ist dieser Weg eine echte Unterstützung der Autoren. Bewährt hat sich auch, dass mit der in JDaphne eingesetzten Hyperlink-Syntax die Originalnamen der verlinkten Dokumente als Kommentar in dem Anker-Tag erhalten bleiben. Dies gibt den Autoren während der Ansicht der Dokumente eine Vorstellung davon,

welches Dokument sich hinter einer als Nummer im Link sichtbaren Verknüpfung verbirgt. Dieser Mechanismus kann bei Bedarf noch in der Form weiter entwickelt werden, dass vor dem Transfer des Dokumentes vom Server auf den Client gezielt Mehrinformationen in einen Hyperlink eingetragen werden.

6.3.2.1.5 Digitales Signieren eines Dokuments

Dem prototypischen Charakter gerechtwerdend stellt JDaphne Methoden zum Signieren und Verifizieren von digitalen Dokumenten (vgl. Seite 53) zur Verfügung. Es handelt sich dabei um die Bereitstellung von digitalen Unterschriften für Dokumente zum Gebrauch innerhalb des Unternehmens, beispielsweise im Intranet. Der Signierprozess setzt, wie schon die alternative Anmeldung am System, Smart-Cards ein. Um ein Dokument im Redaktionssystem signieren zu können, muss es sich mindestens im Zustand „zum Abzeichnen“ befinden, d.h. vom Autor als fertiggestellt gemeldet sein, denn für die digitale Signatur wird die fertiggestellte Version eines Dokuments verwendet. Diese zeichnet sich dadurch aus, dass sie noch nicht über das sich eventuell mit jedem Export in die Web-Präsenz ändernde Layout verfügt. Jede Layout-Modifikation würde die Signatur ungültig machen. Deshalb bezieht sich die digitale Signatur auf die Roh-Version des Dokuments ohne Layout. Hyperlinks werden vor der Signatur nicht umgesetzt, weshalb diese nach wie vor auf Dokumentennummern verweisen. Für den internen Gebrauch, bei dem im Normalfall die dynamische Voransicht für das Dokument eingesetzt wird, stellt dies aber kein Hindernis dar.

Ein Benutzer, der ein Dokument digital unterzeichnen möchte [Dör01], muss dieses innerhalb des Redaktionssystems selektieren. Nach dem Aufruf der Signatur-Methode wird der Hash-Wert (vgl. Seite 53) dieses Dokuments berechnet und an die Smart-Card übergeben. Von dieser wird die eigentliche digitale Signatur erzeugt. Je nach Schutzvorrichtung, beispielsweise nach Eingabe einer PIN (Personal Identification Number), wird der Hash-Wert mit dem privaten Schlüssel des Unterzeichners encodiert. Diese Signatur wird zurück an das System übergeben und in einer Base 64 Codierung zusammen mit der Kennung des Signierenden in der Datenbank abgelegt.

Zum Verifizieren wird die digitale Signatur mit dem öffentlichen Schlüssel des Signierenden decodiert. Zu diesem Zweck wird aus dem Verzeichnisdienst das Zertifikat des Signierenden gelesen und auf seine Gültigkeit überprüft. Sofern das Zertifikat nicht widerrufen wurde und der Gültigkeitszeitraum nicht überschritten ist, wird der aus dem Zertifikat extrahierte Schlüssel genutzt, um den Hash-Wert zu entschlüsseln. Dieser wird mit dem anhand des Dokuments neu generierten Hash-Wert verglichen. Stimmen beide überein, wird die Signatur als gültig gewertet.

6.3.2.2 Statistische Auswertungen/Rückkopplung

Von ernormer Bedeutung beim Betrieb einer Web-Präsenz sind die Zugriffsauswertungen. Zugriffsstatistiken geben detailliert darüber Aufschluss [SZAS97], welche Dokumente wann und wie oft aufgerufen wurden. Weiterhin lassen die Kennungen der zugreifenden Rechner Rückschlüsse auf die Benutzerkreise zu. Über eine solche Auswertung kann entsprechend schnell festgestellt werden, ob mit der Web-Präsenz die zielgruppengerechte Umsetzung des Konzepts gelungen ist. Grundlage der Auswertungen sind die Log-Dateien der Web-Server (siehe bzgl. der rechtlichen Rahmenbedingungen z.B. den Artikel von Gerling [Ger01]). Diese protokollieren detailliert jeden Zugriff auf die von ihnen publizierte Web-Präsenz mit einem Eintrag in eine „Log-Datei“ oder alternativ in eine Datenbank. Welche der verfügbaren Informationen in diesen Log-Dateien vom Server abgelegt werden, ist frei konfigurierbar. Um die Auswertung der Dateien zu verallgemeinern, hat sich ein gängiges Format, das „Common Log

File Format“ durchgesetzt. Alle Auswertungsprogramme verstehen Log-Dateien, die die Zugriffe in dieser Form beschreiben. Die Zahl der einsetzbaren Auswertungsprogramme ist sehr groß. Ihr Spektrum reicht von einfachen statistischen Auswertungen mit grafischer Ergebnisdarstellung bis hin zu komplexen Auswertungen des Benutzerverhaltens mit Data-Mining Techniken [ZXH98, JK00, HRH+00a] und künstlicher Intelligenz [CSM97].

Über die Auswertungen lassen sich schnell Bereiche auf der Web-Präsenz identifizieren, die keinen oder nur wenig Publikumsresonanz erfahren. Auf Grundlage dieser Erkenntnis können die betroffenen Bereiche in geeigneter Weise [SPF99] modifiziert bzw. neu positioniert werden. Es hat sich außerdem bewährt, wie bei JDaphne, die Zugriffsinformationen allen an der Web-Präsenz Mitwirkenden über direkt abrufbare Statistiken zugänglich zu machen. Insbesondere für die Autoren einzelner Dokumente ist es hilfreich zu erfahren, wie oft das entsprechende Dokument abgerufen wurde. Diese Rückkopplung motiviert einerseits diejenigen Beteiligten, die ein positives Feedback erhalten haben und spornt andererseits weniger erfolgreiche Autoren zu Modifikationen an.

Neben den die Zugriffe auf Dokumente betreffenden Log-Dateien des Web-Servers, der die Dokumente verteilt, fallen noch eine Reihe anderer Log-Dateien an. Dies sind zum einen die Fehlermeldungen, die in einer eigenen Log-Datei gesichert werden, und zum anderen die Log-Dateien der entsprechenden dynamischen Komponenten, die in die Web-Präsenz integriert sind. Allesamt benötigen aufmerksame Wartung, um einen reibungslosen Betrieb der Web-Präsenz zu gewährleisten.

6.3.2.3 Sicherheit

Die Betriebssicherheit einer Web-Präsenz ist kein Zustand, sondern ein Prozess. Nur wenn das System ständig überwacht wird, fallen Probleme mit ihm sofort auf (siehe Abschnitt 3.2.5). Dabei können die Probleme sowohl ihre Ursache in internen Prozessen haben oder von außen in das System eingebracht werden (Angriffe). Durch den Einsatz entsprechender Programme, welche oft unter dem Namen „Watch-Dog“ zu finden sind, können die wichtigsten Systemkomponenten überwacht werden. Dies betrifft sowohl den verfügbaren Platz im Dateisystem als auch das Vorhandensein und die Zahl der Server-Prozesse.

Zu dem ordnungsgemäßen Betrieb einer Web-Präsenz gehört auch die regelmäßige Sicherung des Programm- und Datenbestandes. Beim Einsatz eines Redaktionssystems lässt sich der Dokumentenbestand durch einen Neu-Export verhältnismäßig einfach restaurieren. Dies gilt jedoch nicht für die auf dem Web-Server arbeitenden Programme sowie deren Konfigurations- und Datendateien. Um die Ausfallzeiten bei dem Auftreten eines Hard- oder Softwareproblems so gering wie möglich zu halten, sind funktionsfähige Backups unbedingt notwendig. Diese am besten täglich aktualisierten Sicherungskopien müssen sorgfältig, unter Berücksichtigung der entsprechenden Datenschutzbestimmungen, gelagert und gegebenenfalls nach entsprechender Zeit wieder gelöscht werden.

Im Laufe der Zeit werden bei den eingesetzten Hard- und Softwarekomponenten Aktualisierungen notwendig. Dies geschieht einerseits, um bekannt gewordene Sicherheitslücken in der Software zu schließen, andererseits werden beim erfolgreichen Betrieb der Web-Präsenz von Zeit zu Zeit Leistungssteigerungen nötig. Nur wenn den Benutzern hinreichend kurze Antwortzeiten geboten werden, lassen diese sich zum Verbleib innerhalb der Web-Präsenz animieren. Weiterhin bringt die Einführung neuer Techniken, z.B. von sicheren Verbindungen, eine oft unterschätzte Leistungsanforderung mit sich. Problematisch sind Updates immer, da nie ausgeschlossen werden kann, dass Inkompatibilitäten zwischen den Versionen auftreten. Aus diesem Grund sind alle Updates im voraus auf Test-Maschinen zu überprüfen. Bei diesen sorgfältig durchzuführenden Untersuchungen können Probleme identifiziert und gegebenenfalls behoben werden. Sind die Probleme zu groß, ist zu entscheiden, ob das Update zwingend

notwendig ist oder ob noch eine gewisse Zeit mit dem aktuellen Stand überbrückt werden kann, bevor tiefergreifende Änderungen durchgeführt werden müssen. Solche „größeren“ Updates, die viele Modifikationen mit sich bringen, sind oft ein guter Punkt, die Konzeption nicht nur technisch, sondern auch inhaltlich zu überdenken. In vielen Fällen ist dann ein Relaunch der Web-Präsenz die Folge.

6.3.3 Evolution/Relaunch

Im schnelllebigen Internet-Geschäft vergeht oft nicht viel Zeit bis eine Technik als nicht mehr trendgemäß betrachtet wird. Gleiches gilt für das Design und den Aufbau von Web-Präsenzen. Werden die Differenzen zwischen aktuellem und angestrebtem Stand zu groß als dass sie sich mit normalen Updates und Restrukturierungen beheben ließen, wird ein Relaunch notwendig; die Konzeption wird den aktuellen Anforderungen angepasst. Steht das neue Ziel fest, so muss der Weg vom aktuellen bis hin zum neuen angestrebten Stand konzeptioniert werden. Können Komponenten übernommen werden, so müssen diese migriert werden, d.h. es müssen Mechanismen geschaffen werden, die diese Komponenten mehr oder weniger automatisiert in den anvisierten Zustand überführen. Ein solcher Restrukturierungsprozess sollte dazu genutzt werden, die Web-Präsenz und die darunter liegende Technik ebenfalls zu modernisieren. Häufig gibt es aber an dem Stand der Web-Präsenz nichts zu bemängeln. In dieser Situation wird lediglich das Einbringen von neuen Funktionalitäten angestrebt. Diese a priori einfach klingende Aufgabe gestaltet sich aufgrund der komplexen Zusammenhänge der beteiligten Komponenten häufig schwieriger als erwartet. Da Zusatzfunktionalitäten oft mit neuen Techniken verbunden sind, treten nicht selten Inkompatibilitäten auf, die den Austausch größerer Komponenten erfordern. Daher ist die Integration von neuen Funktionalitäten in der Praxis oft nicht weit von einem Relaunch entfernt.

Kapitel 7

Konzepte und Systeme zum Web-Präsenz-Management – Einordnung von JDaphne

Moderne Web-Präsenzen lassen sich am besten als Mischung aus Hyper-Media nach Nielsen [Nie95] und Informations-Systemen nach Davis und Olson [DO85] – hypermediale Informationssysteme – beschreiben. Um den vielen verschiedenen Ausrichtungen von Web-Präsenzen Rechung zu tragen, werden für ihren Aufbau und ihre Verwaltung vielfältige Funktionalitäten und Mechanismen benötigt. Aufgrund des großen Bedarfs an Werkzeugen haben sich mit der wachsenden Relevanz von Web-Präsenzen eine Reihe von Konzepten und Ansätzen zur Bewältigung dieser Aufgaben entwickelt. In einem weitgefassten Überblick über die in diesem Kontext einschlägigen Konzepte und Systeme schafft dieses Kapitel die Grundlage für eine Einordnung des mit JDaphne am Institut für Telematik entstandenen Systems.

Da sich im wissenschaftlichen Umfeld vorrangig zur Erprobung neuer Konzepte entwickelte und auch zum Einsatz gebrachte Anwendungen nur schwer mit den umfassenden Produkten kommerzieller Anbieter vergleichen lassen, wird die folgende Übersicht in zwei Bereiche unterteilt: im ersten Abschnitt werden die Ansätze zum Web-Präsenz-Management von kommerziell verfügbaren Systemen betrachtet, der zweite Abschnitt befasst sich mit den Ansätzen und Konzepten, die im wissenschaftlichen Umfeld des Web-Präsenz-Managements diskutiert werden. In den Fällen, in denen aus der wissenschaftlichen Arbeit ein Produkt (z.B. der Hyperwave Server, der aus HyperG [AKM95] abgeleitet wurde) entstanden ist, wird dieses in beiden Kontexten eingeordnet.

Die Untersuchung von Fraternali [Fra99] stellt mit ihrer umfangreichen, auch historischen Evaluation kommerzieller Systeme eine gute Grundlage des die kommerziellen Systeme klassifizierenden Abschnitts in diesem Kapitel dar. Trotz der vergleichsweise großen Breite betrachteter kommerzieller Applikationen kann dieses Kapitel allenfalls eine nach Kategorien gegliederte zusammenfassende Übersicht über die jeweils relevanten Anwendungen geben. Die Konzepte und Ansätze der Systeme aus dem wissenschaftlichen Umfeld werden, eingeordnet in anwendungsspezifische Kategorien, bezüglich ihrer besonderen Eigenschaften betrachtet, die im Zusammenhang mit der Entwicklung von JDaphne relevant erscheinen. Den meisten ist gemeinsam, dass es prototypische Einsätze gegeben hat, in denen die Tragfähigkeit ihrer Konzepte evaluiert wurde. Bereits vom möglichen Ressourcenaufwand gegenüber kommerziellen Produkten stark eingeschränkt, fokussieren diese Systeme auf einzelne, spezielle Funktionalitäten und Konzepte, die sie aus der Masse der verfügbaren Systeme abheben, legen grundsätzlich aber wenig Wert auf eine vollständige Nutzbarkeit als Gesamtanwendung.

Die am Institut für Telematik entwickelten Applikationen entstehen, von der Konzeption

des Instituts ausgehend, auf einer Gratwanderung zwischen kommerziellen und wissenschaftlichen Aspekten. An der Bugwelle der Entwicklung schwimmend werden Pilotsysteme für den produktiven Einsatz im Unternehmen maßgeschneidert entwickelt. Auf der einen Seite werden im Projektumfeld im Unternehmen tiefe Einblicke in die benötigten Funktionalitäten und Ideen für mögliche Konzepte gewonnen; durch diese konkreten Anforderungen wird der Entwicklungsprozess stark befruchtet. Auf der anderen Seite muss am Ende eines jeden Projekts ein speziell auf die Bedürfnisse angepasstes einsetzbares Pilotsystem entstehen. Aus der Forderung nach dessen Funktionstüchtigkeit und Einsetzbarkeit erwachsen offenkundig Randbedingungen für die konzeptionelle und implementatorische Arbeit. Diese verhindern die Verwendung von Technologien, die sich noch in einem so frühen Stadium befinden, dass ihr Einsatz den Umgang mit dem entwickelten System eher behindert als vorantreibt. Unter Berücksichtigung dieser Gegebenheit erfolgt, basierend auf der Übersicht kommerzieller und wissenschaftlicher Systeme, bei deren Darstellung bereits komparativ Funktionalitäten von JDaphne diskutiert werden, eine summarische Einordnung von JDaphne.

7.1 Systeme aus dem kommerziellen Umfeld

Die große Anzahl von kommerziellen Systemen, die auf dem Markt angeboten werden, macht eine vollständige Evaluation des Angebots unmöglich. Im Rahmen dieses Abschnitts wird dennoch eine umfangreiche Klassifizierung von Konzepten und Anwendungen versucht. Die große Zahl der, hier im Kontext von JDaphne und DAPHNE bewerteten Systeme, mit denen Web-Präsenzen erstellt und verwaltet werden können, ist grundlegend der Untersuchung von Piero Fraternali [Fra99] zu verdanken. Während für die vorliegende Arbeit die Frage im Vordergrund steht, wie in einem Unternehmen die in einer Web-Präsenz sichtbar zu machenden Informationen erstellt, verwaltet und publiziert werden können, fokussiert Fraternali auf die Frage, inwieweit die untersuchten Systeme die abstrakte Planung und automatisierte Durchführung des Publikationsprozesses ermöglichen. Dazu betrachtet er Web-Präsenzen als datenintensive Applikationen, sogenannte Webapplikationen. Diese werden durch drei Hauptaspekte charakterisiert: Sie dienen der Organisation von Informationen als Teile von strukturierten Inhalten, die untereinander in semantischen Beziehungen stehen. Weiterhin stellen sie die Navigationseigenschaften bereit, die den Benutzern den Zugriff zu den strukturierten Informationen geben. Schließlich findet durch sie die Präsentation der Informationen mittels Aufbereitung von Teilen der strukturierten Inhalte zusammen mit den Navigationseigenschaften für die Darstellung statt. Wichtigstes Kriterium für die Bewertung der Konzepte und Anwendungen ist der Grad der Trennung dieser drei Charakteristika bei der Entwicklung einer Web-Präsenz (-applikation). Damit fließen auch für die Einordnung von JDaphne/DAPHNE interessante Aspekte in seine Untersuchung ein. Auch wenn bei Fraternalis Betrachtungen insbesondere von bereits existierenden, strukturierten oder unstrukturierten Datenbeständen ausgegangen wird, gibt die Analyse der in den Anwendungen zur Abstraktion bei der Publikation genutzten Methoden wertvolle Erkenntnisse für die Einordnung von JDaphne/DAPHNE. Im Kontext dieser Arbeit muss aber zusätzlich stärker auf die verteilte Erstellung (Authoring) und das Management von Inhalten eingegangen werden.

Bei der nachfolgend aufgeführten Produktübersicht handelt es sich bei der nahezu unüberschaubaren Menge von am Markt verfügbaren Werkzeugen und Ansätzen lediglich um einen Überblick, bei dem nur die prominentesten Vertreter jeder Gattung genannt werden. Um auch der historischen Entwicklung von Konzepten und Anwendungen Rechnung zu tragen, sind für die Klassifizierung auch Produkte hinzugezogen, die sich im Einsatz befinden, deren Vertrieb mittlerweile allerdings durch die Hersteller eingestellt wurde, da sich nachfolgende Produktgenerationen angeschlossen haben.

Insgesamt ist auf dem Markt der Trend zu integrierenden Lösungen ungebrochen. Spezialisiert in Web-Content-Management-Systemen oder in allgemeinerer Form mittels Application-Servern wird die Systemintegration (vgl. Seite 162) in den Unternehmen vorangetrieben. Die folgende Betrachtung der auf dem Markt verfügbaren kommerziellen Konzepte und Anwendungen wird anhand nachstehender Klassifizierung vorgenommen:

1. Visuelle Editoren und einfache Web-Präsenz-Manager
2. Anwendungen zur Erstellung von Hyper-Medien mit Web-Export
3. Datenbank-Integrationslösungen mit Web-Export
4. WWW-Formular-Editoren, Groupware, Berichterstellungswerkzeuge und Datenbank-Publizierungsassistenten
5. Anwendungen mit Kombination verschiedener Techniken
6. Dokument- und Content-Management-Systeme mit WWW-Export-Schnittstellen
7. Modellbasierte, auf objektorientiertem Design der Web-Präsenz und ihrer Workflows beruhende Web-Präsenz Generatoren
8. Web-Content-Management-Systeme

Für die Bewertung der Anwendungen wird jeweils der Lebenszyklus (siehe Seite 64) einer Web-Präsenz zugrundegelegt. Ausgehend von einer für die Web-Präsenz bereits vorhandenen Anforderungsanalyse und einem durch prototypische Umsetzungen evaluierten Konzept, werden alle Anwendungen auf ihre Einsetzbarkeit hinsichtlich des Designs der oben aufgeführten drei charakterisierenden Aspekte betrachtet. Weiterhin wird die Implementation und schließlich die Verwaltung und Evolution der entstandenen Web-Präsenzen bewertet. Neben der Art und Weise, wie eine Anwendung den Lebenszyklus der mit ihr erstellten und verwalteten Web-Präsenz unterstützt, spielen zusätztlich noch weitere Faktoren für die Bewertung eine Rolle. Diese sind der Grad der Automatisierung, der für die Webapplikation geboten wird, das mögliche Abstraktionsniveau und die Wiederverwendbarkeit von in der Webapplikation umgesetzten Elementen. Abgerundet wird die Einordnung von den Möglichkeiten, die die den Systemen zugrundeliegende Architektur bietet sowie der Überprüfung der bei ihrem Einsatz entstehenden Unterstützungen im Web-Präsenz-Design und Betrieb der Web-Präsenz.

7.1.1 Visuelle Editoren und einfache Web-Präsenz-Manager

Ursprünglich erwachsen aus dem Bedürfnis, HTML-Code nicht mehr manuell erstellen und dateisystembasierte Web-Präsenzen nicht ohne Hilfsmittel verwalten zu müssen, sind in den vergangenen Jahren eine Vielzahl von visuellen HTML-Editoren und Web-Präsenz-Managern entstanden. Ihre Hauptfunktion liegt in der Erstellung von simplen Web-Präsenzen mit hochwertigen HTML-Dokumenten ohne große Kenntnis des zugrundeliegenden Datenformats (HTML). Dementsprechend ist ihre Kernkomponente ein WYSYWYG-Editor, der analog zu einer Textverarbeitung die Erstellung und Modifikation von HTML-Dateien ermöglicht. Hauptaugenmerk wird dabei vor allem auf die Unterstützung der neuesten Standards und Erweiterungen gelegt. Für die visuelle Erzeugung von XML-Code ist die Entwicklung dieser Kategorie von Anwendungen noch im Anfangsstadium und bei den Anwendern besteht dementsprechender Schulungsbedarf beim zweckgemäßen Umgang mit den jeweiligen XML-Varianten.

Mittels Templates, die von den Autoren nur noch mit Inhalt gefüllt werden, wahren die Programme dieser Kategorie (Ausnahme: die neuen XML basierten Werkzeuge) auf der Erstellungsebene die grafische Konsistenz der Web-Präsenz. Neben der Erstellung von Dokumenten gehören zum Funktionsumfang dieser Kategorie von Werkzeugen der Upload des Dokumentenbestandes auf den Web-Server, das Umbenennen, Löschen und Verschieben von Dokumenten sowie das zugehörige Link-Management. Hochwertige Produkte verfügen bereits über rudimentäre Mechanismen, die das Design der hierarchischen Struktur getrennt von den Inhalten ermöglichen. Eine Anzahl der hier besprochenen visuellen Programme verfügt weiterhin über grafische Darstellungen, mit denen sie einen Überblick über die von ihnen erzeugte Web-Präsenz generieren können.

Gemessen an diesen Funktionalitäten bieten die Web-Präsenz-Management-Anwendungen dieser Kategorie hauptsächlich Unterstützung für die Erstellung der Web-Präsenz und die initiale Produktion von Inhalten. Sie sind damit eine hervorragende Lösung für kleine bis mittelgroße Web-Präsenzen, bei denen die Dynamik und insbesondere der Workflow für die Erstellung der Dokumente nicht im Vordergrund stehen. Die Unterstützung von kooperativer Arbeit an Web-Präsenzen ist kein Ziel dieser Anwendungen, Mechanismen für die verteilte Erstellung und die Publikation von Inhalten basierend auf angepassten Workflows sind nicht implementiert. JDaphne und DAPHNE lassen sich dieser Kategorie in den Bereichen des Web-Präsenz-Managements zuordnen. Da sowohl JDaphne als auch DAPHNE mit bereits fertiggestellten Inhalten umgehen, sind die in dieser Kategorie vorgestellten Anwendungen in vielen Fällen diejenigen Editor-Anwendungen, mit denen für JDaphne/DAPHNE Inhalte erstellt und bearbeitet werden. Der Mehrwert von JDaphne/DAPHNE entsteht durch die implementierten Workflows. Dokumente werden nach Durchlauf eines Qualitätssicherungs-Prozesses hyperlinkkonsistent auf die Web-Präsenz publiziert.

Für die Publikation von großen, in Datenbanken gespeicherten Informationsmengen sind die dieser Kategorie zuzuordnenden Programme nicht konzipiert. Mit dem Fortschreiten der Entwicklung werden diese Restriktionen jedoch zunehmend aufgehoben. Die fortschrittlichsten der heute angebotenen Programme bieten in ihren aktuellen Versionen die Möglichkeit, auch dynamisch Verbindungen zu Datenbanken aufzubauen und die Resultate der Abfragen den Benutzern zur Verfügung zu stellen. Die vergleichsweise einfach gehaltene Architektur kommt diesen Anforderungen jedoch wenig entgegen. Neben der Verwendung des lokalen Dateisystems kommen in Ausnahmen Two-Tier-Architekturen zum Einsatz. Insbesondere Plattformunabhängigkeit ist kein Merkmal dieser Kategorie, die sich primär in der Microsoft-Windows Welt zu Hause fühlt. Zu den dieser Kategorie zuzuordnenden Anwendungen, die in den letzten Jahren größeren Bekanntheitsgrad erfuhren, zählen: *Microsoft's Frontpage 97*, *Adobe SiteMill*, *Adobe PageMill*, *SofQuad's HotMetal Pro* und neu auch *XMetal*, *Alaire's Homesite*, *Claris Home Page*, *Macromedia's Backstage Designer* und *Dream Weaver* sowie *Symantec's Visual Page*. Weniger für die Erstellung von Web-Präsenzen geeignet, aber dennoch viel genutzt sind visuelle Editoren auf dem Standard von *Netscape's Composer*. Unter Einbeziehung auch weniger bekannter Produkte, wie den in der Praxis für JDaphne/DAPHNE eingesetzten freien *HTML-Editor von Ulli Meybohm* läßt sich diese Liste noch beliebig verlängern. Da der Höhepunkt der Entwicklung solcher Programme zwischen 1993 und 2000 gelegen hat, sind sie mit einem großen Verbreitungsgrad im Einsatz, obwohl der Vertrieb einiger der hier aufgeführten Produkte mittlerweile entweder ganz eingestellt oder zugunsten darauf aufbauender Web-Content-Management-Systeme abgelöst wurde.

