# Konzeption und Realisierung eines Web-Content-Management-Systems am Beispiel des Rechenzentrums Erlangen-Nürnberg

**Diplomarbeit im Fach Informatik**

vorgelegt von

**Wolfgang Wiese**

Angefertigt am

**Lehrstuhl für Informatik VII**
**(Rechnerarchitektur und –Verkehrstheorie)**
**Institut für Mathematische Maschinen und Datenverarbeitung**
**Friedrich-Alexander-Universität Erlangen-Nürnberg**

In Zusammenarbeit mit dem

**Regionalen Rechenzentrum Erlangen-Nürnberg**
**(RRZE)**

Hochschullehrer: Prof. Dr. Ing. U. Herzog (IMMD7)
Betreuer: Dr. Ing. W. Dulz (IMMD7)
Dr. Ing. G. Hergenröder (RRZE)

Erlangen, Februar - Juli 2000

# Erklärung

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 20. Juli 2000

Wolfgang Wiese

# Inhaltsverzeichnis

ÜBERSICHT ..... 7

1. EINFÜHRUNG ..... 9

1.1. Das World Wide Web ..... 9

1.2. Traditionelles Webpublishing ..... 10

1.3. Der Content-Lifecycle ..... 13

1.3.1. Inhalte erstellen ..... 13

1.3.2. Kontrollieren und freigeben ..... 14

1.3.3. Publizieren ..... 14

1.3.4. Archivieren ..... 15

1.4. Content-Management-Systeme ..... 15

1.5. Kosten von WCMS ..... 17

1.6. Vor- und Nachteile von WCMS ..... 18

2. KOMPONENTEN EINES WCMS ..... 21

2.1. Allgemeines ..... 21

2.2. Benutzer- und Zugriffsverwaltung ..... 21

2.3. Redaktionelle Verwaltung ..... 22

2.3.1. Seitenerstellung ..... 23

2.3.2. Kontroll- und Freigabe-Zyklus ..... 24

2.3.3. Versionskontrolle und Archivierung ..... 25

2.4. Import- und Exportschnittstellen ..... 27

3. SYSTEMUMGEBUNG UND VERFÜGBARE TECHNIKEN ..... 29

3.1. Allgemeines ..... 29

**3.2. Webserver und Betriebsystem** ..... **29**
3.2.1. Apache ..... 29
3.2.2. Microsoft-IIS ..... 30
3.2.3. Lotus Domino ..... 30

**3.3. Webtechniken- und Sprachen** ..... **31**
3.3.1. Allgemeines ..... 31
3.3.2. HTML ..... 31
3.3.3. XML ..... 32
3.3.4. CGI ..... 33
3.3.5. SSI ..... 34
3.3.6. PHP ..... 35
3.3.7. Java ..... 35
3.3.8. JavaScript ..... 36
3.3.9. ASP ..... 36
3.3.10. Sonstiges ..... 37

**3.4. Akzeptanz und Nutzung der Webtechniken** ..... **37**

**4. KONZEPTION** ..... **39**

**4.1. Einleitung** ..... **39**

**4.2. Wahl der Plattform** ..... **39**

**4.3. XML-Integration** ..... **40**

**4.4. Wahl und Konzeption des Benutzer-Interfaces** ..... **40**
4.4.1. Clientseitige Interfaces ..... 40
4.4.2. Serverseitige Interfaces ..... 41

**4.5. Konzepte zur Daten- und Seitenhaltung** ..... **41**
4.5.1. Publishing-/Staging-Server ..... 41
4.5.2. Dynamisches Publishing ..... 43
4.5.3. Mischformen ..... 44
4.5.4. PSE-Konzept ..... 44

**5. IMPLEMENTATION** ..... **47**

**5.1. Definitionsphase ... 47**
5.1.1. Lastenheft ... 47
5.1.2. Pflichtenheft ... 48
5.1.2.1. Zielbestimmung ... 48
5.1.2.2. Einsatzgebiet und Anforderungen an das System ... 49
5.1.2.3. Architekturentwurf ... 50
5.1.2.4. Beschreibung des Funktionsumfangs ... 50
5.1.2.5. System-Daten ... 51
5.1.2.6. Bedienoberfläche ... 53

**5.2. Umsetzung ... 54**
5.2.1. Module ... 54
5.2.1.1. XML_DBI ... 54
5.2.1.2. USER_DBI ... 55
5.2.1.3. GROUP_DBI ... 55
5.2.1.4. TODO_DBI ... 56
5.2.1.5. PAGETYPE_DBI ... 57
5.2.1.6. LINK_DBI ... 57
5.2.1.7. Weitere Module ... 58
5.2.2. Benutzer-Interface / Hauptprogramm ... 58
5.2.3. Überwachungsprogramm ... 59

**6. EINSATZ UND TEST EINES WCMS ... 61**

**6.1. Fallbeispiele ... 61**
6.1.1. Login ... 61
6.1.2. Seite erstellen ... 62
6.1.3. Aufgaben-Management abrufen und Seite freigeben ... 64
6.1.4. Seite bearbeiten ... 66
6.1.5. Benutzer-Administration ... 67
6.1.6. Dokumententyp-Verwaltung ... 68
6.1.7. Verzeichnisse ... 70

**6.2. Datensätze ... 72**
6.2.1. Benutzer-Daten ... 72
6.2.2. Dokumententyp-Daten ... 73

6.2.3. Verzeichnis-Daten ..... 74
6.2.4. Seiten-Daten ..... 74

7. KLASSIFIKATION UND VERGLEICH ..... 76

7.1. Klassifikation ..... 76

7.2. Trendanalyse ..... 78

8. ZUSAMMENFASSUNG UND AUSBLICK ..... 80

A1. LITERATUR- UND WEBVERZEICHNIS ..... 82

A1.1. Buch- und Arbeitsliteratur ..... 82

A1.2. Informationsseiten im WWW ..... 82

A1.3. Artikel aus Fachzeitschriften ..... 83

A1.4. Weitere Quellen ..... 84

A2. ABKÜRZUNGSVERZEICHNIS ..... 86

A3. ANBIETER VON WCMS AUF DER CEBIT 2000 ..... 87

A4. HINWEISE ..... 89

B1. INDEX ..... 90

# Übersicht

Die wachsende Nutzung des World Wide Webs und die damit gestiegende Nachfrage an online abrufbaren Informationsseiten hat dazu geführt, daß große Firmen und