7.1.2 Anwendungen zur Erstellung von Hyper-Medien mit Web-Export

Trotz relativ geringer Relevanz für die Entwicklung von JDaphne/DAPHNE, nur der Vollständigkeit halber, wird diese Kategorie von lediglich für den Aufbau und Betrieb weniger Arten von Web-Präsenzen geeigneten Werkzeugen aufgeführt. Ursprünglich aus dem Umfeld des Offline-Publizierens stammend, bieten die im folgenden charakterisierten Werkzeuge die Mittel für die Erstellung hochwertige Hypermedien, z.B. Multimedia-CD-ROMs. Dementsprechend liegt diesen Werkzeugen häufig eine Erstellungs-Metapher in der Form „Erstellung eines Buches oder eines Filmes“ zugrunde. Über aus dem Bedarf erwachsene WWW-Schnittstellen lassen sich mittlerweile aus diesen Hypermedien beispielsweise über einen HTML-Export, gegebenenfalls mit Generierung clientseitiger Dynamik über Java(script), auch Web-Präsenzen erzeugen. Beispiele für mit solchen Werkzeugen erstellte repräsentative Web-Präsenzen stellen die häufig zu Kinofilmen produzierten multimedialen Web-Präsenzen dar. Hier wird deutlich, dass der Fokus dieser Werkzeuge in der Erstellung, designerischen Gestaltung und Verknüpfung von Inhalten liegt – mit dem Ziel, den fertigen Datensatz, der oft ad hoc zusammengestellt wurde, final zu publizieren. Die große Kontrolle über die grafische Umsetzung wird im Allgemeinen durch Restriktionen in der Flexibilität bezahlt, eine Einschränkung, die aufgrund des „Wegwerf-Charakters“ der damit erstellten Web-Präsenzen jedoch kaum eine praktische Rolle spielt. Werkzeuge dieser Kategorie sind gekennzeichnet durch große designerische Unterstützung während der Erstellung von Web-Präsenzen; ihre Verwaltung hingegen wird wenig unterstützt. Die Synchronisation multimedialer Objekte und gegebenenfalls daraus resultierende Navigation stellt wiederum einen herausragenden Aspekt dar. Eine definitive Einordnung der Architektur ist nicht möglich. Viele Anwendungen sind monolithisch aufgebaut, aber auch Two-Tier-Architekturen, die über den Web-Export realisiert werden, und Three-Tier-Architekturen bei der Einbindung anderer Datenquellen (Datenbanken oder statische Inhalte) sind denkbar. Zu der Gruppe der in dieser Kategorie zu platzierenden Werkzeuge gehören z.B. *Asymetrix Toolbook II Assistant/Instructor*, *Macromedia Director und Authorware* (Flash) sowie *Allen Communication Quest*.

7.1.3 Datenbank-Integrationslösungen mit Web-Export

Die kennzeichnende Gemeinsamkeit dieser Kategorie von Anwendungen ist ihre Unterstützung der Erstellung von Dokumenten für Web-Präsenzen dynamisch aus in Datenbanken gespeicherten Informationen. Die Integration zwischen Web-Präsenz und Datenbank erfordert jedoch Programmierfähigkeiten, da sie in vielen Punkten auf Programmierebene realisiert werden muss. Während die Systeme, die in den vorherigen Abschnitten diskutiert wurden, auch für Nicht-Entwickler auf visueller Basis den Aufbau von datenbankbasierten Kleinstapplikationen ermöglichen, liegt die Stärke der zu dieser Kategorie gehörenden Systeme in der großen Mächtigkeit und Flexibilität, die mit der Integration von Web-Präsenzen und Datenbanken auf der Programmierebene einhergeht. Zu diesem Zweck werden die Vorteile der „Internet-Sprachen“ wie HTML und Java mit Datenbankabfrage- und Programmiersprachen verbunden. Java-Erweiterungen, z.B. mit einem gemeinsamen Interface wie JDBC [Ree97], das auch von JDaphne zum Datenbankzugriff genutzt wird, helfen dabei, die Interoperabilität mit verschiedenen Datenbank-Management-Systemen (DBMS) zu gewährleisten. In den meisten Fällen, wie JDaphne in einer datenbankzentrierten Three-Tier-Architektur aufgebaut, können mit diesen Anwendungen bedingt abstrahierbare Applikationen entwickelt werden. Aufgrund der hohen Abhängigkeit von der Programmierlogik ist die Trennung von Struktur, Navigation und Datenpräsentation in den finalen Dokumenten nicht einfach zu realisieren und daher die mögliche Abstraktionsstufe noch recht niedrig. Trotz allem findet sich bei diesem Ansatz eine viel stärkere Trennung von Inhalt und Präsentation als bei den bislang beschrie-

benen Kategorien, da die Inhalte definitionsgemäß in der Datenbank und damit separiert von der Logik abgelegt sind. Die Konzepte dieser Kategorie sind bedingt mit JDaphne/DAPHNE verwandt; anders als die hier beschriebenen Anwendungen fokussiert JDaphne nicht allein auf die Publikation, sondern legt mit den Benutzerschnittstellen und Workflows großen Wert auf die Erstellung und Verwaltung von Inhalten durch die Mitarbeiter eines Unternehmens. Auch wenn JDaphne mittels JDBC Inhalte aus beliebigen Datenbanken dynamisch und statisch in die Dokumente der Web-Präsenz integriert und diese Funktion über Mediatoren [Wie92] auf der Programmierebene über die Abfrage von parametrisierten HTML-Erweiterungen realisiert sind, ist der Ansatz ein anderer, weniger allgemeinerer. Entsprechend bietet es sich für den Einsatz von JDaphne/DAPHNE an, alternativ Datenbankinhalte über die Einbindung von Web-Dokumenten, die dynamisch oder statisch über den Web-Export einer Datenbankintegrationslösung erzeugt werden, einzubinden. Mit entsprechenden Anpassungen im Layout bzw. durch Nutzung der Layout-Mechanismen von JDaphne/DAPHNE können so konsistente Web-Präsenzen erstellt werden.

Kommerzielle Produkte lassen sich innerhalb dieser Kategorie in zwei Subkategorien einordnen – die HTML-Erweiterungen und die Datenbankprogrammierschnittstellen-Erweiterungen. Erstere nutzen Templates, die analog zu den templatebasierten Export-Funktionen von JDaphne HTML-Code mit „ausführbarem“ Programm-Code vermischen. Während der Laufzeit des Systems werden die Templates gelesen und von einer Art Interpreter ausgeführt. Die durch die Ausführung des Programm-Codes erzeugten Daten werden in das Template an der Stelle des Programm-Codes eingebettet und an den anfragenden Client, den Web-Browser, zurückgeliefert. Zu den für HTML eingeführten Erweiterungen gehören neben den Datenbankanbindungen in den meisten Fällen noch einfache Elemente, mit denen ein Kontrollfluss aufgebaut werden kann und die Modularisierung erlauben. Bei den Datenbankprogrammierschnittstellen-Erweiterungen wird die für die Programmierung des DBMS vorhandene API um Methoden erweitert, mit denen vorgefertigt HTML-Code als Resultat einer Programmausführung zurückgeliefert werden kann. Entwickler müssen hier Datenbankanwendungen schreiben, die anhand der Datenbankabfragen HTML-Ausgaben generieren, ein Konzept, dass bei den Export-Skripten von DAPHNE umgesetzt wird.

Beispiele für kommerzielle Produkte, die mit HTML-Erweiterungen arbeiten, sind *Allaire Inc. Cold Fusion Web Database Construction Kit*, *Microsoft Active Server-Pages* und *Internet Database Connector (IDC)*, *Vignette Corporation StoryServer* und *Informix AppPages*. Insbesondere für Datenbankhersteller bietet sich der Ansatz der Datenbankprogrammierschnittstellen-Erweiterungen an. Hier können exemplarisch *Oracle PL/SQL Web Toolkit* und die *Sybase PowerBuilder Web* Klassenbibliothek genannt werden. Analog zu den beiden zuvor genannten Kategorien befinden sich die hier beschriebenen Anwendungen in weitverbreitetem Einsatz. In vielen Fällen sind ihre Funktionalitäten jedoch von der Produktseite betrachtet mittlerweile in die entsprechenden Enterprise-Lösungen eingeflossen. In Komponenten von Application-Servern (vgl. Seite 180) integriert, wird über diese die hier beschriebene Funktionalität unter anderem als Teil der Gesamt-Funktionalität mitbereitgestellt.

7.1.4 WWW-Formular-Editoren, Groupware, Berichterstellungswerkzeuge und Datenbank-Publizierungsassistenten

Diese Kategorie fasst die datenbankgebundenen Webapplikationen, die sich im Unterschied zu der vorherigen Kategorie durch eine deutlich höhere Ebene, d.h. mehr Abstraktion bei weniger Programmieraufwand im Umgang mit den Datenstrukturen auszeichnen, zusammen. Basierend auf den Datenmodellen der Datenbanken ermöglichen die dieser Kategorie zuzu-

ordnenden Werkzeuge den Aufbau von Web-Formularen und Übersichten (Reports) durch die Konfiguration von vorgefertigten Komponenten. Datenbankverbindungskomponenten ermitteln aus der Datenbank das zugrundeliegende Datenmodell und erlauben anhand dieser Metadaten die Erstellung von Master-Detail-Ansichten, generieren Verknüpfungen anhand von Integritätsbeziehungen und können dementsprechend einfach Eingabemasken erstellen, mit denen auf die Datenbank über das Web zugegriffen werden kann. Aufgrund der Komponentenorientierung ist die Architektur dieser Anwendungen von den Entwicklern frei konfigurierbar. Sowohl Two- als auch Three-Tier- Architekturen kommen zum Einsatz. Eine Besonderheit stellt die Möglichkeit dar, den in den Komponenten gekapselten Code auch auf der Client-Seite auszuführen. Dies kann über proprietäre Anwendungen geschehen, die in Plug-ins vom Web-Browser ausgeführt werden, bereits im Web-Browser implementierte Modelle wie ActiveX [Aar97] oder Java Laufzeitumgebungen. Über „remote“ zugreifbare Objekte und Methoden mittels CORBA [Pop98], RMI [Far98] oder DCOM [RBR98] lassen sich problemlos verteilte Architekturen, wie dies beispielsweise auch bei JDaphne genutzt wird, aufbauen.

Charakteristisch für die in dieser Kategorie zusammengefassten Ansätze und Werkzeuge ist ihre aus der Sicht der Web-Entwicklung große Ähnlichkeit zu IDEs (Integrated Development Environments) und Rapid Application Development (RAD) Tools. Sie decken den Webapplikations-Entwicklungszyklus vollständig ab und steigern durch ihre visuellen Unterstützungen beim Prozessmanagement, dem Debugging und einem Konfigurationsmanagement die Produktivität stark. Sie unterstützen in der Regel keine konzeptionelle Modellierung und sind bei Modifikationen der Datenstrukturen nur schwer anzupassen. Anders als die in der vorherigen Kategorie besprochenen Werkzeuge liefern die Formular-Generatoren und Datenbankschnittstellengeneratoren in den meisten Fällen ein festes, vorgegebenes Design inklusive einer Benutzerschnittstellenlogik, die sich oft nur schwer modifizieren lässt. Webapplikationen lassen sich mit diesen Werkzeugen zwar schneller, aber bei weitem nicht so frei und flexibel aufbauen, wie dies mit den programmierbaren Datenbankschnittstellen-Integrationslösungen der Fall ist. Bezüglich der Erstellung und dem Editierprozess von Informationen gilt für die hier aufgeführten Anwendungen Analoges wie für die zuvor besprochene Kategorie. Anders als in der Konzeption von JDaphne/DAPHNE, die der verteilten Erstellung und Bearbeitung von Dokumenten mit anschliessender Qualitätssicherung basierend auf Workflows einen großen Raum gibt, steht bei den Anwendungen dieser Kategorie der Publikationsvorgang von strukturierten, in Datenbanken abgelegten Informationen im Vordergrund. Diese lassen sich durch die mit den Assistenten generierten Formulare modifizieren, bieten aber für Workflows wenig Raum.

Kommerzielle Produkte, die zu der Formular-Editor-, Report- und Datenbank-Publizierungsassistenten-Kategorie zählen, sind *Microsoft's Access* bzw. *Visual InterDev* und *Visual Basic*, *Oracle's Reports for the Web* und *Developer2000*, *Inprise's IntraBuilder*, *Sybase's Powerbuilder*, *NetDynamics' Visual Studio* und *Allaire Cold Fusion Application Wizards*. Diese Produkte befinden sich im Unternehmen noch weitreichend im Einsatz. Ihre Funktionalitäten, insbesondere die Assistenten sind jedoch aktuell in großem Maße mit den in der vorherigen Kategorie beschriebenen Anwendungen verschmolzen und ein essentieller Bestandteil der Application-Server geworden.

7.1.5 Anwendungen mit Kombination verschiedener Techniken

Bereits im Vorangegangenen wurde angedeutet, dass die aktuelle Entwicklung auf eine Integration der bislang beschriebenen Konzepte zum Generieren von Web-Präsenzen, insbesondere Webapplikationen, hinausläuft, die in den Application-Servern ihren momentanen

Höhepunkt auch hinsichtlich der Komplexität der darunter liegenden Mechanismen und Methoden findet. Als Zwischenstufe auf dem Weg zu diesen omnipotenten Systemen fasst die hier beschriebene Kategorie eine noch mit geringerer Integrationstiefe arbeitende Produktpalette zusammen. Die hier aufgelisteten Programme zeichnen sich dadurch aus, dass sie die Konzepte und Mechanismen der Systeme aus den bisher erläuterten Kategorien in der einen oder anderen Weise vollständig oder teilweise kombinieren und integrieren. Eine typische Konfiguration, die in dieser Kategorie zu finden ist, besteht aus einer Komponente zum visuellen Editieren von HTML-Dateien sowie einer Web-Präsenz-Management Lösung, die durch weitere Komponenten mit denen ein Datenbankanschluss realisiert wird, ergänzt werden. Dieser kann durchaus in vielfältigen Formen dargeboten werden, z.B kann direkt nach HTML generierend oder über Assistenten mit clientseitiger Funktionalität eine datenbankenzugreifende Oberfläche in Form einer Webapplikation erstellt werden.

Auch wenn bei JDaphne/DAPHNE der Produktcharakter nicht gegeben ist, lassen sich die beiden bezüglich ihrer Architektur und der kombinierenden Konzeption am trefflichsten mit den hier beschriebenen Produkten vergleichen. Diese kombinierenden Anwendungen unterstützen effektiv den Aufbau von Web-Präsenzen mit dynamischem und statischem Content, ohne grundsätzlich neue Konzepte zu benötigen. Die Leistung besteht in der Verbindung der vorhandenen Anwendungen und der Bereitstellung und Optimierung der zwischen ihnen benötigten Schnittstellen. Die anzutreffenden Systemarchitekturen decken das gesamte Spektrum der bisher beschriebenen Werkzeuge ab, in den meisten Fällen werden Three- oder Multi-Tier-Architekturen eingesetzt. Vom Lebenszyklus einer Web-Präsenz deckt diese Kategorie das Design, beschränkt auf hierarchische Layoutdefinitionen, die Implementation und Verwaltung ab. Über Templates und Präsentationsstile lassen sich konsistente Layouts inhaltsunabhängig für die gesamte Webapplikation erstellen. Kommerzielle Programme, die zu dieser Kategorie gehören, sind Produkte wie *Microsoft Frontpage 1998/2000*, *Lotus Domino* und *Domino Designer* basierend auf dem Domino-Server, *NetObject's Fusion (Version 3.0)* mit *Fusion2Fusion* (verbindet *NetObject's* visuellen HTML-Editor mit *Allaire's HTML-SQL Integrator*), sowie frühere Versionen von *Apple's Web Objects*.

7.1.6 Middleware/Application-Server

Weitreichende Fähigkeiten zur Systemintegration innerhalb eines Unternehmens und der Erstellung hochfunktionaler Webapplikationen bieten zur Zeit die Application-Server. Sie sind die Middleware (vgl. S. 49) zwischen Datenbank und Webserver und dienen der integrierenden Erstellung von Webapplikationen. Application-Server enthalten typischerweise eine IDE und eine Reihe von bereits vorgefertigten Modulen für die Datenbankabfrage und die Datenaufbereitung. Sie stellen definierte Laufzeitumgebungen für komponentenbasierte Webapplikationen dar. In vielen Fällen auf der Java-Enterprise-Architektur [CIP01] aufsetzend, bieten sie eine umfassende Basis für die beispielsweise in Java-Beans [MH00] gekapselte Implementation von Applikationen, insbesondere von Informations-Systemen und transaktionsverarbeitenden Anwendungen mit Web-Schnittstelle.

In Kombination mit den herstellerspezifischen Datenbank-Management-Systemen wie im Fall von *IBM* mit *WebSphere* und *DB2*, *Sybase EAServer* mit dem *SQL-Server* und schließlich der als Internet-Datenbank titulierten *Oracle* Lösung *8i* bzw. *9i*, entfalten die Application-Server ihren vollen Funktionsumfang. Aber auch die Anbindung anderer Datenquellen ist bei ihnen unproblematisch und schlägt sich lediglich in Einschränkungen beim Entwicklungskomfort nieder.

Bereits mit wenigen Schritten lassen sich über Application-Server entweder durch manuelle Code-Generierung oder durch Konfiguration und Erweiterung vordefinierter Komponenten

umfangreiche Webapplikationen erstellen. Gerade für den Einsatz in kommerziellen Umgebungen macht ihre hohe Skalierbarkeit und Performance sie wegen der großen Flexibilität für die Datenintegration sehr interessant. Auf Middleware-Ebene erlauben sie nativ die Verteilung von Funktionalitäten mittels CORBA und RMI, auf Microsoft-Seite mittels DCOM.

Weiterhin kann ein Application-Server dazu genutzt werden, diverse Applikationen gleichzeitig zu betreiben. Versuche, Application-Server selbst über den Einsatz von Java plattformunabhängig zu machen, sind aufgrund des hohen Performance-Bedarfs bislang gescheitert. Im Gegenteil dazu zielt beispielsweise die Entwicklung bei Oracle darauf ab, ein eigenständiges minimales Betriebssystem dem eigenen Application-Server als Basis mitzugeben.

Anders als bei den modellbasierten Webapplikationsgeneratoren und den bislang beschriebenen Anwendungen steht bei den Application-Servern der Applikationsgedanke mehr auf der Seite der Business-Logik, die in der Server-Komponente (Java-Enterprise-Bean, bzw. Servlet) entwickelt werden muss als auf der konzeptionellen Erstellung von Navigationsstrukturen und Präsentationsvorgaben. Diese werden mittels vielfältiger Konzepte in den IDEs entworfen.

Zurückblickend auf die Einleitung dieses Kapitels steht bei den mit Application-Servern generierten Web-Präsenzen bzw. -applikationen weniger der hypermediale Aspekt als vielmehr die informationsverarbeitenden, transaktionellen Belange im Vordergrund. Wie Entwicklungen im Umfeld von JDaphne [QEM00a, QEM00b] zeigen, lassen sich die hier beschriebenen Application-Server insbesondere gut als Entwicklungsbasis und Laufzeitumgebung für Web-Content-Management-Systeme einsetzen. Sie bieten alle notwendigen Schnittstellen wie die für die Web-Browser, die für Datenbanken und andere verteilt arbeitende Komponenten. Im Spezialfall von JDaphne/DAPHNE wurde gezielt auf einen Application-Server als Basis verzichtet, um ein vergleichsweise schlankes System, das nicht auf die für den Application-Server anfallenden Lizenzmodelle angewiesen ist, bereitstellen zu können. Für die vollständig in Java implementierte Geschäftslogik von JDaphne besteht insbesondere mit den mittels RMI zugreifbaren Komponenten bei Bedarf jederzeit die Möglichkeit einer Einbettung der gesamten Anwendung in einen Application-Server als Laufzeitumgebung.

7.1.7 Dokument- und Content-Management-Systeme mit WWW-Export-Schnittstellen

Einen umfassenderen, mehr inhaltszentrierten Ansatz verfolgen die Produkte aus der Dokument- und Content-Management-System-Kategorie. Bei den Dokument-Management-Systemen handelt es sich von ihrer ursprünglichen Konzeption um Systeme, die das Publizieren, „Sharen“ und Lesen von Dokumenten im Intranet-Bereich ermöglichen. Beim Content-Management spielen neben dem Dokument-Management zunehmend elektronische Geschäftsprozesse eine Rolle. Dazu gehören beispielsweise auch E-Commerce und Customer-Relationship-Management (CRM). Content-Management-Systeme unterstützen diese Vorgänge unter Berücksichtigung der zugrundeliegenden Lebenszyklen. Sie zielen unabhängig von dem letztendlichen Publikationsmedium auf die Inhalte ab und stellen eine Vorstufe zum Wissens-Management dar. Ihre Hauptfunktion liegt in der Zusammenführung und Verwaltung von Inhalten, die von verschiedenen Plattformen und Applikationen stammen. Durch eine einheitliche personalisierbare Benutzerschnittstelle ermöglichen sie den Zugriff auf verschiedenste Inhalte. Über Weiterverarbeitungs-, Integrations- und Publikationsmechanismen werden die Inhalte, die intern medienneutral verwaltet werden, für die Präsentation in vielen verschiedenen Medien aufbereitet. Neben der Publikation in Printmedien, dem ursprünglichen Zielmedium, haben außer Offline-Multimedia-Publikationen insbesondere die Online-Medien in Form von Web-Präsenzen und Internet-Portalen an Bedeutung gewonnen.

Während die ersten Schritte vom Dokument- und Content-Management bei der Publikati-

on von Web-Präsenzen noch zahlreiche Probleme mit sich brachten, wie dies am Beispiel von *Documentum* in einer früheren Version in einem Aufsatz von Balasubramanian und Bashian [BB98] beschrieben wird, können mit den aktuellen Dokumenten- und Content-Management-Systemen hochwertige Web-Präsenzen erstellt und verwaltet werden. Durch die vollständige Trennung von Inhalten und Layout lassen sich mit Content-Management-Systemen hohe Abstraktionsniveaus bezüglich der die Web-Präsenz aufbauenden Struktur erreichen. Sie kann unabhängig von den Inhalten entworfen und modifiziert werden und wird erst beim Publikationsvorgang ihnen gefüllt. Zusammen mit den auf das Content-Management abgestimmten Workflows zum Erstellen und Modifizieren von Inhalten, die speziell für die Nutzung im WWW-Umfeld erweitert und ergänzt wurden, kann so der Lebenszyklus einer Web-Präsenz weitreichend abgedeckt werden.

Die Architektur der Systeme aus dieser Kategorie liegt naturgemäß im Multi-Tier-Bereich. Als Client-Server Anwendungen, die über verschiedene Schnittstellen eigene Back-Ends, z.B. das Datenbank-basierte Content-Repository, oder über Middleware-Komponenten unternehmensweite Informationsressourcen integrieren, stellen sie eine vielschichtige und komplexe Lösung dar. Ähnlich wie die Application-Server, deren Konzeption in der vorherigen Kategorie erläutert wurde, ist ihre Komplexität und Funktionalität hoch. Durch ihre vereinheitlichende Ausrichtung und den Anspruch, alle im Unternehmen anfallenden Inhalte zu verwalten, geht ihre Funktionalität weit über die allein für den Betrieb einer Web-Präsenz benötigte hinaus. Andererseits kann aufgrund des generalisierenden Ansatzes dieser Produkte nur mit großen Anstrengungen der speziell für eine Web-Präsenz aufbereitete Inhalt in das Gesamtsystem integriert werden. Zu den Produkten, die der Kategorie der umfassenden Dokument- und Content-Management-Systeme zuzuordnen sind, zählen beispielsweise die *Documentum 4i eBusiness Edition Platform* und der *Content Manager VIP 5e* von *Gauss*.

7.1.8 Web-Content-Management-Systeme

Auf dem Markt werden immer mehr Produkte unter dem Label „Web-Content-Management-Systeme“ (WCMS) angeboten – eine Blickweise, die wie die Produktübersicht dieses Kapitels zeigt, durchaus ihre Berechtigung hat, da sie alle auf die eine oder andere Weise für das Management einer Web-Präsenz eingesetzt werden können. In die hier beschriebene Kategorie der Web-Content-Management-Systeme fallen insbesondere alle diejenigen Systeme, die speziell zum Verwalten von Web-Contents, dies sind Informationsobjekte wie beispielsweise Kataloge, Hyperlinks, GIF-, JPG- oder PNG-Dateien sowie eingelagerte Prozeduren (Skripte), konzipiert wurden.

Während bei den Content-Management-Systemen der vereinheitlichende Ansatz und die neutrale Verwaltung von Inhalten das Fundament des Konzeptes bildet und die Erstellung einer Web-Präsenz aus diesen Inhalten lediglich eine mögliche Nutzung des Systems ist, verfolgen die Systeme der Web-Content-Management-Kategorie einen spezifischeren Ansatz, der vom erwünschten Resultat der damit betriebenen Web-Präsenz ausgeht. Sie bieten die Möglichkeit, ohne Programmierkenntnisse Intranet- und Internet-Seiten anzulegen und zu verwalten. Web-Content-Management-Systeme ermöglichen es, Informationen dezentral und arbeitsteilig über das Web dort einzugeben, wo sie erarbeitet wurden. Dabei steuert eine zentrale Stelle die Verwaltung und Verteilung der strukturierten Inhalte. Unterstützt wird insbesondere der Lebenszyklus einer Web-Präsenz bzw. der dafür benötigten Inhalte von der Erstellung der Web-Seiten bis zu ihrer Veröffentlichung und Archivierung.

Das Abstraktionsniveau der mit solchen Systemen realisierbaren Web-Präsenzen/Webapplikationen ist im Normalfall nicht so hoch wie das der reinen Content-Management-Systeme, da hier die Inhalte anhand von konkreten Vorgaben für die Web-Präsenz aneinander ausge-

richtet werden. Durch die medienneutrale, navigationsstrukturunabhängige Verwaltung der Inhalte lassen sich jedoch geänderte Strukturen in der Web-Präsenz mit nur geringem Aufwand aufbauen.

Entsprechend ihrer Konzeption für das Web auch als webbasierte Systeme implementiert, sind bei den WCMS immer mindestens Three-Tier-, in einigen Fällen auch Multi-Tier-Architekturen realisiert. Plattformunabhängigkeit wird in jedem Fall auf der Client-Seite angestrebt, obwohl auch hier proprietäre Lösungen genutzt werden, um den Benutzerkomfort zu steigern. In vielen Fällen erlauben die als Web-Seiten realisierten, mit Eingabemasken vergleichbaren Benutzerschnittstellen vollständig webbasiertes Arbeiten. Auf der Server-Seite arbeiten, anders als JDaphne, viele Systeme noch nicht plattformunabhängig. Gründe dafür sind neben Performance-Fragen die unter den Systemen liegenden plattformabhängigen Funktionalitäten, z.B. die Einbindung in eine Windows-Benutzerverwaltung etc. Aber auch das Aufsetzen auf andere Lösungen, z.B. den *Coldfusion-Server*, stellt durchaus eine Einschränkung dar. Eine Produktübersicht in dieser Kategorie zu liefern stellt sich aufgrund des oben erwähnten Trends zu dem Label WCMS als eine schwierige Aufgabe heraus. Produkte, die am passendsten dieser Kategorie zugeordnet werden können, sind exemplarisch *Infosite* der Fa. Sitepark und *inSite* der Fa. *SICOM*. Stärker aus dem Umfeld des hypermedialen Dokument-Managements stammt das schon fast der Kategorie konzeptbasierter Werkzeuge zuzuordnende Produkt *Hyperwave*. Viele der in der Kategorie der visuellen Editoren und Web-Präsenz-Manager auf Seite 175 aus Vollständigkeitsgründen aufgeführten Systeme sind, sofern ihre Produktion nicht eingestellt wurde, im Laufe der Zeit zu Systemen herangereift, die der Kategorie der WCMS zugeordnet werden können.

7.1.9 Online-Redaktionssysteme

Online-Redaktionssysteme können als eine Unterkategorie der WCMS betrachtet werden. Sie werden von ihrer originären Herkunft zur Erstellung eines speziellen Typs von Web-Präsenzen, den Online-Zeitungen, verwendet. Ihre Stärke liegt in der Erstellung sich ständig ändernder Internetauftritte. Viele der Redaktionssysteme stammen von den ursprünglichen Druck-Redaktionssystemen ab. Als die Verlage und damit auch die Programmzulieferer festgestellt haben, dass das Printmedium durch das Internet ergänzt werden muss, haben sie sich darauf ausgerichtet, die bereits im Printmedium angebotenen Informationen zusätzlich auch online bereitzustellen. Die Redaktionssysteme für Printmedien haben eine proprietäre Entwicklung von mehreren Jahrzehnten hinter sich. Die Technik in den Verlagshäusern war bis dato auf das Printprodukt ausgerichtet, sodass die Entwicklung von Redaktionssystemen, die Printmedium und Online-Publikation gleichermaßen beherrschen, noch nicht abgeschlossen ist. Insbesondere der Weg von Inhalten aus dem Online-Medium in das Printmedium wird erst langsam als gangbar erkannt.

Mit DAPHNE entstammt die Konzeption von JDaphne einem solchen System, das auf den Import, die Weiterverarbeitung und Ergänzung von für das Printmedium erstellten Dokumenten ausgerichtet ist (vgl. Seite 74). Redaktionssysteme sind speziell auf die Belange von Redakteuren zugeschnitten, sie bieten eine einfache, effiziente Bedienung und entziehen dem Redakteur die Kontrolle über das Layout. Ihr Schwerpunkt liegt in dem Umgang mit und der Verwaltung von ständig aktuellen Texten und Bildern. Aktuelle Redaktionssysteme unterstützen den Lebenszyklus einer Web-Präsenz sehr weitreichend. Mit der starken Tendenz zu dynamischen Web-Präsenzen, die am besten dem ständigen Informationsfluss bei den Online-Zeitungen gerecht werden, hat sich bei den Redaktionssystemen ein hohes Abstraktionsniveau beim Entwurf der Web-Präsenz durchgesetzt. Eine an die Struktur von Printmedien angelehnte Aufteilung der Web-Präsenz in Ressorts hilft bei ihrer Definition unabhängig

von den publizierbaren Inhalten. Die Erstellung bzw. der Import von Inhalten ist eine der Standardaufgaben eines solchen Systems. Wie bereits erwähnt, spielen proprietäre Lösungen auf dem Redaktionssystem-Markt aufgrund der engen Kopplung an das Printmedium bislang noch eine große Rolle. Den Vorteil der Internet-Technik erkennend werden aktuelle Systeme in Multi-Tier-Architekturen aufgebaut und verfügen über Web-Benutzerschnittstellen. Exemplarisch als zu dieser Kategorie zählend genannt werden können die Produkte *DocMe 3.0* der Fa. *Arago*, *InterRed 3.5* der Fa. *InterRed* und die *NetObjects Authoring Server Suite* der Fa. *NetObjects*.

7.2 Im wissenschaftlichen Umfeld diskutierte Ansätze und Konzepte

Aufgrund der umfassenden Funktionalitäten, die von einem System zum Management einer Web-Präsenz gefordert werden, ist für die im folgenden in Kategorien eingeordneten Systeme schwierig, den vollen Funktionsumfang abzubilden. Nur einige wenige Ausnahmen, wie das *HyperG* Projekt, aus dem letztendlich das Produkt *Hyperwave* hervorgegangen ist, stellen eine umfassende Web-Präsenz-Verwaltung zur Verfügung. Aus den zahlreichen für die Verwaltung einer Web-Präsenz notwendigen Funktionalitäten werden in diesem Abschnitt zwei verschiedene Kategorien herausgegriffen, die eng mit dem Web-Präsenz-Management im Sinne von JDaphne verbunden sind. Im folgenden werden in Systemen umgesetzte Konzepte aus den folgenden Bereichen betrachtet:

- Multiautoren-Systeme
- Modellbasierte Web-Präsenzen – dynamisches Publizieren strukturierter Informationen

Hierbei beleuchtet die Kategorie der Multiautoren-Systeme die kooperative Erstellung und Manipulation von hypermedialen Dokumenten. Je nach Ausprägung des Systems steht gegebenenfalls lediglich der gemeinsame Zugriff auf die Dokumente oder aber deren Publikation im Vordergrund. Dem allgemeinen Trend in Richtung Web-Engineering folgend werden in der Kategorie der modellbasierten Web-Präsenz-Generatoren Ansätze und Systeme beschrieben, die darauf abzielen, mit den Methoden und Mitteln der Software-Entwicklung – gemäß der Vorstellung einer Web-Präsenz als Webapplikation – eine solche in abstrakter Form zu definieren. Anhand dieser Beschreibung, einem Modell, wirken die eigentlichen Systemkomponenten, die Generatoren, die anhand der in den Modellen definierten Abbildungsvorschriften aus einem vorhandenen Datenbestand die Web-Präsenz aufbauen. Die in dieser Kategorie zusammengefassten Systeme vernachlässigen dabei jedoch die Erstellung und Modifikation der Inhalte. Über die Kombination der beiden Kategorien Multiautoren-Systeme und modellbasierte Web-Präsenz-Generatoren ergibt sich somit bei Einhaltung entsprechender medienneutraler Datenformate durchaus die Möglichkeit der Kombination zu einem vollwertigen Web-Präsenz-Management-System.

7.2.1 Multiautoren-Systeme

Schon lange bevor das Konzept des WWW vor mehr als zehn Jahren am CERN geboren wurde [BLCGP92] sind Hypertexte – später auch Hypermedia – als ein bei der Nutzung von Computern geeignetes Medium zur Informationsrepräsentation entdeckt worden. Seit dieser Zeit entwickelte, in vielen Fällen proprietäre (monolithische, z.B. *Notecards* [HMT87]) und erst später auch offene (vgl. [Mey89]) Systeme [Wii99] konnten sich trotz in vielen Fällen

ausgereifterer Technik (z.B. bidirektionale Verknüpfungen bei *HyperG* [MKPS92]) nicht gegen die offene Architektur des WWW behaupten.

Die Ursache dafür liegt zu einem gewissen Teil darin, dass diese Systeme mit ihren umfassenden, z.T. objektorientierten Modellierungen der Hypermedien zu komplex im Vergleich zum WWW, dessen Formate und Protokolle sich zumindest zu Beginn durch Einfachheit auszeichneten, sind. Ein weiterer Grund mag darin liegen, dass die offenen Hypermedia-Systeme stark auf die computerunterstützte kooperative Arbeit (Computer Supported Cooperative Work, CSCW)[Rod91] fokussiert waren. Ihr primäres Ziel war die kooperative Erstellung und Bearbeitung von gemeinsamen Hypertexten [IT89, SHT89, Wii91]. Die dazu entwickelten Systeme haben Autoren-System-Charakter [GLR95], erlauben somit die verteilte Arbeit an einem stark vernetzten multimedialen Gesamtdokument, sehen aber nativ keine Publikationsstufe vor, die die von der Autoren-Umgebung unabhängige Nutzung des Hypertextes (vgl. [KRS00]) in einem gemeinsamen Format über ein gemeinsames Protokoll ermöglicht, wie dies mit dem WWW initial der Fall war. Während aktuell alle Bemühungen des Web-Content-Managements dahin gehen, die Inhalte in möglichst neutraler Form (z.B. in XML) und in Unabhängigkeit von der Struktur des Gesamtdokumentes zu erfassen, um sie bei der Publikation entsprechend aufzubereiten, war in den Multiautorenumgebungen der damaligen Hypertextsysteme der konträre Ansatz verbreitet. Wie z.B. im Aufsatz von Streitz et al. [SHH+92] beschrieben, stand die Sicht des Benutzers auf den fertigen Hypertext im Vordergrund, dieser wurde von den Autoren in gemeinsamer Arbeit als großes Gesamtdokument aufgebaut.

Heute hingegen wird mit der Web-Präsenz über Kategorien und Dokumenttypen eine inhaltliche Struktur geschaffen, die von den Autoren „nur noch“ mit Leben gefüllt werden muss. Die Arbeit des Zusammenbaus und der Verknüpfung der einzelnen, durchaus separaten inhaltlichen Komponenten erfolgt über Export-Mechanismen, die eine vorgegebene Struktur beispielsweise in Form einer Web-Präsenz generieren.

Für die kooperative Erstellung und Bearbeitung von gemeinsamen Hypertexten werden Autorenumgebungen benötigt, die gemeinschaftliches Arbeiten an dem Bestand der Hypertext-Komponenten ermöglichen. Ein wichtiges Element bei dieser Autorenarbeit ist die Bereitstellung einer gemeinsamen Arbeitsoberfläche. Alle Benutzer können gleichzeitig das Gesamtdokument bearbeiten. Wichtiger Bestandteil solcher Systeme ist dementsprechend wie bei *DOLPHIN* [SGHH94] eine Kommunikationskomponente. Über die Arbeit in der Gruppe, in Teams, die sich virtuell treffen – visualisiert über die gemeinsame Oberfläche – entstehen Elemente des Hypertextes, der dem Gesamtdokument hinzugefügt wird.

Mit *HyperG* [MKPS92] ist in Graz ein Hypertext-System entstanden, das aufgrund seiner Funktionalitäten im Bereich der Hyperlink-Konsistenz und der Suche von Dokumenten eine Alternative zum WWW darstellt. Trotz der Verwendung eigener Dokumentenformate und Protokolle in den ersten Versionen konnte im Nachhinein mit dem Hyperwave-Server die Brücke zum WWW geschlagen werden. Auf konzeptioneller Ebene baut *HyperG* auf einer hierarchischen Struktur aus Verzeichnissen (Collections) auf. Weiterhin eingeführte „Cluster“ verbinden zusammengehörige Dokumente zu einer Gruppe, z.B. bei mehrsprachigen Dokumenten oder verschiedenen Formaten eines Dokumentes. Aus Kompatibilitätsgründen wurde der native HyperG-Browser (Harmony) durch entsprechende HTML-Exporte des Hyperwave-Servers dem allgemeinen Trend folgend durch Web-Browser ergänzt [And97]. Auch bei anderen Systemen, z.B. dem Konstanzer Hypertext-System (KHS) [Ham96], steht die verteilte Autorenarbeit im Vordergrund; Anpassungen in Richtung WWW wurden erst mit dessen starker Verbreitung geschaffen. Andere Systeme bei denen ebenfalls die gemeinsame Erstellung von Inhalten im Vordergrund steht und die Publikationskomponente fehlt, sind die gemeinsamen virtuellen Arbeitsplätze (Basic Support for Cooperative Work, BSCW) [BAB+97]

oder der dokumentenzentriertere Ansatz wie bei Tietze et al [TBR98]. Das HTTP erweiterndes Protokoll *Web-DAV* [WG99] betrifft wiederum die Autorenkomponente eines Web-Präsenz-Management-Systems und erfordert zusätzliche Komponenten, die neben der Konsistenzhaltung und Qualitätssicherung den eigentlichen Publizierungs- und Generierungsprozess beinhalten müssen. Versuche, diesen allgemein akzeptierten Standard zu nutzen, sind unter Berücksichtigung entsprechender Infrastruktur, wie sie der *WebSphere* Application-Server bietet, erfolgreich verlaufen [QEM00a, QEM00b]. Für JDaphne wird diese Funktionalität, ausgerichtet an den Randbedingungen wie sie durch die Projektpartner vorgegeben sind, bei Bedarf übernommen oder nachempfunden.

7.2.2 Modellbasierte Web-Präsenzen – dynamisches Publizieren strukturierter Informationen

Dem Trend zum Web-Engineering (vgl. [Pow98]) folgend, befassen sich viele aktuelle Ansätze mit modellbasierten Verfahren zum Generieren von datenintensiven transaktionellen Web-Präsenzen unter Ausnutzung der aus der Software-Entwicklung bekannten Ansätze. Um Applikationen zu entwickeln, die Hypermedia-Navigation mit komplexen transaktionellen Verhalten kombinieren, wird ein systematischer Entwicklungsansatz benötigt [RSL99]. Grundlegender Einstieg in die Entwicklung einer datenintensiven transaktionellen Web-Präsenz ist ihre konzeptionelle Modellierung unter Nutzung geeigneter Modellierungswerkzeuge und -sprachen, wie z.B. UML [RJB99] und Design-Pattern [GHJV94]. Die Web-Präsenz wird als Darstellung (View) des konzeptionellen Modells implementiert. Für die Entwicklung einer Webapplikation z.B. mit einer objektorientierten Designmethode wie OOHDM (Object Oriented Hypermedia Design Method, [DGB96]) werden mindestens vier Schritte benötigt:

1. konzeptionelle Modellierung
2. Navigations-Design
3. Interface-Definition
4. (automatisierte) Implementation

Mit *WebFlow* [BBC+97] arbeitende Modellierungen basieren (wie z.B. [PS00]) auf einem dreistufigen Vorgehen: dem Business-Prozess, dem Workflow und schließlich dem WebFlow. Jede Stufe jeweils in strukturelle und Verhaltenseigenschaften unterteilend, wird der Entwicklungsprozess auf diese Weise abgedeckt, wobei sowohl die (Daten-)Fluss-Dimension als auch die Objekt-Dimension der Applikation berücksichtigt werden. Auch andere Ansätze wie z.B. WebML (Web Modeling Language) [CFB00] versuchen umfassende Notationen speziell für komplexe Webapplikationen auf der konzeptionellen Ebene zu entwerfen. Dementsprechend steht auch bei WebML die abstrakte Beschreibung der verschiedenen orthogonalen Dimensionen der Webapplikation – Struktur-, Kompositions-, Navigations-, Präsentations- und Personalisierungs-Modell – im Vordergrund. Die auf abstrakter Ebene verifizierten Modelle werden schließlich für den Aufbau von Interfaces und Objekten herangezogen. Diese dienen wiederum der automatisierten Implementation der Webapplikation. Dazu interpretiert/compiliert ein Interpreter/Compiler, am besten als Generator bezeichnet, die Templates oder objektorientierten Interface-Definitionen und baut dynamische Komponenten zusammen, die in eine Laufzeitumgebung auf dem Web-Server eingebettet die Repräsentation der Webapplikation vornehmen können. Ein Beispiele für eine solche Vorgehensweise ist das *Strudel*-System [FFK+97], bei dem Abbildungsvorschriften für strukturierte Daten analog zu SQL-Datenbankanfragen mit abstrakten Navigationselementen verknüpft werden und über einen Generator die fertige Web-Präsenz erzeugt wird.

Mit *JESSICA* [RS98] ist ein mehr objektorientierter Ansatz gewählt worden. Hier beschreibt eine objektorientierte Sprache die Komponenten der Webapplikation mit den damit verbundenen Vorteilen der Wiederverwendbarkeit und der durch die weniger detailliertere Sicht einfacheren Verwaltung. Auf diese Objekte kann zu Wartungszwecken über den vollständigen Lebenszyklus zugegriffen werden. Jede Modifikation abschließend bildet ein Compiler die abstrakten Dienstbeschreibungen in ein dateibasiertes Repository für einen beliebigen Web-Server ab.

Auch mit dem *Araneus*-Projekt [MAM+98], *Autoweb* [FP98] und dem Database-Publishing wie bei Loser [Loe00] oder Kirda und Kerer [KK00] wurden analoge Vorgehensweisen für die Erzeugung einer Webapplikation entwickelt. Mit den hier aufgeführten Ansätzen steht ein signifikant anderer Aspekt im Vordergrund als bei den oben beschriebenen Verfahren zum verteilten gemeinsamen Erstellen von Hypertexten. Hier steht eindeutig der Aspekt Web-Präsenz als Informationssystem und mehr noch als (transaktionelle) Applikation im Vordergrund; während für die Systeme aus der vorherigen Kategorie die gemeinsame Erstellung und Modifikation von Inhalten das primäre Ziel und eine Web-Präsenz lediglich eine Darstellung der Inhalte ist, finden die in diesem Abschnitt erläuterten Konzepte ihre Anwendung bei der Publikation ebensolcher, bereits in strukturierter Form vorliegender Inhalte, klammern deren Erstellung aber weitgehend aus.

Für das Web-Präsenz-Management ergibt sich die grundlegende Frage, in welche Richtung die Web-Präsenz des Unternehmens tendieren soll. Während momentan noch das Attribut „datenintensiv“ genutzt wird, um zwischen Webapplikationen und Web-Präsenzen im herkömmlichen Sinne, d.h. als großes Hypertext- bzw. Hypermedia-Dokument unterscheiden, ist für die Zukunft hier eine Annäherung zu erwarten. Über den Schritt des Content-Managements mit dem Vorhalten medienneutraler strukturierter Daten wird in absehbarer Zeit die noch bestehende Lücke zwischen den beiden Vorgehensweisen geschlossen werden können.

7.3 Einordnung von JDaphne

Nach dem exemplarischen Überblick über Systeme und Konzepte, die derzeit zum Web-Präsenz-Management eingesetzt werden, ist die Einordnung von JDaphne zu einer der hier dargestellten Kategorien nicht eindeutig. Dem allgemeinen Trend folgend auch bei JDaphne von einem Web-Content-Management-System zu sprechen, entspricht nur einer oberflächlichen Einordnung. Mit DAPHNE aus seinen Wurzeln heraus eindeutig ein Online-Redaktionssystem, wie es für die Erstellung einer Online-Zeitung eingesetzt wird, lässt sich JDaphne durch die speziell auf sich ändernde Inhalte ausgerichteten Mechanismen in Teilbereichen auch in andere der oben beschriebenen Kategorien einordnen. Einerseits bietet JDaphne mit seinen Web-Präsenz-Management-Mechanismen durchaus Fähigkeiten von Systemen, die unter dem einfachen Web-Präsenz-Management mit den Editor-Lösungen eingeordnet sind; andererseits sind durch die mit JDaphne realisierten Workflows Funktionalitäten hinzugekommen, die auch von Dokument-Management-Systemen geboten werden. Von diesen unterscheidet sich JDaphne bereits in der Art und Weise, wie Inhalte verwaltet werden: während JDaphne sich hier auf die im Internet gebräuchlichen Formate beschränkt und dafür den Mangel an Kenntnis über die inhaltliche Struktur der verwalteten Dateien in Kauf nimmt, sind die Dokument-Management-Systeme und noch mehr die Content-Management-Systeme darauf ausgelegt, die in den Dateien abgelegten Inhalte in ihrer Struktur zu erfassen. Dafür setzen sie verstärkt XML als flexible Datenbeschreibungssprache ein. Der Vorteil der gemeinsamen Datenbeschreibungssprache wird dabei von ihnen durch speziell auf die in Dokument-Typ-Definitionen (DTD) oder Schemata abgestimmte Handler- und Enabler- Module erkauft.

Die in JDaphne eingesetzten Templates und die erst beim Export den Dokumenten hinzugefügten Navigationselemente mitsamt Layout ermöglichen durchaus die Realisierung einer Abstraktionsebene zwischen dem eigentlichen Inhalt und der letztendlichen Repräsentation auf der Web-Präsenz. Offensichtlich ist aber, dass in keinem Fall der hohe Abstraktionsgrad der modellbasierten Lösungen erreicht wird. Dieser wird bei JDaphne auch nur bedingt angestrebt, da vorrangig mit semistrukturierten Inhalten gearbeitet wird. Deren Erstellung und Verknüpfung über Hyperlinks in einer kooperativen Umgebung bringt JDaphne wiederum der Kategorie der Multiautorensysteme nahe. Die umfangreichen Mechanismen solcher Systeme zum gemeinsamen Bearbeiten eines einzelnen Dokumentes in Form von gemeinsamen virtuellen Arbeitsoberflächen fehlen jedoch bei JDaphne, so dass obwohl auch hier eine vollständige Zuordnung schwer möglich erscheint, JDaphne dennoch mit seiner Fähigkeit, ein großes hypermediales Gesamtdokument, die Web-Präsenz, mit vielen Benutzern gleichzeitig zu bearbeiten, in gewisser Hinsicht als Teil dieser Kategorie zu bewerten ist.

Über die Einbindung von externen Datenquellen mit der Mediator-Lösung, die im Anwendungsfall über einen entsprechenden Programmieraufwand realisiert werden muss, zumindest aber genaue Kenntnis des Datenmodells auf programmtechnischer Ebene erfordert, nähert sich JDaphne in gewisser Weise den Datenbank-Integrationslösungen an; auch hier wieder über die Anforderung aus dem Unternehmen gesteuert, werden vor allem einzelne konkrete Abfragen unterstützt. Abstraktere Konzepte zur Datenbankabfrage, wie sie von den komponentenorientierten Datenbank-Exportwerkzeugen geboten werden, sind bei JDaphne nicht integriert.

Wie bereits bei der Kategorie der Middleware und Application-Server beschrieben, ist hier ein Vergleich mit JDaphne nicht möglich, da sie primär nicht zum Web-Präsenz-Management eingesetzt werden und stattdessen eher Anwendungslogik mit einer Web-Schnittstelle versehen. Ihre Server-Funktionalität als Laufzeitumgebung für JDaphne einzusetzen, um damit einen Redaktionssystem-Mechanismus mit dem Application-Server zu implementieren, ist hingegen sehr wohl möglich.

Kapitel 8

Zusammenfassung und Ausblick

Diese Arbeit ist im konkreten Projekt-Umfeld mit dem Ziel entstanden, ein System zu entwickeln, das die Erstellung und den Betrieb einer Web-Präsenz im Unternehmen durch dessen Mitarbeiter ermöglicht. Neben dem rein technischen Aspekt der Entwicklungsarbeit wurde großer Wert auf die übergreifende konzeptionelle Komponente bei der Umsetzung der entsprechenden Projekte gelegt.

Grundlegende Ansätze zur Verwaltung einer Web-Präsenz für einem Unternehmen wurden zu diesem Zweck gegenübergestellt; mit dem Ergebnis, dass die Pflege der Web-Präsenz über eine Agentur (Outsourcing) für einen Einstieg auf Grund der hohen Kostentransparenz auf der einen und einem kompakten Ergebnis auf der anderen Seite geeignet erscheint, eine Integration der Web-Präsenz in das Unternehmen mit einer Eigenverwaltung jedoch aus mehreren Gründen vorzuziehen ist. Wichtigster Grund ist die zu erwartende deutlich höhere Aktualität der auf diese Weise publizierten Inhalte, deren direkte Publikation ohne den Umweg über eine zu Verzögerungen führende Engstelle (Agentur) abgewickelt werden kann.

Weiterhin können unter Ausnutzung der offenen Standards und des vereinheitlichenden Charakters von internetbasierter Technologie intern bereits bestehenden Datenquellen in die Web-Präsenz unter Ausnutzung des daraus resultierenden Technologie-Transfers integriert werden. Unterschieden nach internen und extern nutzbaren Datenbeständen, besteht bei der vom Unternehmen selbst verwalteten Web-Präsenz die zusätzliche Möglichkeit, einen übergreifenden Datenbestand für den analogen Aufbau eines internen Informationssystems zu nutzen (Intranet).

Auf dieser Grundlage wurde im Rahmen dieser Arbeit die Entwicklung und die Verwaltung einer Unternehmens-Web-Präsenz mittlerer Größe, d.h. an deren Betreuung 20 bis 100 Mitarbeiter beteiligt sind, die sich für 1000 bis 5000 Dokumente verantwortlich zeichnen, dargestellt. Für die Bewältigung dieser Aufgabe wurde ein – in einem ersten Schritt für eine Online-Zeitung entworfenes – Online-Redaktionssystem für die speziellen Anforderungen und Rahmenbedingungen von Unternehmens-Web-Präsenzen weiterentwickelt. Unter Berücksichtigung der sich durch verschiedene Projekte jeweils ergebenden Randbedingungen wurde ein webbasiertes System aufgebaut, mit dem den Mitarbeitern direkt vom Arbeitsplatz die kooperative Pflege der Web-Präsenz ermöglicht wird. Entsprechend ihrer Rolle und ihrer Gruppenzugehörigkeit mit verschiedenen Berechtigungen und Funktionen versehen, werden sie von dem Redaktionssystem bei der gemeinsamen Arbeit an der Web-Präsenz unterstützt.

Der Beschreibung dieses durch konkrete Anpassungen jeweils maßzuschneidernden Systems sowie die aus seiner Konzeption und seinem Einsatz gewonnenen Erkenntnisse orientieren sich in dieser Arbeit an dem zugrunde gelegten sechsphasigen Lebenszyklus einer Web-Präsenz: Ausgehend von der Idee und der Projektierung einer Web-Präsenz erfolgt die Konzeptionsphase, die in Wechselwirkung mit prototypischen Entwicklungen in die Imple-

mentationsphase übergeht. An die Betriebsphase schließt sich die Evolutionsphase mit diversen Modifikationen der Web-Präsenz und des diese generierenden Systems an. Bedingt durch die hohe Innovationsrate im IT-Bereich schließt sich der Zyklus mit einer "Neukonzeption", wenn die technischen Inkompatibilitäten sich nicht mehr durch angemessene Modifikationen im Rahmen der Evolution beheben lassen.

Die Entwicklung des produktiv eingesetzten, prototypischen Online-Redaktionssystems „JDaphne" in Java erfolgte in mehreren Schritten und setzte dabei auf drei im Rahmen dieser Arbeit entstandene Bausteine auf. Bei diesen handelt es sich um das in einem ersten Schritt für eine Online-Zeitung entwickelte skriptbasierte Online-Redaktionssystem „DAPHNE", eine dazu entworfene Hyperlink-Management-Komponente sowie eine auf dem Gateway-Prinzip aufgebaute „Online-Recherche-Umgebung".

Die in PERL implementierte Variante „DAPHNE" des Online-Redaktionssystems zeichnete sich durch einfache Anpassbarkeit bei gleichzeitig hoher Funktionalität aus. Einschränkungen ergaben sich neben der schlechten Wartbarkeit des Skript-Codes vor allem durch die fehlende Interaktivität der HTML-basierten Benutzerschnittstelle. Diese wurde durch diverse, vor allem in Java implementierte, Add-ons ergänzt, die letztendlich den Ausgangspunkt für die Entwicklung von JDaphne darstellten.

Ausgehend von der Ausrichtung auf die Online-Zeitung hat sich beim produktiven Einsatz von DAPHNE schnell der Bedarf an umfassenden Hyperlink-Management-Fähigkeiten offenbart. Diese wurden in einem separaten, in Java implementierten Modul erarbeitet, das sowohl für den integrierten Betrieb, als auch für den alleinstehenden Einsatz konzipiert wurde. Neben der Herstellung und Erhaltung von Hyperlink-Konsistenz standen insbesondere Mechanismen für die Verknüpfungen von multilingualen Dokumenten und die Verbesserung der Hyperlink-Verteilung innerhalb der Web-Präsenz durch Hyperlink-Vorschläge im Fokus dieses Prototyps. Beispielsweise aufbauend auf Konzepten des Case Based Reasonings (CBR) leitet die als HLM- (Hyperlink-Management) Komponente bezeichnete Entwicklung vom bestehenden Dokumentenbestand unter Berücksichtigung der verfügbaren Metadaten und der existierenden Verlinkungsstruktur für neue Dokumente Hyperlink-Vorschläge für die Autoren ab.

Aus dem Bedürfnis heraus entwachsen, während des Browsens im Internet, mehr als bislang von den Web-Browsern angeboten, unterstützt zu werden, wurde ein Browser-Assistenten-System entwickelt. Dieses als Gateway aufgebaute, vom Web-Browser wie ein Proxy angesprochene System nimmt dem Benutzer u.a. die Aufgabe ab, seine Browser-Sitzung selbst zu verwalten, indem es alle während des Einsatzes beispielsweise im Rahmen einer Online-Recherche aufgerufenen Dokumente automatisch im Dateisystem ablegt und in der zum System gehörigen Datenbank indiziert. In verschiedenen Ausprägungen, über einfache Abfragemasken für die Datenbank bis hin zu komplexen Navigationsunterstützungen durch eine grafische Visualisierung der Historie von Browser-Sitzungen, erleichtert das System den Autoren die professionelle Arbeit mit dem WWW.

Aus dem erweiterten Arbeitsumfeld der PKI ist die Variante des „Signierenden Gateways" entstanden. Für den Einsatz im wissenschaftlichen Umfeld aufgebaut, ermöglicht diese auf einem zu dem Assistenten-System analogen Prinzip arbeitende Komponente die Beglaubigung von aus dem Internet geladenen Dokumenten durch eine vertrauenswürdige dritte Partei. Dem Trend zur Referenzierung von nur online verfügbaren Dokumenten folgend besteht dringender Bedarf an Verfahren, welche die Integrität und Authentizität solcher Referenzen bestätigen.

Der „Signierende Gateway" nimmt sich dieses Problems an, indem er nach dem Transfer von Dateien diese dem anfragenden Client in signierter Form zur Verfügung stellt; eine Funktion, die unter anderem auch vom Betreiber einer Web-Präsenz integriert und bereitgestellt werden kann, um den Benutzern die Arbeit mit den online angebotenen Dokumenten

insbesondere im offiziellen Umfeld zu erleichtern.

Den Aufbau und die kooperative Verwaltung einer Unternehmens-Web-Präsenz zum Ziel hat das am Institut für Telematik als verteiltes System entwickelte und in Java implementierte, oben erwähnte Online-Redaktionssystem „JDaphne“. Die für ein solches System übliche Eingabekomponente wurde nicht in das System integriert, stattdessen werden bestehende Editor-Anwendungen von Dritt-Herstellern eingebunden. Daher steht bei JDaphne die jeweils aktuellste im Markt verfügbare Eingabekomponente als Dokumentenbearbeitungsschnittstelle zur Verfügung. Die Kehrseite der daraus resultierenden Dateizentrierung liegt in der Erfassung semi-strukturierter Inhalte (HTML).

Eine Erweiterung bzw. Neuausrichtung des Systems für zukünftige Entwicklungen erfordert in diesem Zusammenhang in der Basis die Umstellung auf ein neues, strukturiertes, internes Dokumentenformat, beispielsweise XML-basiert, um besser für eine hardware-abhängige Präsentation (Content-Syndication, z.B. für Mobilgeräte) von Inhalten gerüstet zu sein.

JDaphne's Architektur zur Verwaltung von Dokumenten anhand von eindeutigen Kennungen und durch strukturgebende Ressorts sowie die Funktionalitäten zum umfassenden, Dokumentversion übergreifenden Hyperlink-Management haben sich beim Aufbau von Web-Präsenzen mit den dabei anfallenden Arbeiten bewährt.

Die Erfahrungen aus der Entwicklung und dem Einsatz von DAPHNE und JDaphne haben wertvolle Einblicke in den Entwicklungsprozess von Web-Präsenzen, aufgesplittet in drei Phasen – die Konzeption, die Entwicklung und den Betrieb – erlaubt. Neben den die technischen und organisatorischen Aspekte betreffenden Erkenntnissen war vor allem auch die Beobachtung des Anwenderverhaltens aufschlussreich. Aufgrund der Nähe zu den Projektpartnern, bei denen seit langem der produktive Einsatz erfolgt, konnten wichtige Hinweise bezüglich der benötigten Funktionalitäten aufgegriffen werden. Die direkte Rückkopplung hat dabei zu unternehmensspezifischen Detaillösungen (z.B. Hilfsapplikationen) geführt, die durch eine leichte Abstraktion auch in anderen Kontexten (z.B. einem web-basierten Arztbriefsystem) eingesetzt werden konnten. Andere Überlegungen, wie die Mehrfachnutzung der im System verfügbaren Inhalte für parallele, jeweils benutzergruppenspezifische Internet-Auftritte, zeigen analog die hohe Praxisnähe bei der Entwicklung.

JDaphne wurde weiterhin in das breite Spektrum der Web-Content-Management-Systeme eingeordnet. Zu diesem Zweck sind die heute häufig unter dem globalen Label „Web-Content-Management-System“ geführten Systeme differenzierteren Kategorien zugeordnet und JDaphne im jeweiligen Kontext diskutiert worden. Während bei DAPHNE/JDaphne für die Web-Präsenz-Verwaltung großer Wert auf die webbasierte, verteilte Erstellung und Bearbeitung von Inhalten durch die Mitarbeiter eines Unternehmens gelegt wird, ist für zukünftige Entwicklung mehr Gewicht auf die Publikation und damit die Integration bereits vorhandener Datenquellen zu legen. Die immer komplexer werdenden Web-Präsenzen, in diesem Kontext besser bereits als Webapplikationen bezeichnet, erfordern neue Mechanismen zu ihrer Erstellung und Verwaltung. Gerade im datenintensiven Umfeld wird eine stetig wachsende Dynamik bei ständig ansteigendem Automatisierungsgrad beruhend auf formalen Beschreibungen der Applikation benötigt. Insbesondere für datenintensive Szenarien, die eine von dem System betreute Web-Präsenz über vorgegebene Pfade und komplexe Bedienelemente in die Nähe einer Webapplikation bringen, sind die bislang verfügbaren Mechanismen noch weiter zu entwickeln.

Ziel für die Fortentwicklung von Web-Präsenz-Management-Systemen am Institut für Telematik muss die intensivierte Nutzung der aus dem Software-Engineering bekannten Techniken und Werkzeuge für die automatisierte Generierung von Web-Präsenzen sein, die neben semistrukturierten insbesondere strukturierte Inhalte effizient publizieren und eine Applikation aufbauen können. Basierend auf den im Umfeld von JDaphne erfolgreich eingesetzten Rollen

für die Mitarbeiter und Dokument-Workflows können dann über Qualitätssicherungsprozesse auch für datenintensive Web-Präsenzen auf effektive Weise hochwertige Ergebnisse erzielt werden.

Literaturverzeichnis

[Aar97] B. Aaron. *ActiveX Technical Reference*. Prima Publishing, US, 1997.

[ABC+00] Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Parnell, Jeremy Richman und Steve Zilles, 2000. W3C Candidate Recommendation 21 November).

[ABS00] S. Abiteboul, P. Buneman und D. Suciu. *Data on the Web*. San Francisco: Morgan Kaufmann Publishers, 2000.

[AHK+95] A. Aimar, I. Hannell, A. Khodabandeh, P. Palazzi, B. Rousseau und M. Ruggier. WebLinker, A Tool for Managing WWW cross-references. In *Papers from the Second World Wide Web Conference*, Jgg. 28. Computer Networks and ISDN Systems, 1995.

[AKM95] K. Andrews, F. Kappe und H. Maurer. Hyper-G and harmony: towards the next generation of networked information technology. In *Conference on Human factors in computing systems May 7 - 11, Denver, CO USA*, Seiten 33–34, 1995.

[AMM+98] P. Atzeni, G. Mecca, P. Merialdo, A. Masci und G. Sindoini. Design and Maintenance of Data Intensive Web Sites. In *Proc. of International Conference on Extending Database Technology, EDBT98 in Valencia, Spain.*, Seiten 436–450, 1998.

[And97] K.M. Anderson. Integrating Open Hypermedia Systems with the World Wide Web. In *UK Conference on Hypertext*, Seiten 157–166, 1997.

[Ang96] J. Angel. Web Publisher's Construction Kit With Netscape Plug-Ins, 1996.

[BAB+97] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor und G. Woetzel. Basic Support for Cooperative Work on the World Wide Web. *International Journal of Human Computer Studies*, 46:827–846, 1997.

[Bal97] M. Baldschus. *Das Medium Internet ist heute nicht mehr wegzudenken. Verlegerische Überlegungen zum Zusammenspiel von Content und Werbung*, Seiten 46–47. Media-Daten & Fakten Net-Book, 1 1997.

[BB98] B. Balasubramanian und A. Bashian. Document Management and Web-Technologies: Alice Marries the Mad Hatter. *Communications of the ACM*, 41(7):107–115, 1998.

[BB00] J. Busse und J. Abbott Bulka. Special Report: Design Usability - One Size Doesn't Fit All. *Internet World Magazine*, 12 2000.

[BBC+97] D. Bhatia, V. Burzevski, M. Camuseva, G. Fox, W. Furmanski und G. Premchandran. WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing. *Concurrency - Practice and Experience*, 9(6):555–577, 1997.

[BDM96] J. Bern, C. Damm und C. Meinel. ECCC – ein elektronisches Kolloquium im Internet. *Informatik-Spektrum*, 19(4):230–231, 1996.

[Ber99] T. Berker. *Internetnutzung im Alltag. Zur Geschichte, Theorie, Empirie und Kritik der Nutzung eines „jungen" Mediums.* Dissertation, Johann Wolfgang Goethe-Universität, Frankfurt am Main, 1999.

[Ber01] L. Bergmann. Website-Marketing, 2001. Online verfügbar, Juli 2001 unter http://www.e-publishing.de/.

[BLCGP92] T. Berners-Lee, R. Cailliau, J-F. Groff und B. Pollermann. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 1(2), Spring 1992.

[BLFF96] T. Berners-Lee, R. Fielding und H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0, 1996. RFC-1945.

[BLFM98] T. Berners-Lee, R. Fielding und L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax, 1998. RFC-2396.

[BM98] R. Barrett und P.P. Maglio. Intermediaries: New Places for Producing and Manipulating Web Content. In *Seventh International World Wide Web Conference, Brisbane, Australia*, 1998.

[BMK97] R. Barrett, P.P. Maglio und D.C. Kellem. How to Personalize the Web. In *Proc. of Computer-Human Interaction (CHI)*, Seiten 75–82, 1997.

[BMS01] J. Bestgen, T. Meier und C. Schmidt. *IT-Konzepte für das Wissensmanagement.* Books on Demand, Norderstedt, 2001.

[BPSMM98] T. Bray, J. Paoli, C. Sperberg-McQueen und E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition), 1998. W3C Recommendation.

[BR01] R. Belew und C. J. Van Rijsbergen. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW.* Cambridge Univ Press, 2001.

[Bra00] R. Brandtweiner. *Differenzierung und elektronischer Vertrieb digitaler Informationsgüter.* Symposion Publications, 2000.

[BS98] D. Boles und M. Schlattmann. Multimedia-Autorensysteme: Grafisch-interaktive Werkzeuge zur Erstellung multimedialer Anwendungen. *LOG IN - Informatische Bildung und Computer in der Schule*, 18(1):10–18, 1998.

[BUD+00] D. Buser, C. Ullman, J. Duckett, J. Kauffman, J. Llibre, B. Francis, D. Sussman und J. T. Llibre. *Beginning Active Server Pages 3.0.* Wrox Press Inc., 2000.

[CC01] „Online-Nutzung 2001" - Technikausstattung der Nutzer. Panel report, ComCult Research GmbH - Marktforschung für die Neuen Medien, 2001.

[CDHV00] Y.-F. Chen, F. Douglis, H. Huang und K.-P. Vo. TopBlend: An Efficient Implementation of HtmlDiff in Java. In *Proc. of WebNet 2000 - World Conference on the WWW and Internet*, number 1, Seiten 88–94. AACE, 2000.

[CDK00] G. Coulouris, J. Dollimore und T. Kindberg. *Distributed Systems: Concepts and Design, Edition 3.* ADDISON WESLEY, 2000.

[CFB00] S. Ceri, P. Fraternali und A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. In *Proc. of the 9th World Wide Web Conference (WWW9), Amsterdam*, Mai 2000.

[CIP01] R. Cattell, J. Inscore und Enterprise Partners. *J2EE(tm) Technology in Practice: Building Business Applications with the Java(tm) 2 Platform, Enterprise Edition.* Addison-Wesley, 2001.

[Con87] J. Conklin. Hypertext: An introduction and survey. *Computer*, 20(9):17–41, 1987.

[CP95] L.D. Catledge und J.E. Pitkow. Characterizing browsing strategies in the World-Wide Web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.

[Cri96] M. Crispin. Internet Message Access Protocol - Version 4 Revision 1, 1996. RFC-2060.

[CSM97] R. Cooley, J. Srivastava und B. Mobasher. Web Mining: Information and Pattern Discovery on the World Wide Web. In *Proc. of the 9th International Conference on Tools with Artificial Intelligence (ICTAI'97).* IEEE Computer Society, 1997.

[CVJ99] W. Chisholm, G. Vanderheiden und I. Jacobs. Web Content Accessibility Guidelines 1.0, 1999. W3C Recommendation.

[CW00] D. Connolly und World Wide Web Consortium (W3C). The 'text/html' Media Type, 2000. RFC-2854.

[DD97] C.J. Date und H. Darwen. *A Guide to SQL-Standard, Reading, MA.* Addison - Wessley, 1997.

[Dem97] L. Dempsey. Metadata: An Overview of Current Resource Description Practice. *Journal of Documentation*, 1997.

[DGB96] D.Schwabe, G.Rossi und S.D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proc. of Conference on Hypertext, UK*, Seiten 116–128, 1996.

[DH89] R.T. Durst und K.D. Hunter. US Patent no. :5,022,080 Electronic Notary; Filing date: April 16, 1989.

[DMO01] S. DeRose, E. Maler und D. Orchard. XML Linking Language (XLink) Version 1.0, Juni 2001. W3C Recommendation.

[DO85] G.B. Davis und M. Olson. *Management Information Systems: Conceptual Foundations, Structure, and Development.* New York: McGraw-Hill Book Company, 1985.

[Dör01] T. Dörstling. Digitale Signaturen und Dokumentenmanagement – Der elektronische Fingerabdruck – Sicherheit für alle Dokumente. *IT-Sicherheit*, 7(2):23–26, April 2001.

[Eck00] B. Eckel. *Thinking in Java, 2nd Edition.* Prentice-Hall, 2000.

[End96] S. Endter. Internet – (k)ein urheberrechtlich geschützter Raum? *Neue Juristische Wochenschrift*, Seiten 967–968, 1996.

[Far98] J. Farley. *Java Distributed Computing.* O'Reilly & Associates, 1998.

[FBM+00] B. Forta, D. Bromby, R. Mandel, P. Fonte, K. Lauver und R. Juncker. *WAP Development with WML and WMLScript.* Sams, 2000.

[FFK+97] M. Fernandez, D. Florescu, J. Kang, A. Levy und D. Suciu. STRUDEL: a Web site management system. In *Proc. of International Conference on Management of Data*, Seiten 549–552. ACM-SIGMOD, 1997.

[FKK96] A.O. Freier, P. Karlton und P.C. Kocher. The SSL Protocol Version 3.0, 1996. (INTERNET-DRAFT).

[Fla97] D. Flanagan. *JavaScript: The Definitive Guide.* O'Reilly & Associates, 1997.

[Fle98] Jennifer Fleming. *Web Navigation: Designing the User Experience.* O'Reilly & Associates, 1998.

[FP98] P. Fraternali und P. Paolini. A conceptual model and a tool environment for developing more scalable and dynamic Web applications. In *Proc. of the Conference On Extended Database Technology (EDBT)*, 1998.

[Fra99] P. Fraternali. Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys*, 31(3):227–263, 1999.

[Fre94] S. Freisler. Hypertext – Eine Begriffsbestimmung. *Deutsche Sprache*, (1):19–50, 1994.

[Fre97] S. Freisler. *Objekt-orientierte Konzepte für die Hypertext-Produktion.* Verlag Schmidt-Römhild, 1997.

[Ger01] R.W. Gerling. Betrieb von WWW-Servern – Rechtliche und technische Aspekte. *IT-Sicherheit*, 3:18–22, 2001.

[Ges00] Bibliographische Gesellschaft, Hrsg. *Der Brockhaus in einem Band, 9. vollständig überarbeitete und aktualisierte Auflage.* F.A. Brockhaus GmbH, Leipzig - Mannheim, 2000.

[GHJV94] E. Gamma, R. Helm, R. Johnson und J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley, Massachusetts, 1994.

[GL00] D. Goh und J. Leggett. Patron-augmented Digital Libraries. In *Proc. of the Fifth ACM International Conference on Digital Libraries*, 2000.

[GLR95] A. Ginige, D.B. Lowe und J. Robertson. Hypermedia Authoring. *IEEE Multimedia*, 2(4), 1995.

[GM95] J. Gosling und H. McGilton. The Java Language Overview: A White Paper. Technical report, Sun Microsystems, 1995.

[GM01] A. Ginige und S. Murugesan. Web Engineering: An Introduction. *MultiMedia*, 8(1):14–18, Januar-März 2001.

[GMM+99] J. Gettys, J. Mogul, L. Masinter, P. Leach und T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, 1999. RFC-2616.

[Gol90] C. Goldfarb. *The SGML Handbook*. Oxford University Press, 1990.

[Gro99] The Open Group. Product Standard Directory: LDAP 2000, 1999. Document Number: X99DI.

[Gro01a] W. Grosky, Hrsg. *Web Engineering Part 1*, Jgg. 8. IEEE Computer Society, Januar-März 2001.

[Gro01b] W. Grosky, Hrsg. *Web Engineering Part 2*, Jgg. 8. IEEE Computer Society, April-Juni 2001.

[Gun00] S. Gundavaram. *CGI Programming on the World Wide Web, 2nd Edition*. O'Reilly & Associates, 2000.

[GZEM97] R. Graßmann, P. Zimmermann, T. Engel und C. Meinel. DAPHNE - eine Multiautorenumgebung zur Erstellung komplexer Hypertext-Dokumente, 1997. Workshop Digitale Bibliotheken Aachen 97.

[Haf01] E.-G. Haffner. *Request-Prediction and Hyperlink-Proposals. Methodologies and Mathematics behind Web-Applications*. Dissertation, Universität Trier, FB IV; Informatik, Trier, Deutschland, 2001.

[Hal00] M. Hall. *Core Servlets and JavaServer Pages (JSP)*. Prentice Hall PTR/Sun Microsystems Press, 2000.

[Ham96] R. Hammwöhner. *Offene Hypertextsysteme: Das Konstanzer Hypertext-System im technischen und wissenschaftlichen Kontext*. Habilitationsschrift Universität Konstanz, Informationswissenschaft, 1996.

[HC00] J. Hunter und W. Crawford. *Java Servlet Programming (2nd Edition)*. O'Reilly & Associates, 2000.

[HFPS99] R. Housley, W. Ford, W. Polk und D. Solo. Public Key Infrastructure Certificate and CRL Profile, 1999. RFC 2459.

[HHR+99a] E.-G. Haffner, A. Heuer, U. Roth, T. Engel und C. Meinel. The Importance of Link-Transformation and Link-Proposals for Hyperlink-Management-Systems. In *Proc. of WebNet'99, Honolulu (Hawaii)*. AACE, 1999.

[HHR+99b] A. Heuer, E.-G. Haffner, U. Roth, Z. Zhang, T. Engel und C. Meinel. Hyperlink Management System for Multilingual Web-Sites. In *Proc. of World Wide Web: Technologies and Applications For The New Millenium*, Seiten 303–307, 1999.

[HHRM00] A. Heuer, E.-G. Haffner, U. Roth und C. Meinel. A Hyperlink Focused Browse Assistent for the World Wide Web. In *Proc. of 1th IC'2000, Las Vegas (USA)*, Seiten 79–84, 2000.

[HHZM00] A. Heuer, E.-G. Haffner, Zhongdon Zhang und C. Meinel. Role-based Web-Authoring and Web-Site Management for Commercial Sites. In *Proc. of the International Conference on Internet Computing IC'2000*. CSREA Press, 2000.

[HLM00a] A. Heuer, F. Losemann und C. Meinel. Logging and Signing Document-Transfers on the WWW - A Trusted Third Party Gateway. In *Proc. of Web Information Systems Engineering*, Seiten 146–152, 2000.

[HLM00b] A. Heuer, F. Losemann und C. Meinel. Signed Preservation Of Online References. In *Proc. of World Conference on the WWW and Internet, AACE WebNet 2000, San Antonio, (Texas, USA)*. AACE, 2000.

[HM98] K. Holtman und A. Mutz. Transparent Content Negotiation in HTTP, 1998. RFC-2295.

[HM99a] A. Heuer und C. Meinel. Database based History Browse Assistant. In *Proc. of IASTED International Conference, Internet and Multimedia Systems and Applications (IMSA '99)- Nassau, Bahamas*. IASTED, October 18-21 1999.

[HM99b] A. Heuer und C. Meinel. Database based Navigation Assistant. In *Proc. of WebNet 99 - World Conference on the WWW and Internet, Honolulu, Hawaii, USA, October 24-30*, number 1, Seiten 505–510. AACE, 1999.

[HMT87] F.G. Halasz, T.P. Moran und R.H. Trigg. NoteCards in a nutshell. In J. M. Carroll und P. P. Tanner, Hrsg., *Proc. of Human Factors in Computing Systems and Graphics Interface '87*, Seiten 45–52, 1987.

[HNW+01] A. Le Hors, G. Nicol, L. Wood, M. Champion und S. Byrne. Document Object Model (DOM) Level 3 Core Specification Version 1.0, 2001. W3C Working Draft.

[Hor99] T. Horn. *Internet - Intranet - Extranet – Potentiale im Unternehmen*. R. Oldenbourg Verlag, 1999.

[HRH+00a] E.-G. Haffner, U. Roth, A. Heuer, T. Engel und C. Meinel. Advanced Techniques for Analyzing Web Server Logs. In *Proc. of 1th IC'2000, Las Vegas (USA)*, Seiten 71–78, 2000.

[HRH+00b] E.-G. Haffner, U. Roth, A. Heuer, T. Engel und C. Meinel. Link Proposals with Case-Based Reasoning Techniques. In *Proc. of WebNet 2000, San Antonio, (Texas, USA)*. AACE, 2000.

[HRH+00c] E.-G. Haffner, U. Roth, A. Heuer, T. Engel und C. Meinel. What do Hyperlink-Proposals and Reqauest-Prediction have in Common? In *Proc. of ADVIS'00, Turkey*, 2000.

[HRHM00] E.-G. Haffner, U. Roth, A. Heuer und C. Meinel. Advanced Studies on Link Proposals and Knowledge Retrieval of Hypertexts with CBR. In *Proc. of EC-Web 2000, Grennwich, (United Kingdom)*, 2000.

[HS94] F. Halasz und M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

[HS00] I. Harms und W. Schweibenz. Testing Web Usability. *Information Management & Consulting*, 15(3):61–66, 2000.

[Hun98] C. Hunt. *TCP/ IP. Netzwerk-Administration.* O'Reilly & Associates, Inc., 1998.

[HZ00] S. Hughes und A. Zmievski. *PHP Developer's Cookbook.* Sams, 2000.

[HZEM99] A. Heuer, Z. Zhang, T. Engel und C. Meinel. DAPHNE - Distributed Authoring and Publishing in a Hypertext and Network Environment. In *Proc. of IuK99 - Dynamic Documents, Jena*, 1999.

[I01] BGBl. I. Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften, 2001.

[ICL96] D. Ingham, S. Caughey und M. Little. Fixing the "Broken-Link'' problem: the W3Objects approach. *Computer Networks and ISDN Systems*, 28(7–11):1255–1268, 1996.

[Inc00] Adobe Systems Inc. *PDF Reference, Second Edition: Version 1.3.* Addison-Wesley Pub Co, 2000.

[IT89] P.M. Irish und R.H. Trigg. Supporting Collaboration in Hypermedia: Issues and Experiences. *Journal of the American Society for Information Science*, 40(3):192–199, 1989.

[Jac96] M.A. Jackson. Designing a leading-edge World Wide Web site. In *Proc. of the 14th annual international conference on Marshaling new technological forces: building a corporate, academic, and user-oriented triangle, Annual ACM Conference on Systems Documentation October 19 - 22, 1996, Research Triangle United States*, Seiten 281–283. ACM, 1996.

[JHP00] A. Jain, L. Hong und S. Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[JK00] A. Joshi und R. Krishnapuram. On Mining Web Access Logs. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Seiten 63–69, 2000.

[KK00] E. Kirda und C. Kerer. MyXML: An XML based template engine for the generation of flexible web content. In *WebNet 2000 - World Conference on the WWW and Internet San Antonio, Texas October 30 - November 4*, Jgg. 1, Seiten 317–. Association for the Advancement of Computing in Eduction (AACE), 2000.

[KK01] M. Kofler und D. Kramer. *MySQL.* aPress, 2001.

[KM99] R. Kogeler und J. Müffelmann. *Multimedia und die Zukunft der Printmedien aus der Sicht des Axel Springer Verlages.* Gabler Verlag, 1999.

[Koh78] L. Kohnfelder. Towards a Practical Public-key Cryptosystem. Bachelor's thesis, MIT, Mai 1978.

[KRS00] G. Kappel, W. Retschitzegger und W. Schwinger. Modeling Customizable Web Applications - A Requirement's Perspective. In *Proc. of the International Conference on Digital Libraries, ICDL*, 2000.

[Lan01] R. Lankau. *Webdesign und -publishing, Grundlagen und Designtechniken, 3. aktualis. u. überarb. Aufl.* Carl Hanser Verlag, 2001.

[Law01] S. Lawrence. Online or Invisible? *Nature*, 411(6837):521, 2001.

[LB99] H.W. Lee und B. Bos. Cascading Style Sheets, level 1,W3C Recommendation 17 Dec 1996, revised 11 Jan 1999, 1999. REC-CSS1.

[LC98] K. Larson und M. Czerwinski. Web page design: Implications of memory. In *CHI '98 Conference:Human Factors in Computer Systems*, Seiten 25–32. ACM Press, 1998.

[Len00] Markus Lenz. "Unternehmensstrategie in konvergierenden Branchen" Eine Analyse am Beispiel Internet, Medien-, Informationstechnologie- und Telekommunikationsmärkte. Technical report, Philipps-Universität Marburg, Fachbereich Wirtschaftswissenschaften Lehrstuhl für Allgemeine Betriebswirtschaftslehre, Organisation und Personalwirtschaft, 2000.

[LG99] S. Lawrence und C.L. Giles. Accessibility of Information on the Web. *Nature*, 400:107–109, 1999.

[LH01] C. Ladewig und M. Henkes. Verfahren zur automatischen inhaltlichen Erschließung von elektronischen Texten – ASPECTIX. *nfd – Information – Wissenschaft und Praxis*, 52:159–164, April/Mai 2001.

[Lin01] M. Linden. Check it out. *Page*, 07 2001. MACup Verlag.

[LL99] B. Laurie und P. Laurie. *Apache the Definitive Guide.* O'Reilly & Associates, 1999.

[LM90] X. Lai und J. L. Massey. A Proposal for a New Block Encryption Standard. In I. B. Damgård, Hrsg., *Advances in Cryptology–EUROCRYPT 90*, Jgg. 473 of *Lecture Notes in Computer Science*, Seiten 389–404. Springer-Verlag, 1991, 21–24 Mai 1990.

[LNHL00] J. Lin, M.W. Newman, J.I. Hong und J.A. Landay. DENIM: finding a tighter fit between tools and practice for Web site design. In *Proc. of Computer-Human Interaction (CHI)*, Seiten 510–517, 2000.

[Loe00] H. Loeser. Shift it to the Server! -Let the Database Server Update Your Web-Sites. In *Web Information Systems Engineering*, Seiten 50–54, 2000.

[LS99] O. Lassila und R. Swick. Resource Description Framework Model and Syntax Specification, W3C Recommendation, Februar 1999.

[Luo98] A. Luotonen. *Web Proxy Servers.* Prentice Hall, 1998.

[MAM+98] G. Mecca, P. Atzeni, P. Merialdo, A. Masci und G. Sindoni. From Databases to Web-Bases: The ARANEUS Experience. Technical Report RT-DIA-341998, Universita Degli Studi Di Roma Tre, Mai 1998.

[McL00] B. McLaughlin. *Java and XML.* O'Reilly & Associates, Juni 2000.

[MCS+01] B.A. Myers, J.P. Casares, S. Stevens, L. Dabbish, D. Yocum und A. Corbett. A Multi-View Intelligent Editor for Digital Video Libraries. In *JCDL'01: Proc. of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, Tools for Constructing and Using Digital Libraries, Seiten 106–115, 2001.

[Mer00] P. Mertens. *Integrierte Informationsverarbeitung, 2 Bde., Bd.1, Administrations- und Dispositionssysteme in der Industrie, 12. Auflage.* Th. Gabler Verlag, 2000.

[Mey89] N. Meyrowitz. *he Society of Text: Hypertext, Hypermedia, and the Social Construction of Information – The missing link: why we're all doing hypertext wrong*, Seiten 107–114. MIT Press, 1989.

[MH00] R. Monson-Haefel. *Enterprise Javabeans (2nd edition).* O'Reilly & Associates, 2000.

[Mil56] G.A. Miller. The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63:81–96, 03 1956.

[MKPS92] H. Maurer, F. Kappe, G. Pani und F. Schnabel. Hyper-G: A Modern Hypermedia System. In *Proc. Network Services Conference (NSC)'92, Pisa, Italy*, Seiten 35–36, 1992.

[MR94] J. Myers und M. Rose. Post Office Protocol - Version 3, 1994. RFC-1725.

[Nat92] National Institute of Standards and Technology (NIST). The Digital Signature Standard, proposal and discussion. *Communications of the ACM*, 35(7):36–54, Juli 1992.

[NBO+99] J. Nail, B. Bass, C. O'Connor, J. Aldort und T. Grimsditch. The New Business Portals. Technical report, The Forrester Report, Februar 1999.

[Net01] Netcraft. The Netcraft Web Server Survey, July. Online verfügbar Juli 2001, http://www.netcraft.com/survey/index-200107.html, 2001.

[Nie95] J. Nielsen. *Hypertext and Hypermedia: the Internet and Beyond.* Academic Press, 1995.

[Nie00] J. Nielsen. *Designing Web Usability: The Practice of Simplicity.* New Riders Publishing, Indianapolis, 2000.

[NL00] M.W. Newman und J.A. Landay. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In *Symposium on Designing Interactive Systems*, Seiten 263–274, 2000.

[O'N99] E. O'Neill. "Web Sites: Concepts, Issues, and Definitions". Technical report, CLC Online Computer Library Center, Inc., Januar 1999. Research Notes.

[otWHWG00] Members of the W3C HTML Working Group. XHTML 1.0: The Extensible HyperText Markup Language A Reformulation of HTML 4 in XML 1.0, 2000. W3C Recommendation.

[PJ96] J.E. Pitkow und R.K. Jones. Supporting the Web: A distributed hyperlink database system. *Computer Networks and ISDN Systems*, 28(7–11):981–991, 1996.

[Poh01] N. Pohlmann. Bessere Identifizierung, sichere Authentisierung. *IT-Sicherheit*, 7(2):13–21, April 2001.

[Pop98] A. Pope. *The CORBA Reference Guide.* Addison-Wesley, 1998.

[Pos82] J.B. Postel. Simple Mail Transfer Protocol, 1982. RFC-821.

[Pow98] T. Powell. *Web Site Engineering.* Prentice Hall, 1998.

[Pra01] A. Pradka. Gestörte Verbindung zur Außenwelt. *CYbiz*, Seiten 62–65, August 2001.

[Pre93] B. Preneel. *Analysis and Design of Cryptographic Hash Functions.* Dissertation, K. U. Leuven, Leuven, Belgium, Januar 1993.

[PS00] G. Preuner und M. Schrefl. A Three-Level Schema Architecture for the Conceptual Design of Web-Based Information Systems: From Web-Data Management to Integrated Web-Data and Web-Process Management. *World Wide Web Journal, Special Issue on World Wide Web Data Management*, 3(2):125–138, 2000.

[QEM00a] C. Qu, T. Engel und C. Meinel. Implementation of a Document Management System Based on WebDAV Protocol. In *1th IEEE International Conference on Management of Innovation and Technolgoy (ICMIT'2000), Singapore (Malaysia)*, 2000.

[QEM00b] C. Qu, T. Engel und C. Meinel. Implementation of an Enterprise-level Groupware System Based on J2EE Platform and WebDAV Protocol. In *4th IEEE International Enterprise Distributed Computing Conference EDOC'00, Makuhari (Japan)*, 2000.

[Rae01] M. Raepple. *Sicherheitskonzepte für das Internet, 2.Auflage.* dpunkt.verlag, 2001.

[Rap01] K. Rapke. Automatische Indexierung von Volltexten für die Gruner+Jahr Pressedatenbank. *nfd – Information – Wissenschaft und Praxis*, 52:251–262, Juli/August 2001.

[RBR98] W. Rubin, M. Brain und R. Rubin. *Understanding DCOM.* Prentice Hall, 1998.

[RC99] H.C.-H. Rao und Y.-F. Chen. A Proxy-Based Personal Portal. In *WebNet'99, Honolulu (Hawaii).* AACE, 1999.

[Ree97] G. Reese. *Database Programming with JDBC and Java.* NY,O'Reilly & Associates, 1997.

[Ren00] O. Rengelshausen. *Online-Marketing in deutschen Unternehmen, Untertitel Einsatz - Akzeptanz - Wirkungen.* Dissertation, Georg-August-Universität Göttingen, 2000.

[RF97] D. Rusch-Feja. Mehr Qualität im Internet: Entwicklung und Implementierung von Metadaten. In *Online-Tagung der DGD. Die Zukunft der Recherche - Rechte, Ressourcen und Referenzen. Frankfurt/Main*, Seiten 113–130,123–127. M. Ockenfeld and R. Schmidt, 1997.

[RHH+99] U. Roth, E.-G. Haffner, A. Heuer, T. Engel und C. Meinel. Hyperlinkmanagment HLM. Technical Report 99-05, Institut für Telematik, Trier, 1999.

[Rho97] J. Rhoton. *SMTP, X.500, X.400: An Introduction*. Digital Press, 1997.

[Riz00] T. Rizzo. Know Your Users. *Internet World Magazine*, 12 2000.

[RJB99] J. Rumbaugh, I. Jacobson und G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, Massachusetts, USA, 1999.

[RM98] L. Rosenfeld und P. Morville. *Information Architecture for the World Wide Web*. O'Reilly & Associates, 1998.

[Rod91] T. Rodden. A survey of CSCW systems. *Interacting with Computers*, 3(3):319–353, 1991.

[RS98] R.A.Barta und M.W. Schranz. JESSICA: an Object-oriented Hypermedia Publishing Processor. *Computer Networks and ISDN Systems*, 30:239–249, 1998.

[RSA78] R. L. Rivest, A. Shamir und L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, Februar 1978.

[RSL99] G. Rossi, D. Schwabe und F. Lyardet. Web Application Models Are More Than Conceptual Models. In *ER Workshops*, Seiten 239–253, 1999.

[RSVW94] W. Reinhard, J. Schweitzer, G. Volksen und M. Weber. CSCW Tools: Concepts and Architectures. *Computer*, 27(5):28–36, 1994.

[SCFY96] R.S. Sandhu, E.J. Coyne, H.L. Feinstein und C.E. Youman. Role-based Access Control Models. *IEEE Computer*, 20(2):38–47, 1996.

[Sch89] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In J.-J. Quisquater und J. Vandewalle, Hrsg., *Advances in Cryptology–EUROCRYPT 89*, Jgg. 434 of *Lecture Notes in Computer Science*, Seiten 688–689. Springer-Verlag, 1990, 10–13 April 1989.

[Sch96] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., New York, NY, USA, second. Auflage, 1996.

[Sch01] P. Schaar. Persönlichkeitsprofile im Internet. *DuD – Datenschutz und Datensicherheit*, 25(7):383–388, 2001.

[SD92] H. Shen und P. Dewan. Access Control for Collaborative Environments. In Jon Turner und Robert Kraut, Hrsg., *Proc ACM Conf. Computer-Supported Cooperative Work, CSCW*, Seiten 51–58. ACM Press, 31 –4 1992.

[SG99] J. Schneeberger und S. Göttel. Navigation in zusammengesetzten Hypertexten. *tekom Nachrichten*, (2), 1999.

[SGHH94] N. Streitz, J. Geiler, J. Haake und J. Hol. DOLPHIN: Integrated meeting support across LiveBoards, local and desktop environments. In *Proc. of CSCW '94 Conference*, Seiten 345–358. ACM Press, 1994.

[Sha96] K.E. Shafer. Persistent Uniform Resource Locators (PURLs). In *International Conference on Digital Libraries*. ACM Press, 1996.

[SHH+92] N.A. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt und M. Thüring. SEPIA: A Cooperative Hypermedia Authoring Environment. In *Proc. of the ACM Conference on Hypertext (ECHT'92)*, Seiten 11–22. ACM Press, 1992.

[SHT89] N.A. Streitz, J. Hannemann und M. Thüring. From Ideas and Arguments to Hyperdocuments: Traveling through Activity Spaces. In *Proc. of the 2nd ACM Conference on Hypertext (Hypertext 89)*, Seiten 343–364. ACM Press, 1989.

[Sie98] D. Siegel. *Das Geheimnis erfolgreicher Websites. Business, Budget, Manpower, Lizenzen, Design*. Markt & Technik, 1998.

[Sik97] K. Sikkel. A Group-based Authorization Model for Cooperative Systems. In *Proc. European Conf. on Computer Supported Cooperative Work (ECSCW'97), Lancaster*, Seiten 345–360, 1997.

[SPF99] M. Spiliopoulou, C. Pohle und L. Faulstich. Improving the Effectiveness of a Web Site with Web Usage Mining. In *WEBKDD*, Seiten 142–162, 1999.

[Sta01] W. Stalling. *Sicherheit im Internet - Anwendungen und Standards*. Addison Wesley, 2001.

[SZAS97] C. Shahabi, A.M. Zarkesh, J. Abidi und V. Sha. Knowledge discovery from user's web-page navigation. In *Proc. Seventh IEEE International Workshop on Research Issues in Data Engineering(Ride)*, Seiten 20–29, 1997.

[TBR98] D.A. Tietze, A. Bapat und R. Reinema. Document-Centric Groupware for Distributed Governmental Agencies. In *Conference on Advanced Information Systems Engineering*, Seiten 173–190, 1998.

[TG97] L. Tauscher und S. Greenberg. How people revisit web pages: empirical findings and implications for the design of history mechanisms. *Int. J. of HumanComputer Studies*, 47:94–137, 1997.

[Tho01] P. Thomas. Einfluss eines Integrierten Content Management Systems auf Arbeitsabläufe in der Rundfunkanstalt. *nfd – Information – Wissenschaft und Praxis*, 52:283–291, Juli/August 2001.

[TL97a] F. Yellin T. Lindholm. *The Java Virtual Mashine Specification*. Adison-Wesley, 1997.

[TL97b] K. Takahashi und E. Liang. Analysis and design of Web-based information systems. *Computer Networks and ISDN Systems*, 29(8–13):1167–1180, September 1997.

[WG99] E. James Whitehead, Jr. und Y.Y. Goland. WebDAV: A network protocol for remote collaborative authoring on the Web. In *Proc. of the Sixth European Conf. on Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, September 12-16*, Seiten 291–310, 1999.

[Wie92] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25:38–49, 1992.

[Wii91] U. K. Wiil. Issues in the design of EHTS: a multiuser hypertext system for collaboration. In *Proc. of HICSS-25 (Kauai, HI, Jan)*, Jgg. 2, Seiten 629–639. IEEE Computer Society Press, 1991.

[Wii99] U.K. Wiil. Multiple Open Services In A Structural Computing Environment. In Peter J. Nürnberg, Hrsg., *Proc. of the First Workshop on Structural Computing,Aalborg University Esbjerg Computer Science Department, Technical Report AUE-CS-99-04*, 1999.

[Wil99] E. Wilde. *World Wide Web. Technische Grundlagen.* Springer-Verlag, ISBN:3540647007, 1999.

[WKLW98] S. Weibel, J. Kunze, C. Lagoze und M. Wolf. Dublin Core Metadata for Resource Discovery. Technical Report 2413, IETF. The Internet Society, September 1998.

[WS92] L. Wall und R.L. Schwartz. *Programming Perl.* O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1992.

[WS99] R. Wilkinson und A.F. Smeaton. Automatic link generation. *ACM Computing Surveys*, 31(4es), 1999.

[ZHEM99a] Z. Zhang, A. Heuer, T. Engel und C. Meinel. DAPHNE - A Distributed Tool for Web Authoring and Publishing. In *Proc. of ASIS'99 - Annual Conference of American Society for Information Science, Washington, USA*, 1999.

[ZHEM99b] Z. Zhang, A. Heuer, T. Engel und C. Meinel. DAPHNE - A Tool for Distributed Web-Authoring and Publishing. Technical Report 99-03, Institut für Telematik, Trier, 1999.

[ZHH+99] Z. Zhang, E.-G. Haffner, A. Heuer, T. Engel und C. Meinel. Role-based Access Control in Online Authoring and Publishing Systems vs. Document Hierarchy. In *Proc. of SIGDOC'99*, 1999.

[ZHZ+00] Z. Zhang, A. Heuer, X. Zuo, T. Engel und C. Meinel. Redaktionssystem DAPHNE. Technical Report 00-01, Institut für Telematik, Trier, 2000.

[Zib00] K. Zibell. Klare's Usefull Information is Usefull for Web Designers. *Journal of Computer Documentation*, 24(3):141–147, 2000.

[ZXH98] O.R. Zaiane, M. Xin, und J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Proc. Advances in Digital Libraries Conference (ADL'98), Santa Barbara, CA, April*, Seiten 19–29, 1998.

Glossar

Apache Open Source Web-Server Programm

Applet Ein über das Internet geladenes Programm (Java), dass von dem Web-Browser ausgeführt werden kann. Applets werden in HTML-Seiten eingebunden.

ASCII (*American Standard Code for Information Interchange*) Eine von zwei klassischen Codierungen von Zeichen (Buchstaben, Zahlen, Satzzeichen) in einem Code, der die ersten sieben Bit eines Bytes benutzt. Umlaute sind nicht vorgesehen. In amerikanischen Mailboxen und im Internet werden Texte normalerweise im ASCII-Code dargestellt.

Auflösung (Grafikkarte / Monitor) Unter Auflösung versteht man die Anzahl der waagerechten und senkrechten Bildpunkte (Pixel), aus denen sich ein Monitorbild zusammensetzt. Grundsätzlich gilt: je höher die Auflösung des Bildes, desto detailreicher ist die Darstellung des Bildschirminhaltes und desto größer ist der verfügbare Arbeitsbereich auf dem Bildschirm.

Auszeichnungssprache HTML ist zum Beispiel eine Auszeichnungssprache (Markup Language). Eine Auszeichnungssprache hat die Aufgabe, die logischen Bestandteile eines Dokuments zu beschreiben. HTML enthält daher Befehle zum Markieren typischer Elemente eines Dokuments, wie Überschriften, Textabsätze, Listen, Tabellen oder Grafikreferenzen.

Authentifizierung Mit der Authentifizierung wird versucht, die Identität einer sich ausweisenden Person festzustellen. In der Kryptographie wird oft vereinbart, dass ein bestimmtes Geheimnis eindeutig einer Person zuzuordnen ist.

Authentisierung Mit der Authentisierung versucht sich eine Person durch spezifische Merkmale auszuweisen. Allgemein kann dazu eine Fähigkeit, der Besitz eines Merkmals oder das Wissen um ein Geheimnis dienen.

Authentizität Authentizität bedeutet, dass der Sender einer Nachricht dem Empfänger die Identität sowie die Integrität und Echtheit der Nachricht nachweisen kann.

Autorensystem Komfortable Entwurfsumgebung welche Autoren die Erstellung interaktiver Multimediaanwendungen ohne tiefere Systemkenntnisse oder Programmierkenntnisse ermöglicht.

Autorisierung Vergabe von Rechten nach einer erfolgreichen Identifikation und Authentisierung.

Betriebssystem Ein Betriebssystem ist die Software eines Computers, die angeschlossene Geräte, Dateien und Programme kontrolliert, steuert und überwacht, so dass ein Arbeiten mit dem PC erst ermöglicht wird. Betriebsysteme sind u.a. *UNIX*, *Windows*, *MacOS*, *Linux*, *BeOS* oder *DOS*.

b2b (*Business-to-Business*) Elektronischer Handel über das Internet zwischen Unternehmen und Unternehmen.

b2c (*Business-to-Customer*) Elektronischer Handel über das Internet zwischen Unternehmen und Verbrauchern.

Cache(Local) Lokaler Speicher eines Rechners, der von einem Programm verwaltet wird, mit dem Ziel, Antwortzeiten und Speicherbelastung bei gleicher Anfrage zu reduzieren.

Client Ein Client nimmt Dienste in Anspruch, die von einem Server zur Verfügung gestellt werden.

Content Redaktioneller Inhalt einer Website.

Datenbank Mit einer Datenbank ist eine Sammlung von Daten gemeint, die miteinander in Beziehung stehen (Relation, Relationale Datenbank). Über Datenbanken werden verschiedenartige Datensätze beispielsweise Aufträge, Kundenadressen, Bilder oder Archivinformationen verwaltet. Dazu werden spezifische Informationen in Tabellen zusammengefaßt, die wiederum aus einzelnen Feldern bestehen.

Datenkonvertierung Texte, Grafiken u.a. werden in bestimmten Datenformaten gespeichert. Um mit „fremden“ Daten umgehen zu können, müssen diese dem eigenen Format angepaßt werden - also durch Übersetzung konvertiert werden.

Digitale Signatur Mechanismus zur Gewährleistung der Authentizität einer Nachricht durch Ergänzung um einen kryptographischen Code. Wird bei Verwendung von Verfahren der asymmetrischen Verschlüsselung mit dem privaten Schlüssel erstellt. Digitale Signaturen bilden eine Untergruppe der elektronischen Unterschriften.

DNS (*Domain Name System*) Auf dem Transportprotokoll UDP aufbauendes Internet Protokoll, das die Abfrage von und die Kommunikation zwischen DNS-Servern definiert. Ziel des DNS ist die Zuordnung von Internet-Adressen zu DNS-Rechnernamen und umgekehrt. Siehe auch DNS-Spoofing.

Domain-Name Die komplette eigene Namens-Adresse im Internet, z.B. ti.fhg.de Die Buchstaben hinter dem letzten Punkt sind die TLD (Top Level Domain), die davor die SLD (Second Level Domain).

ECCC (*Electronic Colloquium on Computational Complexity*)– Research reports, surveys and books in computational complexity, ISSN 1433-8092. Ein elektronisches wissenschaftliches Journal.

E-Business Elektronischer Handel über das Internet zwischen Unternehmen und Unternehmen, auch b2b.

E-Commerce Elektronischer Handel über das Internet zwischen Unternehmen und Kunden, auch b2c.

Editor Ein Editor ist ein Programm, mit dem Texte und Dateien erstellt, ergänzt oder stellenweise gelöscht werden können.

Firewall Eine Hard- und Software-Kombination, die einen Rechner oder ein lokales Netzwerk vor einem unbefugten externen Eindringen, z.B. aus dem Internet, schützt.

Frame Programmierelement für HTML-Seiten, das die aufgebaute Seite in mehrere fest definierte Bereiche aufteilt. Damit bleiben bestimmte Elemente auch dann sichtbar, wenn der Nutzer z.B. über einen Hyperlink ein anderes Seitenelement aufruft.

Frameset Seitenlayout eines HTML-Dokuments, das sich aus mehreren Frames zusammensetzt.

Gateway Ein Gateway ist die Schnittstelle eines internen Netzes zu anderen Netzen (Internet), oder zwischen Internet-Netzen wie AOL und T-Online, um die unterschiedlichen z.T. proprietären Protokolle zu übersetzen.

GIF (*Graphics Interchange Format*) Von CompuServe entwickeltes Grafikformat, das die Datengröße von Bildern komprimiert.

GUI *Graphical User Interface* Technische Bezeichnung der grafischen Benutzeroberfläche von z.B. WINDOWS oder OS/2.

Homepage Erster Anlauf- und Einstiegspunkt (Hauptseite) eines Online-Angebots. Die Homepage ist quasi vergleichbar mit der Titelseite und einem groben Inhaltsverzeichnis einer Zeitschrift.

Host Bezeichnung für einen Rechner ("Gastgeber"), der über eine TCP/IP-Adresse an das Internet angeschlossen ist und für andere Rechner zugänglich ist.

HTML (*HyperText Markup Language*) Seitenbeschreibungssprache für Internet-Seiten.

HTTP (*HyperText Transmission Protocol*) Datenübertragungsprotokoll im Internet für HTML-Dokumente.

Hyperlink (Link) Hervorgehobene Textstellen oder Grafiken in WWW-Dokumenten, die per Maus angeklickt werden können. Dadurch springt der Nutzer zu anderen Stellen in der Website oder zu externen Seiten/Angeboten.

IETF (*Internet Engineering Task Force*), Organisation die sich mit der technischen Weiterentwicklung des Internets beschäftigt und Empfehlungen abgibt.

Interface Schnittstelle zwischen Programmaufruf und Programmcode.

Internet-Provider (*Internet-/Service-Provider*, ISP) Ein Anbieter von technischen Internet-Leistungen, der Zugänge zum physikalischen Netzwerk bereitstellt.

Intranet Firmeninternes Netzwerk, das technisch genau gleich aufgebaut ist wie ein Internet, auf das aber nur von innerhalb der Firma zugegriffen werden kann.

IP (*Internet Protocol*) Das Übertragungsprotokoll definiert die Regeln und Vereinbarungen, die den Informationsfluss in einem Kommunikationssystem steuern. Hauptaufgabe des IP ist die netzübergreifende Adressierung. Das Protokoll arbeitet nicht leitungs-, sondern paketvermittelt: Sogenannte Datagramme suchen sich über die jeweils verfügbaren Verbindungen ihren Weg zum Empfänger

IP-Adresse Adresse, die im Rahmen des zentralen Vermittlungsprotokolls im Internet (Internet Protocol) jedem Internetzugang zugewiesen wird. Eine Internet-Adresse besteht aus 4 Byte, die durch Punkte getrennt sind, z.B. 153.96.230.40.

IVW *Informationsgemeinschaft zur Festlegung der Verbreitung von Werbeträgern e.V.* (www.ivw.de) Die IVW befaßt sich unter anderem mit der Messung der Werbeträgerleistung eines INTERNET-Angebotes.

Java Java ist eine objektorientierte und plattformunabhängige Programmiersprache der Firma *SUN* Microsystems. Java-Programme basieren nicht auf Maschinencode, sondern auf einem speziellen Bytecode, der in einer „Virtual Machine“ (VM) ausgeführt wird. Damit ein Java-Programm ausgeführt werden kann, muss auf dem Betriebssystem eine VM verfügbar sein.

JPEG (*Joint Photographic Expert Group*) Grafikformat, das ähnlich wie GIF im WorldWideWeb verwendet wird, aber eine höhere Kompression aufweist und qualitativ hochwertigere Grafiken erlaubt.

LDAP (*Lightweight Directory Access Protocol*) Auf dem Transportprotokoll TCP aufbauendes Internet-Protokoll für den Zugriff auf Directory-Server. LDAP ist dem OSI-Protokoll „DAP“ nachempfunden, ist jedoch wegen der geringeren Komplexität (“Light“) sowie der Verkodierung aller Daten in der Form von Zeichenketten weniger rechenintensiv. Das hierarchische Informationsmodell des X.500-Directory und die Identifizierung von Einträgen mit Domain-Namen kommen unverändert zum Einsatz.

Logfile Auf einem Server gelagerte Datei, die alle Zugriffe und abgerufenen Elemente eines Online-Angebots aufzeichnet.

MIME (*Multipurpose Internet Mail Extensions*) Verfahren zur Spezifikation unterschiedlicher (auch zusammengesetzter) Dokumenttypen und Verkodierungsverfahren, das für den Einsatz mit E-Mail (siehe auch SMTP) entwickelt wurde.

Multimedia Wörtlich: ”viele Medien”. Darstellung, die mehrere Medien einsetzt. In den letzten Jahren ist die Bedeutung des Begriffs eingeengt worden auf Computeranwendungen die nicht nur aus Text bestehen, sondern auch Musik, Bilder und Videos enthalten.

MySQL MySQL ist ein relationales Datenbank Managementsystem (rdbms), welches von der Firma T.c.X. DataKonsult in Schweden entwickelt wurde. MySQL unterstützt User mit einer leistungsstarken Multi-User, Multi-threaded SQL (Structured Query Language) Datenbanklösung, welche schnell, robust und einfach im Gebrauch ist.

Objekt Im Gegensatz zu prozeduralen Programmiersprachen, in denen Funktionen mit Parametern aufgerufen werden, existieren in objektorientierten Programmiersprachen Objekte, die Eigenschaften und Methoden besitzen. Objekte können Methoden und Eigenschaften von anderen Objekten erben, so dass eine Objekt-Hierarchie aufgebaut werden kann.

Offline Zeitversetzt – Es besteht keine Verbindung zu einem Netzwerk, eine zeitgleiche, interaktive Kommunikation und Informationsübermittlung mit anderen Rechnern ist nicht möglich.

Online Zeitgleich – Eine stehende Verbindung zu einem Netzwerk wird unterhalten, eine zeitgleiche, interaktive Kommunikation und Informationsübermittlung mit anderen Rechnern ist möglich.

Open Source Der Programmierer Eric Raymond prägte diesen Begriff für frei verfügbare und veränderbare Betriebssysteme und Programme wie Linux.

Persistenz Persistenz bezeichnet die Möglichkeit, verwendete Daten dauerhaft, d.h. über die Verwendung innerhalb eines Programms hinaus, zu speichern.

Pixel Bildpunkte, aus denen ein gerastertes Bild zusammengesetzt wird.

Port Bei der Internet-Kommunikation über TCP/IP wird neben der IP-Adresse auch eine Portnummer benötigt, die den Dienst des angesprochenen Rechners definiert.

PKIX (*Public Key Infrastructure X.509*) Arbeitsgruppe der IETF, die Standards für den einheitlichen Einsatz von X.509-Zertikaten und der X.509v3-Erweiterungsfelder im Internet entwickelt.

POP3 (*Post Office Protocol, Version 3*) Auf dem Transportprotokoll TCP aufbauendes Internet-Protokoll, das zum Abholen von E-Mail-Nachrichten von einem Mail-Server dient. Der Benutzer wird durch die Angabe von Benutzername und Kennwort authentifiziert. Das Versenden erfolgt oft komplementär über das SMTP.

Proxy(-Server) Proxy-Server speichern Daten, die von Nutzern besonders häufig abgerufen werden, zwischen. Die Strecke, die die Daten zum anfragenden Rechner transportieren, verringert sich u. U. und entlastet das Netzwerk.

Remote Object Ein auf einem „entfernten“ Rechner zum Aufruf bereites Objekt.

RFC (*Request For Comment*) Vom Internet Architecture Board herausgegebene Norm. Zentrales Dokument im Standardisierungsprozess der Internet-Technik. Der Status eines RFCs (von Informational über Proposed Standard bis Internet Standard) gibt Aufschluss über den Stand des Textes.

RMI (*Remote Methode Invocation*) Möglichkeit von Java auf entfernte Objekte zugreifen zu können.

RPC (*Remote Procedure Call*) Aufruf einer Funktion in einem anderen Adressraum, insbesondere einem anderen Rechner.

Server Ein Computer oder eine Software zur Bereitstellung von Informationen für alle anderen Netzwerkzugänge (z.B. Webserver).

SGML (*Standard Generalized Markup Language*) Ein ISO-Standard für eine Hypertextsprache zur Beschreibung von Dokumentenstrukturen, aus der die erweiterte HTML hervorging.

S-HTTP (*Secure-HTTP*) Erweiterung von HTML und HTTP, welche die Übertragung von verschlüsselten und/oder digital unterschriebenen MIME-Entitäten in beide Richtungen vorsieht. Diese Absicherung von HTTP auf der Anwendungsebene erlaubt den applikationsspezifischen Einsatz kryptographischer Verfahren.

S/MIME Auf dem MIME-Standard aufbauende Erweiterung zur Übertragung von verschlüsselten und/oder digital signierten E-Mail-Nachrichten über das Internet-Protokoll SMTP. Zur Authentifizierung der Kommunikationsparteien sind X.509-Zertifikate vorgesehen.

SMTP (*Simple Mail Transfer Protocol*) Auf dem Transportprotokoll TCP aufbauendes Internet-Protokoll, das zur Übermittlung von E-Mail dient. Eine Authentifizierung des Benutzers wird nicht durchgeführt.

SQL *Structured Query Language* In den 70er Jahren des 20. Jahrhunderts von der Firma IBM entwickelte Abfragesprache für die relationale Datenbank DB2. Es handelte sich dabei um eine nichtprozedurale (Programmier-)Sprache, die weder Schleifen, Unterprogramme noch Funktionen enthielt.

SSL (*Secure Sockets Layer*) Von der Firma Netscape entwickeltes Verfahren zur Absicherung der Internet-Protokolle auf der Transportschicht. Auf diese Weise können einzelne Anwendungsdienste (wie etwa HTTP oder LDAP) ohne Modifikation um symmetrische Verschlüsselung der Nutzlast und Authentifizierung der Kommunikationspartner mit Hilfe von X.509-Zertifikaten ergänzt werden.

Suchmaschine (Search Engine) Anbieter, die Inhalte und Websites in Datenbanken sammeln und nach Themen sortieren. Diese können nach Eingabe der Nutzer abgefragt werden und erleichtern die Suche nach spezifischen Themen. Suchmaschinen sind z. B. *Google*, *Yahoo!*, *AltaVista*.

Surfen Navigation durch das World Wide Web. Wenn der Benutzer mit der Maus auf einen Link klickt, wird die dazugehörige Website geladen und dargestellt. So kann man mit Mausklick von einer Seite zur nächsten hüpfen, wie ein Surfer, der von einem Wellenkamm auf den nächsten springt.

TCP (*Transmission Control Protocol*) Verbindungsorientiertes Kommunikationssteuerungsprotokoll auf Transport-Ebene aus der Familie der Internet-Protokolle. Übernimmt die Sequenzierung der einzelnen Segmente, die Korrektur etwaiger Übertragungsfehler und die Durchführung der Flusskontrolle.

TCP/IP Bezeichnung für die Familie der Internet-Protokolle, die auf die zentrale Bedeutung der Protokollstandards TCP und (Internet-Protokoll) abstellt.

URI (*Uniform Resource Identifier*) Die einheitliche Menge aller Namen/Adressen die als kurze Textzeilen auf Ressourcen im Internet verweisen.

URL (*Uniform Resource Locator*) Eine informelle Bezeichnung (in technischen Spezifikationen nicht mehr verwendet), die mit populären URI-Schemen assoziiert ist: http, ftp, mailto etc.

URN (*Uniform Resource Name*) Eine URI die über ein institutionelles Versprechenen von Persistenz, Verfügbarkeit etc. verfügt.

VM (*Virtual Machine*) Laufzeitumgebung, insbesondere für ein Java Programm. Wird für jedes Betriebssystem benötigt, auf dem ein Java-Programm laufen soll.

XHTML (*eXtensible Hypertext Markup Language*) Eine relativ junge Auszeichnungssprache für das World Wide Web. Das „X“ steht für „Extensible“ und weist zugleich auf die entscheidende Neuerung hin: XHTML ist erweiterbar, da die Sprache auf XML basiert.

XML (*eXtensible Markup Language*) Dokumentenaustauschformat im Internet.

WAP (*Wireless Application Protocol*) Bei WAP handelt es sich um einen Standard, mit dem Internetinhalte und andere Services auf digitale Mobiltelefone und andere schnurlose Geräte übertragen werden. WAP greift teilweise auf die Extensible Markup Language (XML) für die Strukturierung der Inhalte sowie das Internet-Protokoll für die Übertragung der Daten zurück.

Web Engl.: "Netz", synonym verwendet mit WWW als durch die Hyperlinks aufgespanntes Netz.

Web-Browser Programm zum Anzeigen und Verarbeiten von HTML-Seiten und mit Hilfe von Plug-Ins darin enthaltenen Inhalten (Bilder oder andere Multimediadaten).

WebDAV (*Web-Distributing, Authoring and Versioning*) Mit dieser von der IETF Ende 1998 vorgeschlagene Technik soll sich das Publizieren von Web-Sites vereinfachen. WebDAV besteht aus HTTP-Erweiterungen, die einen Standard für den Datenaustausch zwischen Web-Authoring-Tools und Webservern festlegt.

Webmaster Die für die Einrichtung und den Unterhalt einer Website verantwortliche Person.

Website Hierarchisch aufgebautes WWW-Angebot, bestehend aus mehreren Seiten mit Hyperlinks innerhalb und außerhalb des Angebotes – Synonym: Online-/Web-Angebot, Web-Präsenz.

WEP Das Zentrum für *Wissenschaftliches Elektronisches Publizieren* der Universität Trier ist eine fachübergreifende Einrichtung, die als Informationsbasis und Koordinationszentrum für Aktivitäten auf dem Gebiet des wissenschaftlichen elektronischen Publizierens dienen soll.

WWW (*World Wide Web, W3*) Anwendung im Internet zum leichten Auffinden von speziellen Informationen, die gleichzeitig die Vernetzung aller Informationen zusammen mit einer ansprechenden grafischen Aufbereitung ermöglicht.

WWW-Seite Dokument im World Wide Web, meist als Seite innerhalb einer Website.

W3C (*Word Wide Web Consortium*) Von verschiedenen mit dem INTERNET eng verbundenen Firmen und Konzernen gegründeter Interessenverband unter der Leitung des Laboratory for Computer Science am Massachusetts Institute of Technology in Cambridge, Massachusetts. Das Konsortium fördert Standards und die Interoperabilität von World Wide Web-Produkten. Ursprünglicher Sitz des Konsortiums war die Europäische Organisation für Kernforschung (CERN) in Genf, dort, wo die Technologie des World Wide Webs entwickelt wurde.

Zertifikat (Digital-ID, digitaler Personalausweis) Durch die digitale Signatur des Ausstellers hergestellte Bindung eines öffentlichen Schlüssels an eine bestimmte Datei. Im Fall der im Internet verbreiteten X.509-Zertifikate handelt es sich dabei um die Identität einer Person oder eines Server-Dienstes (Identitätszertifikat). In diesem Fall übernimmt die ausstellende Zertifizierungsstelle die Überprüfung der Identität.

Anhang A

Die Java-Pakete und -Klassen des Online-Redaktionssystem JDaphne

A.1 Die Java-Pakete von JDaphne

crypto	Klassen für die Bereitstellung eines digitalen Signaturmechanismus innerhalb von JDaphne.
crypto.service	Für die digitalen Signaturen genutzte Serviceklassen.
ti.daphne	Die Basis Java-Klassen, neben zentralen Objekten vor allem Klassen für die clientseitige Nutzung.
ti.daphne.display.task	Für darstellbaren Task von JDaphne (remote).
ti.documentObjects	Von den Streaming Readern genutzte Objekte für die Kapselung von Daten.
ti.modules	Testprogramme für die HLM-Komponente.
ti.reader	Streaming Filter für die HLM-Komponente und JDaphne.
ti.remote.document	Die Java-Klassen für den remote zugreifbaren Dokumenten-Service.
ti.remote.file	Die Java-Klassen für den remote Datentransfer auf den Web-Server.
ti.remote.hlm.mdm	Die Basisobjekte der HLM-Komponente.
ti.remote.hlm.mdm.hdp	Die für den HyperDataPool definierten Interfaces.
ti.remote.hlm.mdm.hdp.manager	Die im HyperDataPool implementierten Managerklassen mit ihren Datenobjekten.
ti.remote.hlm.mdm.hdpdb	Die datenbankbasierte Implementation des HyperDataPools.
ti.remote.hlm.mdm.hdpdb.linkmanager	Die datenbankbasierte Implementation des LinkManagers, einer Managerklasse für die Verwaltung von Hyperlinks.

ti.remote.hlm.mdm.hdpdb.metamanager	Die datenbankbasierte Implementation des MetaManagers, einer Managerklasse für die Verwaltung von Metadaten zu den Dokumenten.
ti.remote.hlm.mdm.hdpdb.nativedb	Erweiterungen des Datenstores durch datenbankspezifische Erweiterungen, die von den allgemeinen Klassen eingebunden werden.
ti.remote.hlm.mdm.hdpdb.ressortmanager	Die datenbankbasierte Implementation des RessortManagers, einer Managerklasse für die Verwaltung von Ressortinformationen.
ti.remote.hlm.mdm.hdpdb.similaritymanager	Die datenbankbasierte Implementation des SimilarityManagers,einer Managerklasse für die Ermittlung von Dokumentähnlichkeiten.
ti.remote.hlm.mdm.hlmgui	Die Klassen für die Standalone-Implementation der HLM-Komponente mit GUI.
ti.remote.hlm.mdm.main	Die ausführbaren Klassen, die die Module initialisieren.
ti.remote.hlm.mdm.mdp	Die für die MultiDataPool-Implementation vorgegebenen Interfaces.
ti.remote.hlm.mdm.mdpfilesystem	Die dateisystembasierte Implementation des MDP-Interfaces mit ihren Datenobjekten.
ti.remote.hlm.mdm.modul	Die Definition eines Moduls innerhalb der HLM-Komponente.
ti.remote.hlm.mdm.modul.hlm	Die Implementation der HyperlinkManagement-Komponente als Modul.
ti.remote.hlm.mdm.modul.hyperlinkscanner	Alle Klassen des Hyperlink-Scanner-Moduls, welches eine bestehende Web-Präsenz einlesen kann.
ti.remote.hlm.utils	Für die HLM-Komponente genutzte Hilfsobjekte.
ti.remote.hlm.utils.base64	Von der HLM-Komponente genutzte Klassen zur Base64 Codierung.
ti.remote.mediator	Die Java-Klassen für den remote zugreifbaren MediatorDienst, der Datenquellen in die Web-Präsenz einbindet.

A.2 Die Java-Klassen des Pakets ti.daphne in der Übersicht

AboutDialog	Ein Info-Dialog mit einem OK-Button.
AccessController	Der AccessController entscheidet darüber ob ein Benutzer die Berechtigung für eine Aktion hat.
ActionInfoDialog	Der ActionInfoDialog stellt dem Benutzer den Status der gerade abgearbeiteten Tasks dar.
ActionModul	Diese Klasse beinhaltet die Methoden, mit denen Aktionen an Dokumenten und Ressorts durchgeführt werden können.
AttentionDialog	Der Attention-Dialog liefert dem Benutzer eine Rückmeldung zu einer Aktion.
BasicModul	Das BasicModul kapselt die für die Initialisierung der Display-Komponente notwendigen Funktionen an und handelt das Login über einen Dialog.
BearbeitenDialog	Der BearbeitenDialog stellt eine Übersicht der Dokumente dar, die der Benutzer gerade auf seinem lokalen Arbeitsplatz bearbeitet, und ermöglicht deren Transfer zurück auf den Server.
BemerkungsFeld	Ein Dialog mit dem der Benutzer zur Eingabe einer Bemerkung aufgefordert wird.
BrokenLinkInformation	Ein Kapselobjekt für HyperlinkInformationen (eingehende und ausgehende) zu den Dokumenten.
BrokenLinksInfoDialog	Der Dialog der die HyperlinkInformationen zu einem Dokument in verschiedenen Panels durch Tabellen darstellt.
DaphneAction	Diese Klasse liegt jeder Aktion die von JDaphne ausgeführt wird zugrunde, sie kapselt die Art der Aktion, das betroffene Objekt und den Benutzer, der sie angestossen hat.
Database	Die zentrale Klasse mit der JDaphne Daten in der relationalen Datenbank ablegt und von dort abfragt.
DCMetaData	Ein Objekt, das sämtliche DublinCore Metadaten zu einem Dokument kapselt.
DetailInfoFeld	Ein Dialog für detailierte Informationen zu einem Dokument.
DocEditor	Ein simpler Editor in Form eines Applets.
Document	Die Klasse die alle für das Handling eines Dokumentes notwendigen Informationen kapselt.
DocumentFormat	Die zu jedem Dokument gehörige FormatInformation, die beispielsweise über Suffix und Export-Verfahren entscheidet.
DocumentInformation	Ein das Dokument erweiterndes Informationsobjekt genutzt für digitale Signaturen.

DocumentManager	Die von dem Frontend genutzte Klasse, die die Benutzeraktionen umsetzt.
DocumentPropertiesDialog	Ein Dialog zur Darstellung von übergreifenden Dokumentinformationen.
EventListenerList	A class that holds a list of EventListeners.
EXMGR	Der default Exception Handler der Displaykomponente.
GuiFreeAccessController	Die dem mit Dialogen versehenen AccessController zugrunde liegende Komponente ohne GUI.
GuiFreeBasicModul	Die dem mit Dialogen versehenen BasicModul zugrunde liegende Komponente ohne GUI.
HtmlEditor	Ein simpler HTML-Editor für Dokumente von JDaphne, er bietet nur rudimentäre Funktionen.
JDaphneExplorerFrame	Das interaktive Frontend von JDaphne in Form eines Dateimanagers.
JdaphneFileFilter	Der von JDaphne genutzte Dateifilter.
JDaphneLinkTableColumnModel	Ein eigenes TableColumnModel für die farbige Darstellung von Dokumenten mit Broken-Links.
JDaphneProgressBar	Eine Progressbar zur Anzeige des Erfolgsstandes remote ausgeführter Aktionen.
JDaphneTableCellRenderer	Ein eigener CellRenderer zur farbigen Darstellung von Tabellenzellen.
JDaphneTableColumnModel	Ein eigenes Tabellenmodell zur Erstellung eigener Ansichten.
JDaphneTableMap	Eine Umleitung von Tabellenevents an die entsprechenden Listener.
JDaphneTableModel	Ein eigenes TabellenModell zur Darstellung von JDaphne spezifischen Daten.
JDaphneTableSorter	Eine TableMap, die das Sortieren der Tabellenspalten ermöglicht.
JDaphneTaskTableColumnModel	Ein eigenes Tabellenspaltenmodell für die Darstellung von Task in JDaphne.
JDaphneTreeCellRenderer	Eine eigene Konfiguration für die Verzeichnisobjekte in der Baumdarstellung des Dateimanagers.
JUserEditor	Ein Editor für die Bearbeitung von Benutzerdaten.
JUserManager	Eine Übersicht über die Benutzer JDaphnes.
LinkDeActivaionInformation	Die zu den Hyperlinks verwalteten Informationen über den (De)-Aktivierungsstatus.
Lock	Eine Bearbeitungssprerre für ein Dokument, die gesetzt wird, wenn ein Benutzer es bearbeitet.
LoginDialog	Der Dialog für die Anmeldung an dem System.
Message	Ein Objekt das die Rückmeldung von remote ausgeführten Aktionen kapselt.

Ressort	Das der Web-Präsenz struktur gebende Element in JDaphne, es kann mit einem Verzeichnis verglichen werden.
RessortDialog	Ein Dialog mit dem ein Ressort angelegt wird.
RessortEigenschaftDialog	Ein Dialog mit dem die Eigenschaften eines Ressorts bearbeitet werden können.
RessortSprachDaten	Für die multilinguale Verwaltung benötigte Informationen werden von diesem Objekt gekapselt.
RMGR	Das Applet, welches den JDaphneExplorer als Frontend startet und das Actionmodul initialisiert.
Rolle	Das Objekt das die Rolleneigenschaften kapselt.
SignaturInfo	Objekt zum Kapseln der digitalen Signaturen.
SignierenDialog	Der Dialog der den Signaturvorgang handelt.
StatistikDialog	Ein Dialog der die Zugriffszahlen aus dem WWW auf einzelne Dokumente visualisiert.
SuchenDialog	Eine Suchmaske die die Suche im Dokumentenbestand von JDaphne ermöglicht.
Task	Eine Task kapselt ToDos innerhalb von JDaphne, sie enthält Datums sowie Zuteilsungswerte und spezifiziert die betroffenen Objekte.
TemplateChooser	Ein Auswahldialog zur Selektion eines Templates, mit dem der Benutzer ein neues Dokument erstellen möchte.
ToDoInfoFrame	Die Visualisierung der ToDo-Liste, sie stellt die für die Benutzer angefallenen Tasks dar.
UMGR	Das Applet, das den Usermanager initialisiert und das GUI erstellt.
User	Das Benutzerobjekt das Benutzer in JDaphne repräsentiert.
WaitDialog	Ein Dialog mit der Meldung „bitte Warten“, der eingeblendet wird während eine Aufgabe initialisiert wird.
WirklichJDialog	Ein Dialog, der den Benutzer um eine Bestätigung für Aktionen bittet.

Anhang B

Java-Pakete und -Klassen der Remote-Komponenten von JDaphne

B.1 Package ti.remote.file

Package Contents *Page*

Interfaces

Classes

B.1.1 Interfaces

B.1.1.1 Interface **FileStoreServiceInterface**

Dieses Inferface definiert die Methoden die der Dateitransfer-Dienst bieten soll. * @author Heuer

Declaration

public interface FileStoreServiceInterface
implements java.rmi.Remote

Methods

- *buildEntschuldigungsDeckdokument*
  `public void` **buildEntschuldigungsDeckdokument**`( java.io.File` **f**`, java.lang.String` **sprache** `)`
  - **Usage**
    - Generiert eine Entschuldigungs-Seite wenn Für ein publiziertes Ressort ein Deckdokument entfernt wird.
  - **Parameters**
    - `f` - Die Datei die geschrieben werden soll.
    - `sprache` - Die Sprache in der das Dokument angelegt werden soll.

- *copyFile*
  `public boolean` **copyFile**`( java.io.File` **file_alt**`, java.io.File` **file_neu** `)`
  - **Usage**
    - Kopiert eine Datei.
  - **Parameters**
    - `file_alt` - Alte Datei.
    - `file_neu` - Neue Datei.
  - **Returns** - True, wenn Kopie angelegt werden konnte.

- *existsFileInInternet*
  `public boolean` **existsFileInInternet**`( java.io.File` **datei**`, java.lang.String` **sprache** `)`
  - **Usage**
    - Diese Methode liefert True, wenn die gesuchte Datei in der entsprechenden Sprache auf dem Web-Server vorliegt.
  - **Parameters**
    - `datei` - Die gesuchte Datei.
    - `sprache` - Die Sprache, in der die Datei vorliegen soll.
  - **Returns** - True, wenn die Datei vorhanden ist.

- *moveFile*
  `public boolean` **moveFile**`( java.io.File` **file_alt**`, java.io.File` **file_neu** `)`
  - **Usage**
    - Verschiebt eine Datei.
  - **Parameters**
    - `file_alt` - Alte Datei.
    - `file_neu` - Neue Datei.
  - **Returns** - True, wenn Verschiebevorgang erfolgreich durchgeführt werden konnte.

- *removeFile*
  `public boolean` **removeFile**`( java.io.File` **file** `)`
  - **Usage**
    - Löscht die angegebene Datei aus dem Filesystem.
  - **Parameters**
    - `file` - Die Datei, die gelöscht werden soll.
  - **Returns** - True, wenn Löschen erfolgreich.

- *restoreBinary*
  `public byte` **restoreBinary**`( java.io.File` **file** `)`
  - **Usage**
    - Liest eine Datei aus dem Dateisystem in Form von Bytes.
  - **Parameters**
    - `file` - Die zu lesende Datei.
  - **Returns** - Ein Byte-Array mit dem Inhalt der Datei.

- *restoreTextual*
  `public char` **restoreTextual**`( java.io.File` **f** `)`
  - **Usage**
    - Liest eine Datei aus dem Dateisystem in Form von Characters.
  - **Parameters**
    - `file` - Die zu lesende Datei.
  - **Returns** - Ein char-Array mit dem Inhalt der Datei.

- *store*
  `public boolean` **store**`( byte [] data, java.io.File` **file**`, boolean` **overwrite** `)`
  - **Usage**
    - Speichert die übergebene Datei als Bytes im Dateisystem.
  - **Parameters**
    - `data` - Das Byte-Array mit dem Inhalt der Datei.
    - `file` - Die Datei.
    - `overwrite` - True setzen, wenn eine bereits bestehende Datei überschrieben werden soll.

  - **Returns** - True, wenn Vorgang erfolgreich.

- *store*
  `public boolean` **store**`( char [] data, java.io.File file, boolean overwrite )`
  - **Usage**
    - Speichert die übergebene Datei als Characters im Dateisystem.
  - **Parameters**
    - `data` - Das Character-Array mit dem Inhalt der Datei.
    - `file` - Die Datei.
    - `overwrite` - True setzen, wenn eine bereits bestehende Datei überschrieben werden soll.
  - **Returns** - True, wenn Vorgang erfolgreich.

B.1.2 Classes

B.1.2.1 Class **FileStore**

Eine RMI-Server-Komponente, die einen Datei-Transferdienst bereitstellt.

Declaration

public class FileStore
extends java.lang.Object

Constructors

- *FileStore*
  `public` **FileStore**`( )`

Methods

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    - Die ausführbare Methode des Servers, die den Dienst initalisiert und an die RMI-Registry bindet.
  - **Parameters**
    - `Zum` - Start des Servers von Nöten sind folgende Parameter:

      -Djava.security.policy=FullAccess.txt ti.remote.file.FileStore properties-file

B.1.2.2 Class **FileStoreService**

Der Datei-Transfer-Dienst, wie er z.B. auf dem Web-Server installiert werden kann, um Dateien aus dem Redaktionssystem zu publizieren.

Declaration

public class FileStoreService
extends java.rmi.server.UnicastRemoteObject
implements FileStoreServiceInterface

Constructors

- *FileStoreService*
  `public` **FileStoreService**`( java.lang.String` **filesystem** `)`
  - **Usage**
    - Der Konstruktor dieser Klasse muss mit dem Dateisystem, d.h. dem Basispfad auf dem Server-Rechner initialisiert werden.
  - **Parameters**
    - `filesystem` - der Pfad auf dem Server, in dem die Dateien mit einem relativen Pfad abgelegt werden sollen.

Methods

- *buildEntschuldigungsDeckdokument*
  `public void` **buildEntschuldigungsDeckdokument**`( java.io.File` **f**`, java.lang.String` **sprache** `)`
  - **Usage**
    - Generiert eine Entschuldigungs-Seite wenn Für ein publiziertes Ressort ein Deckdokument entfernt wird.
  - **Parameters**
    - `f` - Die Datei die geschrieben werden soll.
    - `sprache` - Die Sprache in der das Dokument angelegt werden soll.

- *copyFile*
  `public boolean` **copyFile**`( java.io.File` **file_alt**`, java.io.File` **file_neu** `)`
  - **Usage**
    - Kopiert eine Datei.
  - **Parameters**
    - `file_alt` - Alte Datei.
    - `file_neu` - Neue Datei.
  - **Returns** - True, wenn Kopie angelegt werden konnte.

- *existsFileInInternet*
  `public boolean` **existsFileInInternet**`( java.io.File` **datei**`, java.lang.String` **sprache** `)`
  - **Usage**

    * Diese Methode liefert True, wenn die gesuchte Datei in der entsprechenden Sprache auf dem Web-Server vorliegt.
  - **Parameters**
    * `datei` - Die gesuchte Datei.
    * `sprache` - Die Sprache, in der die Datei vorliegen soll.
  - **Returns** - True, wenn die Datei vorhanden ist.

- *moveFile*
  `public boolean` **moveFile**`( java.io.File` **file_alt**`, java.io.File` **file_neu** `)`
  - **Usage**
    * Verschiebt eine Datei.
  - **Parameters**
    * `file_alt` - Alte Datei.
    * `file_neu` - Neue Datei.
  - **Returns** - True, wenn Verschiebevorgang erfolgreich durchgeführt werden konnte.

- *removeFile*
  `public boolean` **removeFile**`( java.io.File` **file** `)`
  - **Usage**
    * Löscht die angegebene Datei aus dem Filesystem.
  - **Parameters**
    * `file` - Die Datei, die gelöscht werden soll.
  - **Returns** - True, wenn Löschen erfolgreich.

- *restoreBinary*
  `public byte` **restoreBinary**`( java.io.File` **f** `)`
  - **Usage**
    * Liest eine Datei aus dem Dateisystem in Form von Bytes.
  - **Parameters**
    * `f` - Die zu lesende Datei.
  - **Returns** - Ein Byte-Array mit dem Inhalt der Datei.

- *restoreTextual*
  `public char` **restoreTextual**`( java.io.File` **f** `)`
  - **Usage**
    * Liest eine Datei aus dem Dateisystem in Form von Characters.
  - **Parameters**
    * `f` - Die zu lesende Datei.
  - **Returns** - Ein char-Array mit dem Inhalt der Datei.

- *store*
  `public boolean` **store**`( byte [] data, java.io.File` **f**`, boolean` **overwrite** `)`
  - **Usage**

    - Speichert die übergebene Datei als Bytes im Dateisystem.
  - **Parameters**
    - `data` - Das Byte-Array mit dem Inhalt der Datei.
    - `f` - Die Datei.
    - `overwrite` - True setzen, wenn eine bereits bestehende Datei überschrieben werden soll.
  - **Returns** - True, wenn Vorgang erfolgreich.

- *store*
  `public boolean` **store(** `char []` **data,** `java.io.File` **f,** `boolean` **overwrite** **)**
  - **Usage**
    - Speichert die übergebene Datei als Characters im Dateisystem.
  - **Parameters**
    - `data` - Das Character-Array mit dem Inhalt der Datei.
    - `f` - Die Datei.
    - `overwrite` - True setzen, wenn eine bereits bestehende Datei überschrieben werden soll.
  - **Returns** - True, wenn Vorgang erfolgreich.

B.2 Package ti.remote.mediator

Package Contents *Page*

Interfaces

Classes

B.2.1 Interfaces

B.2.1.1 Interface MediatorServiceInterface

Das Interface mit der Definition der remote zugreifbaren Methoden.

Declaration

public interface MediatorServiceInterface
implements java.rmi.Remote

Methods

- *execute*
  `public String` **execute**`( java.lang.String` **parameterblock** `)`
  - **Usage**
    - Diese Methode führt die von dem Mediator verlangte Aufgabe aus, die Abfrage einer Datenbank oder eines HTTP-Dienstes. Welche Aktion durch execute durchgeführt wird, hängt von den übergebenen Parametern (Parameterblock als String) ab.
  - **Parameters**
    - `parameterblock` - Der aus dem Repräsentantendokumt herausgeparste Parameterblock als String.
  - **Returns** - Das Ergebnis der Anfrage als nicht gelayouteter String.

B.2.2 Classes

B.2.2.1 Class Mediator

Der Mediator als solcher ist eine Server-Komponente, die über RMI angesprochen wird, sie bietet die Dienste des MediatorServiceInterfaces an.

Declaration

public class Mediator
extends java.lang.Object

Constructors

- *Mediator*
  `public` **Mediator**`( )`
  - **Usage**
    - Ein leerer Konstruktor.

METHODS

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    - Die ausführbare Methode des Servers, die den Dienst initalisiert und an die RMI-Registry bindet.
  - **Parameters**
    - `String` - der Name und Pfad der Property-Datei im Dateisystem.

B.2.2.2 CLASS MediatorService

Der MediatorService als dienstleistende Klasse implementiert das MediatorServiceInterface mit der execute-Methode.

DECLARATION

public class MediatorService
extends java.rmi.server.UnicastRemoteObject
implements MediatorServiceInterface

CONSTRUCTORS

- *MediatorService*
  `public` **MediatorService**`( )`
  - **Usage**
    - Ein leerer Konstruktor.

METHODS

- *execute*
  `public String` **execute**`( java.lang.String` **parameterBlock** `)`
  - **Usage**
    - Die execute-Methode übergibt den Parameterblock an ein eigens zu diesem Zweck initialisiertes Query-Objekt, welche die Anfrage auswertet und ausführt. Das Resultat wird an Stelle des Parameterblocks zurückgegeben.
  - **Parameters**
    - `parameterBlock` - Ein String, der sämtliche Parameter die für den Service benötigt werden, enthält.
  - **Returns** - Das Ergebnis der Anfrage.

B.3 Package ti.remote.document

Package Contents *Page*

Interfaces

Classes

B.3.1 Interfaces

B.3.1.1 Interface **DocumentServiceInterface**

Diese Klasse enthält die Interface Definition für alle mit Dokumenten ausführbaren Aktionen, da dieser Dienst über einen Remote-Server bereitgestellt wird, erweitert sie das Remote-Interface.

Declaration

```
public interface DocumentServiceInterface
implements java.rmi.Remote
```

Methods

- *dokumentAbzeichnen*
  `public Message` **dokumentAbzeichnen**`( int` **dokumentid**`, java.lang.String` **username** `)`
  - **Usage**
    - Mit dieser Methode wird ein Dokument von einem Benutzer aus dem Zustand „Zum Abzeichnen“ in den Zustand „Zum Publizieren“ überführt.
  - **Parameters**
    - `dokumentid` - Die interne Id des Dokumentes, das abgezeichnet wird.
    - `username` - Der interne Name des Benutzers, der das Dokument abzeichnet
  - **Returns** - Message Ein Nachrichtenobjekt, dass den Erfolg, bzw. Misserfolg der Aktion dokumentiert

---

- *dokumentAbzeichnenAblehnen*
  `public Message` **dokumentAbzeichnenAblehnen**`( int` **dokumentid**`, java.lang.String` **username**`, java.lang.String` **bemerkung** `)`
  - **Usage**
    - Mit dieser Methode wird der Zustand eines Dokumentes durch einen Benutzer vom Zustand „Zum Abzeichnen“ in den Zustand „Abzeichnen abgelehnt“ überführt; die Begründung für die Ablehnung steht in der Bemerkung.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `bemerkung` - Die Begründung für die Ablehnung.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentBearbeiten*
  `public byte` **dokumentBearbeiten**`( int` **dokumentid** `)`
  - **Usage**
    - Mit dieser Methode wird ein Dokument zum Bearbeiten aus dem internen Dokumentenspeicher gelesen und an den diese Methode aufrufenden Client als Byte-Array übertragen.

- **Parameters**
  - `dokumentid` - Die interne Id des Dokumentes, das bearbeitet werden soll.
- **Returns** - byte[], das Array mit dem Dokument.

- *dokumentErsetzen*
  `public Message` **dokumentErsetzen**`( byte [] data, java.io.File file, java.lang.String username, int dokumentid )`
  - **Usage**
    - Mit dieser Methode wird ein bestehendes Dokument in dem internen Dokumentenspeicher durch ein neues vom Client in Form eines Byte-Array übertrages Dokument ersetzt.
  - **Parameters**
    - `dokumentid` - Die interne Id des Dokumentes, das ersetzt werden soll.
    - `data` - Das Byte-Array mit dem Dokument.
    - `file` - Fileinformationen zu dem Dokument.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentGenerieren*
  `public Message` **dokumentGenerieren**`( int dokumentid, java.lang.String username )`
  - **Usage**
    - Ein bereits publiziertes Dokument wird durch diese Aktion neu generiert.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes, da neu generiert werden soll.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentLinkUebersetzung*
  `public int` **dokumentLinkUebersetzung**`( int dokumentid, java.lang.String username, java.lang.String selektierterZustand )`
  - **Usage**
    - Diese Methode parst die Links in einem Dokument und versucht sie auf interne Ressourcen umzusetzen.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `selektierterZustand` - Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - int[] Array mit der Anzahl der gefundenen und der Broken-Links.

- *dokumentLinkVorschlaege*
  `public int` **dokumentLinkVorschlaege**`( int` **dokumentid**`, java.lang.String` **username**`, java.lang.String` **selektierterZustand** `)`

  - **Usage**
    - Diese Methode parst das übergebene Dokument und erzeugt Link-Vorschläge.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - int[] Liste mit Dokument Ids.

- *dokumentLoeschen*
  `public Message` **dokumentLoeschen**`( int` **dokumentid**`, java.lang.String` **username** `)`

  - **Usage**
    - Der Aufruf dieser Methode löscht ein Dokument aus dem internen Dokumenten-Repository.
  - **Parameters**
    - `dokumentid` - Die interne Kennung des zu löschenden Dokumentes
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentNeu*
  `public int` **dokumentNeu**`( byte [] data, java.io.File` **file**`, java.lang.String` **username**`, java.lang.String` **ressort**`, java.lang.String` **sprache** `)`

  - **Usage**
    - Mit dieser Methode wird ein neues Dokument in das Dokumentenrepository eingebracht.
  - **Parameters**
    - `ressort` - Das Kürzel des betroffenen Ressorts.
    - `sprache` - Das Kürzel der selektierten Sprache.
    - `file` - Fileinformationen zu dem Dokument.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Die Id des neuen Dokumentes

- *dokumentPublikationAblehnen*
  `public Message` **dokumentPublikationAblehnen**`( int` **dokumentid**`, java.lang.String` **username**`, java.lang.String` **bemerkung** `)`

  - **Usage**
    - Diese Methode setzt den Zustand eines Dokumentes von „Zum Publizieren“ nach Publikation abgelehnt ab.
  - **Parameters**

    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
    * `bemerkung` - Die Begründung für die Ablehnung.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentPublikationAufheben*
  `public Message` **dokumentPublikationAufheben**`( int` **dokumentid**`, java.lang.String` **username** `)`
  - **Usage**
    * Entfernt ein bereits publiziertes Dokument wieder aus dem Internt-Repository.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentPublizieren*
  `public Message` **dokumentPublizieren**`( int` **dokumentid**`, java.lang.String` **username** `)`
  - **Usage**
    * Publiziert ein Dokument in das Internet-Repository.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentUebersetzungsKopieAnlegen*
  `public int` **dokumentUebersetzungsKopieAnlegen**`( int` **dokumentid**`, java.lang.String` **username**`, java.lang.String` **sprache** `)`
  - **Usage**
    * Diese Methode kopiert das selektierte Dokument und legt eine Kopie an, für die die Paramter in der selektieren Sprache gesetzt werden.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `sprache` - Das Kürzel der selektierten Sprache.
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Die Id die die Übersetzungskopie bekommen hat.

- *dokumentUebersichtsSeiteErstellen*
  `public int` **dokumentUebersichtsSeiteErstellen**`( java.lang.String` **ressort**`, java.lang.String` **filename**`, java.lang.String` **username**`, java.lang.String` **sprache** `)`
  - **Usage**

* Erstellt ein Dokument, dass Referenzen auf alle anderen Dokumente in diesem Ressort enthält, wobei die Sprachatribute der selektierten Sprache genutzt werden.

– **Parameters**
* `ressort` - Das Kürzel des betroffenen Ressorts.
* `sprache` - Das Kürzel der selektierten Sprache.
* `username` - Der interne Name des die Aktion ausführenden Benutzers.
* `filename` - Der Name, den die neue Datei bekommen soll.

– **Returns** - Die Id der Übersichtsseite.

- *dokumentURLFuerAnzeigeInBrowser*
`public String` **dokumentURLFuerAnzeigeInBrowser(** `int` **dokumentid,** `java.lang.String` **selektierterZustand** **)**

– **Usage**
* Diese Methode liefert eine Url, die für die Voransicht des selektierten Dokumentes in dem entsprechenden Zustand genutzt werden kann.

– **Parameters**
* `dokumentid` - Die Id des betroffenen Dokumentes.
* `selektierterZustand` - Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.

– **Returns** - Die Url als String.

- *dokumentVerschieben*
`public Message` **dokumentVerschieben(** `int` **dokumentid,** `java.lang.String` **neuesRessort,** `java.lang.String` **username** **)**

– **Usage**
* Diese Methode verschiebt im internen Dokumenten-Repository ein Dokument in ein anderes Ressort.

– **Parameters**
* `dokumentid` - Die Id des zu verschiebenden Dokumentes
* `neusRessort` - Das Kürzel des neuen Ressorts.
* `username` - Der interne Name des die Aktion ausführenden Benutzers.

– **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentZumAbzeichnen*
`public Message` **dokumentZumAbzeichnen(** `int` **dokumentid,** `java.lang.String` **username** **)**

– **Usage**
* Ändert den Zustand eines Dokumentes von „In Bearbeitung“ nach „Zum Abzeichnen“.

– **Parameters**
* `dokumentid` - Die Id des betroffenen Dokumentes.
* `username` - Der interne Name des die Aktion ausführenden Benutzers.

– **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentZumBearbeiten*
  `public Message` **dokumentZumBearbeiten**`( int` **dokumentid**`, java.lang.String` **username**`, java.lang.String` **bemerkung** `)`
  - **Usage**
    - Ändert den Zustand eines Dokumentes nach „Zum Bearbeiten“, d.h. Überarbeiten.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `bemerkung` - Die Begründung für die Aufforderung zur Überarbeitung.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *getBrokenLinkInformation*
  `public BrokenLinkInformation` **getBrokenLinkInformation**`( int` **dokumentid**`, java.lang.String` **selektierterZustand** `)`
  - **Usage**
    - Liefert ein Objekt zurück, dass die Hyperlink-Informationen für das selektierte Dokument enthält.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Das Objekt mit den Hyperlink-Informationen.

- *getCompleteFileLocator*
  `public String` **getCompleteFileLocator**`( ti.daphne.Document` **dokument**`, boolean` **exportmode**`, boolean` **link**`, boolean` **setExcusePage** `)`
  - **Usage**
    - Liefert den vollständigen Pfad zu dem Dokument in einem Dateisystem.
  - **Parameters**
    - `dokument` - Das betroffene Dokument.
    - `exportmode` - Der Modus, für das die Zugriffsinformation erstellt werden soll.True für Export auf Web-Präsenz.
    - `link` - True, wenn es sich um einen Link handelt, der gerade zugeordnet werden soll.
    - `setExcusePage` - True, wenn eine Entschuldigungsseite an die Stelle dieses Dokumentes gesetzt werden soll, wenn es nicht publiziert ist.
  - **Returns** - Der vollständige Pfad.

- *getCompleteFileLocator*
  `public String` **getCompleteFileLocator**`( ti.daphne.Ressort` **ressort**`, boolean` **exportmode**`, boolean` **link**`, boolean` **setDummyPage** `)`
  - **Usage**
    - Liefert den vollständigen Pfad zu dem Ressort in einem Dateisystem.
  - **Parameters**

    * `ressort` - Das betroffene Ressort.
    * `exportmode` - Der Modus, für das die Zugriffsinformation erstellt werden soll.True für Export auf Web-Präsenz.
    * `link` - True, wenn es sich um einen Link handelt, der gerade zugeordnet werden soll.
    * `setExcusePage` - True, wenn eine Entschuldigungsseite an die Stelle dieses Dokumentes gesetzt werden soll, wenn es nicht publiziert ist.
  - **Returns** - Der vollständige Pfad.

- *getLinkFromInformation*
  `public BrokenLinkInformation` **getLinkFromInformation**`( int` **dokumentid**`, java.lang.String` **selektierterZustand** `)`
  - **Usage**
    * Liefert ein Objekt mit den Links, die von dem übergebenen Dokument ausgehen.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Das Objekt mit den Hyperlink-Informationen.

- *getLinkToInformation*
  `public BrokenLinkInformation` **getLinkToInformation**`( int` **dokumentid**`, java.lang.String` **selektierterZustand** `)`
  - **Usage**
    * Liefert ein Objekt mit den Links, die auf das übergebene Dokument verweisen.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Das Objekt mit den Hyperlink-Informationen.

- *move*
  `public boolean` **move**`( ti.daphne.Ressort` **ressort**`, ti.daphne.Ressort` **newParent** `)`
  - **Usage**
    * Verschiebt ein Verzeichnis mitsamt Inhalten.
  - **Parameters**
    * `ressort` - Das zu verschiebende Ressort.
    * `newParent` - Das Ressort dem das zu verschiebende Ressort untergeordnet werden soll.
  - **Returns** - boolean True wenn erfolgreich.

- *newDokumentAsTemplate*
  `public Message` **newDokumentAsTemplate**`( int` **dokumentid**`, java.lang.String` **ressort**`, java.lang.String` **username**`, java.lang.String` **sprache** `)`

  - **Usage**
    * Legt ein neues, auf dem selektierten Dokument basierendes Template an.
  - **Parameters**
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
    * `dokumentid` - Die Id des als Vorlage zu nutzenden Dokumentes.
    * `ressort` - Das Kürzel des betroffenen Ressorts.
    * `sprache` - Das Kürzel der selektierten Sprache.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *regenerateDocuments*
  `public Message` **regenerateDocuments**`( int [] dokumentid, java.lang.String  username )`

  - **Usage**
    * Generiert die selektierten, bereits publizierten Dokumente neu für den statischen Export.
  - **Parameters**
    * `dokumentid[]` - Die Ids der betroffenen Dokumentes.
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *remove*
  `public boolean` **remove**`( ti.daphne.Ressort  ressort )`

  - **Usage**
    * Löscht ein Ressort aus dem Repository, das Ressort muss vorher geleert worden sein.
  - **Parameters**
    * `ressort` - Das zu löschende Ressort.
  - **Returns** - boolean True wenn erfolgreich.

- *rename*
  `public boolean` **rename**`( ti.daphne.Ressort  ressort, java.lang.String newName )`

  - **Usage**
    * Benennt ein Ressort im Dokumentenrepository um.
  - **Parameters**
    * `ressort` - Das selektierte Ressort.
    * `newName` - Der neue Name, der für das Ressort vergeben wird.
  - **Returns** - True, wenn erfolgreich.

- *restoreDocuments*
  `public int` **restoreDocuments**`( java.lang.String  username )`

  - **Usage**
    * Diese Methode regeneriert Dokumente aus einer Sicherheitskopie, die beim Einspielen angelegt wird.

  - **Parameters**
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - 0 Wenn Aktion erfolgreich

- *restoreKeywords*
  `public int` **restoreKeywords**`( )`
  - **Usage**
    - Ein Update des Keyword-Bestandes durchführen.

B.3.1.2 Interface ExportServiceInterface

Das Interface mit den Methodendefinitionen für den Export-Service, der das Layouten und exportieren von Dokumenten übernimmt.

Declaration

public interface ExportServiceInterface
implements java.rmi.Remote

Methods

- *buildPage*
  `public String` **buildPage**`( int` **dokumentid**`, byte []` **data**`, int` **rootlevel**`, java.lang.String` **ressKurz**`, boolean` **export**`, java.lang.String` **pathroot**`, java.lang.String` **templateType**`, boolean` **physicalExport** `)`
  - **Usage**
    - Die zentrale Methode des Export-Service baut ein fertiges Dokument aus den einzelnen Bestandteilen, dem Template, den Datenbankinformationen und dem Content zusammen.
  - **Parameters**
    - `dokumentid` - Die Id des zu generierenden Dokumentes.
    - `data` - Das Byte-Array mit dem Dokument-Inhalt.
    - `rootlevel` - Die Ebene ab der die Navigation in die Seite eingebaut wird.
    - `ressKurz` - Das eindeutige Ressortkürzel des Ressorts in das das Dokument exportiert werden soll (wichtig bei Mehrfachnutzung eines Dokumentes).
    - `export` - Auf True setzen, wenn finaler Export, False bei Voransicht.
    - `pathroot` - Falls nicht in die Dokumentroot exportiert wird, an dieser Stelle einen Pfadprefix eintragen.
    - `templateType` - Der Name des Templates als String.
    - `phyiscalExport` - True, wenn statischer Export angestrebt, false bei dynamischem Export.
  - **Returns** - String der die gelayoutete Seite enthält.

- *exportToInternet*
  `public int` **exportToInternet**`( int` **dokumentid**`, byte []` **data**`, int` **rootlevel** `)`

  - **Usage**
    * Diese Methode führt den statischen Export, intern Aufruf von buildPage, in das File-System des Web-Servers durch.
  - **Parameters**
    * `dokumentid` - Die ID des Dokumentes, das exportiert werden soll.
    * `data` - Das Dokument in Form eines Byte-Arrays.
    * `rootlevel` - Die Ebene, ab der die Navigationsleisten initialisiert werden sollen.
  - **Returns** - Error-Code: 0 für Erfolg, 1 für das Auftreten eines Fehlers.

- *init*
  `public Properties` **init(** `java.lang.String` **propertiesURL)**
  - **Usage**
    * Methode, die die Initialisierung des Dienstes mit einer Property-Datei in die Wege leitet.
  - **Parameters**
    * `propertiesURL` - Die URL unter der die Property-Datei geladen werden kann.
  - **Returns** - Das zur Initialisierung verwendete Propertyobjekt.

- *removeFile*
  `public boolean` **removeFile(** `int` **dokumentid,** `int` **rootlevel)**
  - **Usage**
    * Löscht eine Datei vom Web-Server unter Berücksichtigung der Hyperlinks.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `rootlevel` - Die Ebene, ab der die Navigation einsetzt.
  - **Returns** - True wenn die Aktion erfolgreich war.

- *saveToInternet*
  `public int` **saveToInternet(** `int` **dokumentid,** `byte []` **data,** `int` **rootlevel)**
  - **Usage**
    * Diese Methode überträgt die entsprehcende Datei auf den Web-Server, noch ohne Layout, damit dieses dynamisch erzeugt werden kann.
  - **Parameters**
    * `dokumentid` - Die ID des Dokumentes, das exportiert werden soll.
    * `data` - Das Dokument in Form eines Byte-Arrays.
    * `rootlevel` - Die Ebene, ab der die Navigationsleisten initialisiert werden sollen.
  - **Returns** - Error-Code: 0 für Erfolg, 1 für das Auftreten eines Fehlers.

B.3.2 Classes

B.3.2.1 Class DocumentServer

Diese Serverkomponente ist für den Document-Service zuständig, wird via RMI zugegriffen und bietet alle für die Modifikation von Dokumenten notwenigen Methoden an.

DECLARATION

public class DocumentServer
extends java.lang.Object

CONSTRUCTORS

- *DocumentServer*
  `public` **DocumentServer**`( )`

METHODS

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    - Die ausführbare Methode des Servers, die den Dienst initalisiert und an die RMI-Registry bindet.
  - **Parameters**
    - `String` - der Name und Pfad der Property-Datei im Dateisystem.

- *startMDPServer*
  `public static void` **startMDPServer**`( java.lang.String` **propertyFileLocation**`, java.lang.String` **propertyFileName** `)`
  - **Usage**
    - Diese Methode startet einen eigenständigen HTTP-Server der durch Aufruf eines Servlets die Voransicht der Dokumente ermöglicht.
  - **Parameters**
    - `propertyFileLocation` - Der Pfad der Parameterdatei im Dateisystem.
    - `propertyFileName` - Der Name der benötigten Parameterdatei.

B.3.2.2 CLASS DocumentServiceHLM

Diese Klasse implementiert das DocumentServiceInterface und nutzt als zugrundeliegenden Datenbestand die Klassenbibliothek des HLM-Projektes. Der DocumentService nutzt dazu sowohl die relationale Datenbank als auch die speziellen Repositories der HLM-Komponente.

DECLARATION

public class DocumentServiceHLM
extends java.rmi.server.UnicastRemoteObject
implements DocumentServiceInterface

CONSTRUCTORS

- *DocumentServiceHLM*
  `public` **DocumentServiceHLM**`( )`
  - **Usage**
    - Ein leerer Konstruktor.

METHODS

- *dokumentAbzeichnen*
  `public synchronized Message` **dokumentAbzeichnen**`( int` **id**`, java.lang.String` **username** `)`
  - **Usage**
    - Mit dieser Methode wird ein Dokument von einem Benutzer aus dem Zustand „Zum Abzeichnen“ in den Zustand „Zum Publizieren“ überführt.
  - **Parameters**
    - `id` - Die interne Id des Dokumentes, das abgezeichnet wird.
    - `username` - Der interne Name des Benutzers, der das Dokument abzeichnet
  - **Returns** - Message Ein Nachrichtenobjekt, dass den Erfolg, bzw. Misserfolg der Aktion dokumentiert

---

- *dokumentAbzeichnenAblehnen*
  `public synchronized Message` **dokumentAbzeichnenAblehnen**`( int` **id**`, java.lang.String` **username**`, java.lang.String` **bemerkung** `)`
  - **Usage**
    - Mit dieser Methode wird der Zustand eines Dokumentes durch einen Benutzer vom Zustand „Zum Abzeichnen“ in den Zustand „Abzeichnen abgelehnt“ überführt; die Begründung für die Ablehnung steht in der Bemerkung.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `bemerkung` - Die Begründung für die Ablehnung.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentBearbeiten*
  `public synchronized byte` **dokumentBearbeiten**`( int` **id** `)`
  - **Usage**
    - Mit dieser Methode wird ein Dokument zum Bearbeiten aus dem internen Dokumentenspeicher gelesen und an den diese Methode aufrufenden Client als Byte-Array übertragen.
  - **Parameters**
    - `id` - Die interne Id des Dokumentes, das bearbeitet werden soll.
  - **Returns** - byte[], das Array mit dem Dokument.

- *dokumentErsetzen*
  `public synchronized Message` **dokumentErsetzen**`( byte [] data, java.io.File f, java.lang.String username, int docid )`
  - **Usage**
    - Mit dieser Methode wird ein bestehendes Dokument in dem internen Dokumentenspeicher durch ein neues vom Client in Form eines Byte-Array übertrages Dokument ersetzt.
  - **Parameters**
    - `docid` - Die interne Id des Dokumentes, das ersetzt werden soll.
    - `data` - Das Byte-Array mit dem Dokument.
    - `f` - Fileinformationen zu dem Dokument.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentGenerieren*
  `public synchronized Message` **dokumentGenerieren**`( int id, java.lang.String username )`
  - **Usage**
    - Ein bereits publiziertes Dokument wird durch diese Aktion neu generiert.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes, da neu generiert werden soll.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentLinkUebersetzung*
  `public synchronized int` **dokumentLinkUebersetzung**`( int id, java.lang.String username, java.lang.String selektierterZustand )`
  - **Usage**
    - Diese Methode parst die Links in einem Dokument und versucht sie auf interne Ressourcen umzusetzen.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `selektierterZustand` - Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - int[] Array mit der Anzahl der gefundenen und der Broken-Links.

- *dokumentLinkVorschlaege*
  `public synchronized int` **dokumentLinkVorschlaege**`( int id, java.lang.String username, java.lang.String selektierterZustand )`
  - **Usage**
    - Diese Methode parst das übergebene Dokument und erzeugt Link-Vorschläge.

  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - int[] Liste mit Dokument Ids.

- *dokumentLoeschen*
  `public synchronized Message` **dokumentLoeschen**`( int` **id**`, java.lang.String` **username** `)`
  - **Usage**
    - Der Aufruf dieser Methode löscht ein Dokument aus dem internen Dokumenten-Repository.
  - **Parameters**
    - `id` - Die interne Kennung des zu löschenden Dokumentes
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentNeu*
  `public synchronized int` **dokumentNeu**`( byte [] data, java.io.File` **f**`, java.lang.String` **username**`, java.lang.String` **ressort**`, java.lang.String` **sprache** `)`
  - **Usage**
    - Mit dieser Methode wird ein neues Dokument in das Dokumentenrepository eingebracht.
  - **Parameters**
    - `ressort` - Das Kürzel des betroffenen Ressorts.
    - `sprache` - Das Kürzel der selektierten Sprache.
    - `f` - Fileinformationen zu dem Dokument.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Die Id des neuen Dokumentes

- *dokumentPublikationAblehnen*
  `public synchronized Message` **dokumentPublikationAblehnen**`( int` **id**`, java.lang.String` **username**`, java.lang.String` **bemerkung** `)`
  - **Usage**
    - Diese Methode setzt den Zustand eines Dokumentes von „Zum Publizieren“ nach Publikation abgelehnt ab.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `bemerkung` - Die Begründung für die Ablehnung.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentPublikationAufheben*
  `public synchronized Message` **dokumentPublikationAufheben**`( int` **id**`, java.lang.String` **username** `)`
  - **Usage**
    - Entfernt ein bereits publiziertes Dokument wieder aus dem Internt-Repository.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentPublizieren*
  `public synchronized Message` **dokumentPublizieren**`( int` **id**`, java.lang.String` **username** `)`
  - **Usage**
    - Publiziert ein Dokument in das Internet-Repository.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentUebersetzungsKopieAnlegen*
  `public synchronized int` **dokumentUebersetzungsKopieAnlegen**`( int` **id**`, java.lang.String` **username**`, java.lang.String` **sp** `)`
  - **Usage**
    - Diese Methode kopiert das selektierte Dokument und legt eine Kopie an, für die die Paramter in der selektieren Sprache gesetzt werden.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `sp` - Das Kürzel der selektierten Sprache.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Die Id die die Übersetzungskopie bekommen hat.

- *dokumentUebersichtsSeiteErstellen*
  `public synchronized int` **dokumentUebersichtsSeiteErstellen**`( java.lang.String` **ressort**`, java.lang.String` **filename**`, java.lang.String` **username**`, java.lang.String` **sprache** `)`
  - **Usage**
    - Erstellt ein Dokument, dass Referenzen auf alle anderen Dokumente in diesem Ressort enthält, wobei die Sprachatribute der selektierten Sprache genutzt werden.
  - **Parameters**
    - `ressort` - Das Kürzel des betroffenen Ressorts.
    - `sprache` - Das Kürzel der selektierten Sprache.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.

    - `filename` - Der Name, den die neue Datei bekommen soll.
  - **Returns** - Die Id der Übersichtsseite.

- *dokumentURLFuerAnzeigeInBrowser*
  `public synchronized String` **dokumentURLFuerAnzeigeInBrowser**`( int` **id**`, java.lang.String` **selektierterZustand** `)`
  - **Usage**
    - Diese Methode liefert eine Url, die für die Voransicht des selektierten Dokumentes in dem entsprechenden Zustand genutzt werden kann.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `selektierterZustand` - Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Die Url als String.

- *dokumentVerschieben*
  `public Message` **dokumentVerschieben**`( int` **dokumentid**`, java.lang.String` **neuesRessort**`, java.lang.String` **username** `)`
  - **Usage**
    - Diese Methode verschiebt im internen Dokumenten-Repository ein Dokument in ein anderes Ressort.
  - **Parameters**
    - `dokumentid` - Die Id des zu verschiebenden Dokumentes
    - `neusRessort` - Das Kürzel des neuen Ressorts.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentZumAbzeichnen*
  `public synchronized Message` **dokumentZumAbzeichnen**`( int` **id**`, java.lang.String` **username** `)`
  - **Usage**
    - Ändert den Zustand eines Dokumentes von „In Bearbeitung“ nach „Zum Abzeichnen“.
  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *dokumentZumBearbeiten*
  `public synchronized Message` **dokumentZumBearbeiten**`( int` **id**`, java.lang.String` **username**`, java.lang.String` **bemerkung** `)`
  - **Usage**
    - Ändert den Zustand eines Dokumentes nach „Zum Bearbeiten“, d.h. Überarbeiten.

  - **Parameters**
    - `id` - Die Id des betroffenen Dokumentes.
    - `username` - Der interne Name des die Aktion ausführenden Benutzers.
    - `bemerkung` - Die Begründung für die Aufforderung zur Überarbeitung.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *getBrokenLinkInformation*
  `public synchronized BrokenLinkInformation` **getBrokenLinkInformation(**
  `int` **dokumentid,** `java.lang.String` **status**)
  - **Usage**
    - Liefert ein Objekt zurück, dass die Hyperlink-Informationen für das selektierte Dokument enthält.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Das Objekt mit den Hyperlink-Informationen.

- *getCompleteFileLocator*
  `public String` **getCompleteFileLocator(** `ti.daphne.Document` **d,** `boolean` **exportmode,** `boolean` **link,** `boolean` **setExcusePage**)
  - **Usage**
    - Liefert den vollständigen Pfad zu dem Dokument in einem Dateisystem.
  - **Parameters**
    - `d` - Das betroffene Dokument.
    - `exportmode` - Der Modus, für das die Zugriffsinformation erstellt werden soll.True für Export auf Web-Präsenz.
    - `link` - True, wenn es sich um einen Link handelt, der gerade zugeordnet werden soll.
    - `setExcusePage` - True, wenn eine Entschuldigungsseite an die Stelle dieses Dokumentes gesetzt werden soll, wenn es nicht publiziert ist.
  - **Returns** - Der vollständige Pfad.

- *getCompleteFileLocator*
  `public String` **getCompleteFileLocator(** `ti.daphne.Ressort` **r,** `boolean` **exportmode,** `boolean` **link,** `boolean` **setExcusePage**)
  - **Usage**
    - Liefert den vollständigen Pfad zu dem Ressort in einem Dateisystem.
  - **Parameters**
    - `r` - Das betroffene Ressort.
    - `exportmode` - Der Modus, für das die Zugriffsinformation erstellt werden soll.True für Export auf Web-Präsenz.
    - `link` - True, wenn es sich um einen Link handelt, der gerade zugeordnet werden soll.
    - `setExcusePage` - True, wenn eine Entschuldigungsseite an die Stelle dieses Dokumentes gesetzt werden soll, wenn es nicht publiziert ist.

  - **Returns** - Der vollständige Pfad.

- *getDBStatus*
  `public String` **getDBStatus**`( java.lang.String` **status** `)`
  - **Usage**
    - Liefert den zu einem HLM-Status gehörigen Statuseintrag in JDaphne.
  - **Parameters**
    - `status` - Der HLM-Status.
  - **Returns** - Der zugehörige JDaphne-Status.

- *getLinkFromInformation*
  `public synchronized BrokenLinkInformation` **getLinkFromInformation**`(`
  `int` **dokumentid**`, java.lang.String` **status** `)`
  - **Usage**
    - Liefert ein Objekt mit den Links, die von dem übergebenen Dokument ausgehen.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Das Objekt mit den Hyperlink-Informationen.

- *getLinkToInformation*
  `public synchronized BrokenLinkInformation` **getLinkToInformation**`( int`
  dokumentid`, java.lang.String` **status** `)`
  - **Usage**
    - Liefert ein Objekt mit den Links, die auf das übergebene Dokument verweisen.
  - **Parameters**
    - `dokumentid` - Die Id des betroffenen Dokumentes.
    - `selektierter` - Zustand Die Zustandsversion, die für die Aktion an dem Dokument genutzt werden soll.
  - **Returns** - Das Objekt mit den Hyperlink-Informationen.

- *getPURIStatus*
  `public String` **getPURIStatus**`( java.lang.String` **status** `)`
  - **Usage**
    - Liefert zu einem JDaphne-Status den zugehörigen HLM-Status.
  - **Parameters**
    - `status` - Der JDaphne-Status.
  - **Returns** - Der HLM-Status.

- *init*
  `public void` **init**`( java.lang.String` **propertiesfileUrl** `)`
  - **Usage**
    - Die Initialisierung des Dienstes durch eine Parameterdatei.

  - **Parameters**
    * `propertiesURL` - Die URL, unter der die Parameterdatei geladen werden kann.

- *loadProperties*
  `public static Properties` **loadProperties**`( java.net.URL` **u** `)`
  - **Usage**
    * Diese Methode lädt eine Parameterdatei aus von einer URL.
  - **Parameters**
    * `u` - Die URL der Parameterdatei.

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    * Eine zu Testzwecken implementierte Methode, die die Ausführung des Services ohne Server ermöglicht.
  - **Parameters**
    * `args` - java.lang.String[]

- *move*
  `public boolean` **move**`( ti.daphne.Ressort` **ressort**`, ti.daphne.Ressort` **newParent** `)`
  - **Usage**
    * Verschiebt ein Verzeichnis mitsamt Inhalten.
  - **Parameters**
    * `ressort` - Das zu verschiebende Ressort.
    * `newParent` - Das Ressort dem das zu verschiebende Ressort untergeordnet werden soll.
  - **Returns** - boolean True wenn erfolgreich.

- *newDokumentAsTemplate*
  `public synchronized Message` **newDokumentAsTemplate**`( int` **id**`, java.lang.String` **ressort**`, java.lang.String` **username**`, java.lang.String` **sprache** `)`
  - **Usage**
    * Legt ein neues, auf dem selektierten Dokument basierendes Template an.
  - **Parameters**
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
    * `id` - Die Id des als Vorlage zu nutzenden Dokumentes.
    * `ressort` - Das Kürzel des betroffenen Ressorts.
    * `sprache` - Das Kürzel der selektierten Sprache.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *parseDocument*
  `public Document` **parseDocument**`( byte [] data )`
  - **Usage**

    * Diese Methode parst aus dem übergebenen Dokument Hyperlinks und Schlagwörter heraus.
  - **Parameters**
    * `data` - Das Dokument als Byte-Array.
  - **Returns** - Ein Objekt mit den Parserergebnissen.

- *pathIsCorrect*
  `public boolean` **pathIsCorrect**`( java.lang.String` **path**`, java.lang.String` **zustand**`, boolean` **exportmode** `)`
  - **Usage**
    * Liefert True, wenn ein Link-Pfad dem Zustand des Dokumentes entsprechend OK ist.
  - **Parameters**
    * `path` - Der genutzte Pfad.
    * `exportmode` - Der Modus, für das die Zugriffsinformation erstellt werden soll.True für Export auf Web-Präsenz.
    * `zustand` - des Dokumentes, auf das verlinkt wird.
  - **Returns** - True, wenn korrekt.

- *regenerateDocuments*
  `public synchronized Message` **regenerateDocuments**`( int [] id, java.lang.String` **username** `)`
  - **Usage**
    * Generiert die selektierten, bereits publizierten Dokumente neu für den statischen Export.
  - **Parameters**
    * `id[]` - Die Ids der betroffenen Dokumentes.
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - Message Das Informationsobjekt mit der Rückmeldung für den Benutzer.

- *remove*
  `public synchronized boolean` **remove**`( ti.daphne.Ressort` **r** `)`
  - **Usage**
    * Löscht ein Ressort aus dem Repository, das Ressort muss vorher geleert worden sein.
  - **Parameters**
    * `ressort` - Das zu löschende Ressort.
  - **Returns** - boolean True wenn erfolgreich.

- *rename*
  `public synchronized boolean` **rename**`( ti.daphne.Ressort` **r**`, java.lang.String` **newName** `)`
  - **Usage**
    * Benennt ein Ressort im Dokumentenrepository um.

  - **Parameters**
    * `ressort` - Das selektierte Ressort.
    * `newName` - Der neue Name, der für das Ressort vergeben wird.
  - **Returns** - True, wenn erfolgreich.

- *restoreDocuments*
  `public synchronized int` **restoreDocuments**`( java.lang.String` **username** `)`

  - **Usage**
    * Diese Methode regeneriert Dokumente aus einer Sicherheitskopie, die beim Einspielen angelegt wird.
  - **Parameters**
    * `username` - Der interne Name des die Aktion ausführenden Benutzers.
  - **Returns** - 0 Wenn Aktion erfolgreich

- *restoreKeywords*
  `public synchronized int` **restoreKeywords**`( )`

  - **Usage**
    * Ein Update des Keyword-Bestandes durchführen.

- *suggestLinks*
  `public String` **suggestLinks**`( java.lang.String` **link**`, java.lang.String` **errortag**`, boolean` **exportmode**`, java.lang.String` **sprache**`, java.lang.String` **ressortKurz**`, java.lang.String` **pathroot**`, int` **rootlevel**`, int` **docId**`, java.lang.String` **status**`, boolean` **parsedDocument** `)`

  - **Usage**
    * Ersetzt in Links der Form **Ein Seminar-Vortrag** (at mittag101.html) die URL mit in dem System verfügbaren Dokumenten, falls verfuegbar. Andernfalls wird der Errortag eingefuegt.
  - **Parameters**
    * `link` - Der vollständige Hyperlinktext.
    * `exportmode` - Der Modus, für das die Zugriffsinformation erstellt werden soll.True für Export auf Web-Präsenz.
    * `ressortKurz` - Das Kürzel des betroffenen Ressorts.
    * `sprache` - Das Kürzel der selektierten Sprache.
    * `errortag` - Eine Fehlermeldung, die in dem Hyperlink platziert wird.
    * `status` - Die Version des Dokumentes das den Hyperlink enthält.
    * `docId` - Die eindeutige Kennung des Dokumentes, das den Hyperlink enthält.
    * `parsedDocument` - True, wenn das Dokument bereits geparst worden ist.
    * `rootlevel` - Die Ebene ab der die Navigation in die Seite eingebaut wird.
    * `pathroot` - Falls nicht in die Dokumentroot exportiert wird, an dieser Stelle einen Pfadprefix eintragen.
  - **Returns** - Der vorgeschlagene Hyperlink.

B.3.2.3 Class **ExportServer**

Der Export-Server übernimmt das Layouten von Dokumenten und deren Transport auf den Web-Server, basierend auf Templates, den Dateien und der Datenbank werden Dokumente für den statischen und dynamischen Export sowie die Voransicht erzeugt. Der Zugriff auf diesen Server erfolgt via RMI.

Declaration

public class ExportServer
extends java.lang.Object

Constructors

- *ExportServer*
  `public` **ExportServer**`( )`

Methods

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    - Die ausführbare Methode des Servers, die den Dienst initalisiert und an die RMI-Registry bindet.
  - **Parameters**
    - `String` - der Name und Pfad der Property-Datei im Dateisystem.

B.3.2.4 Class **ExportService**

Diese Klasse implementiert das ExportServiceInterface und stellt die erforderlichen Methoden bereit.

Declaration

public class ExportService
extends java.rmi.server.UnicastRemoteObject
implements ExportServiceInterface

Constructors

- *ExportService*
  `public` **ExportService**`( )`
  - **Usage**
    - Ein leerer Konstruktor.

METHODS

- *buildPage*
  `public String` **buildPage(** `int` **docId,** `byte []` **data,** `int` **rootlevel,** `java.lang.String` **ressKurz,** `boolean` **export,** `java.lang.String` **pathroot,** `java.lang.String` **templatetype,** `boolean` **physicalExport**)
  - **Usage**
    - Die zentrale Methode des Export-Service baut ein fertiges Dokument aus den einzelnen Bestandteilen, dem Template, den Datenbankinformationen und dem Content zusammen.
  - **Parameters**
    - `dokumentid` - Die Id des zu generierenden Dokumentes.
    - `data` - Das Byte-Array mit dem Dokument-Inhalt.
    - `rootlevel` - Die Ebene ab der die Navigation in die Seite eingebaut wird.
    - `ressKurz` - Das eindeutige Ressortkürzel des Ressorts in das das Dokument exportiert werden soll (wichtig bei Mehrfachnutzung eines Dokumentes).
    - `export` - Auf True setzen, wenn finaler Export, False bei Voransicht.
    - `pathroot` - Falls nicht in die Dokumentroot exportiert wird, an dieser Stelle einen Pfadprefix eintragen.
    - `templateType` - Der Name des Templates als String.
    - `phyiscalExport` - True, wenn statischer Export angestrebt, false bei dynamischem Export.
  - **Returns** - String der die gelayoutete Seite enthält.

- *buildPathToRessort*
  `public String` **buildPathToRessort(** `java.lang.String` **ressortKurz,** `int` **rootlevel,** `boolean` **export**)
  - **Usage**
    - Liefert den Pfad inklusive des ausgewaehlten Verzeichnisses.
  - **Parameters**
    - `rootlevel` - Die Ebene ab der die Navigation in die Seite eingebaut wird.
    - `ressKurz` - Das eindeutige Ressortkürzel des Ressorts zu dem der Pfad bestimmt werden soll.
    - `export` - Auf True setzen, wenn finaler Export, False bei Voransicht.
  - **Returns** - Der Pfad zu dem übergebenen Verzeichnis.

- *buildPathToRessort*
  `public String` **buildPathToRessort(** `java.lang.String` **ressortKurz,** `java.lang.String` **pathroot,** `int` **rootlevel,** `boolean` **export**)
  - **Usage**
    - Liefert den Pfad inklusive des ausgewaehlten Verzeichnisses unter Berücksichtigung einer als Suffix verwendeten Verzeichniswurzel.
  - **Parameters**
    - `rootlevel` - Die Ebene ab der die Navigation in die Seite eingebaut wird.
    - `ressKurz` - Das eindeutige Ressortkürzel des Ressorts zu dem der Pfad bestimmt werden soll.
    - `pathroot` - Die Verzeichniswurzel, die dem Pfad vorangestellt wird.

    * `export` - Auf True setzen, wenn finaler Export, False bei Voransicht.
  - **Returns** - Der Pfad zu dem übergebenen Verzeichnis.

- *exportToInternet*
  `public int` **exportToInternet**`( int` **documentId,** `byte []` **data,** `int` **rootlevel** `)`
  - **Usage**
    * Diese Methode führt den statischen Export, intern Aufruf von buildPage, in das File-System des Web-Servers durch.
  - **Parameters**
    * `dokumentid` - Die ID des Dokumentes, das exportiert werden soll.
    * `data` - Das Dokument in Form eines Byte-Arrays.
    * `rootlevel` - Die Ebene, ab der die Navigationsleisten initialisiert werden sollen.
  - **Returns** - Error-Code: 0 für Erfolg, 1 für das Auftreten eines Fehlers.

- *extractContentFromOrgDocument*
  `public String` **extractContentFromOrgDocument**`( java.lang.String` **orgDoc** `)`
  - **Usage**
    * Schneidet den Teil des Dokumentes zwischen den Body-Tags heraus.
  - **Parameters**
    * `orgDoc` - Das Dokument, aus dem der Body herausgeschnitten werden soll.
  - **Returns** - Der Body des Dokumentes.

- *extractHeadFromOrgDocument*
  `public String` **extractHeadFromOrgDocument**`( java.lang.String` **orgDoc** `)`
  - **Usage**
    * Schneidet den Header aus dem Dokument heraus.
  - **Parameters**
    * `orgDoc` - Das Dokument, aus dem der Header herausgeschnitten werden soll.
  - **Returns** - Der Header des Dokumentes.

- *getFileName*
  `public String` **getFileName**`( int` **docId** `)`
  - **Usage**
    * Liefert den Dateinamen eines Dokumentes für den Export.
  - **Parameters**
    * `docId` - Die Id des Dokumentes, dessen Name erzeugt werden soll.
  - **Returns** - Der Dateiname.

- *getParentRessorts*
  `public Ressort` **getParentRessorts**`( java.lang.String` **ressortKurz** `)`
  - **Usage**

    * Liefert zu einem Ressort den Vater mit den Vaters Vätern.
  - **Parameters**
    * `ressortKurz` - Das Ressortkürzel des Ressorts, dessen Väter gesucht sind.
  - **Returns** - Ein Array von Ressorts mit dem WurzelRessort an der Stelle 0.

- *init*
  `public Properties` **init**`( java.lang.String` **propertiesURL** `)`
  - **Usage**
    * Die Initialisierung des Dienstes durch eine Parameterdatei.
  - **Parameters**
    * `propertiesURL` - Die URL, unter der die Parameterdatei geladen werden kann.

- *loadProperties*
  `public Properties` **loadProperties**`( java.net.URL` **u** `)`
  - **Usage**
    * Diese Methode lädt eine Parameterdatei aus von einer URL.
  - **Parameters**
    * `u` - Die URL der Parameterdatei.

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    * Eine zu Testzwecken implementierte Methode, die die Ausführung des Services ohne Server ermöglicht.
  - **Parameters**
    * `args` - java.lang.String[]

- *metaData*
  `protected String` **metaData**`( ti.daphne.Document` **doc**`, ti.daphne.Ressort` **res** `)`
  - **Usage**
    * Diese Methode fügt in ein zu exportierendes Dokument die zugehörigen Metadaten ein.
  - **Parameters**
    * `doc` - Das Dokument für das die Metadaten erstellt werden sollen.
    * `res` - Das Ressort aus dem die Metadaten entnommen werden sollen.
  - **Returns** - Der Metadatenblock für das Dokument

- *removeFile*
  `public boolean` **removeFile**`( int` **documentId**`, int` **rootlevel** `)`
  - **Usage**
    * Löscht eine Datei vom Web-Server unter Berücksichtigung der Hyperlinks.
  - **Parameters**
    * `dokumentid` - Die Id des betroffenen Dokumentes.
    * `rootlevel` - Die Ebene, ab der die Navigation einsetzt.

  - **Returns** - True wenn die Aktion erfolgreich war.

- *saveToInternet*
  `public int` **saveToInternet**`( int` **documentId**`, byte []` **data**`, int` **rootlevel** `)`
  - **Usage**
    - Diese Methode überträgt die entsprehcende Datei auf den Web-Server, noch ohne Layout, damit dieses dynamisch erzeugt werden kann.
  - **Parameters**
    - `dokumentid` - Die ID des Dokumentes, das exportiert werden soll.
    - `data` - Das Dokument in Form eines Byte-Arrays.
    - `rootlevel` - Die Ebene, ab der die Navigationsleisten initialisiert werden sollen.
  - **Returns** - Error-Code: 0 für Erfolg, 1 für das Auftreten eines Fehlers.

- *toString*
  `public String` **toString**`( )`
  - **Usage**
    - Liefert eine textuelle Beschreibung des Dienstes.
  - **Returns** - Eine Dienstbeschreibung.

Tabellarischer Bildungsgang

Name:	Andreas Lutz Heuer
Geburtsdatum:	22.12.1970
Gebutsort:	Haltern
1977-1981	Besuch der Silverbergschule
1981-1990	Besuch des Städtischen Gymnasiums Haltern
	Abschluß: Abitur
1990-1995	Physik-Studium an der Westfälischen Wilhelms Universität Münster
Oktober 1995	Physik-Diplom
1996-1997	Zivildienst am Institut für Klinische Radiologie (IKR) des Universitätsklinikums Münster
seit 1997	Wissenschaftlicher Mitarbeiter im Institut für Telematik in Trier unter der Betreuung von Prof. Dr. Christoph Meinel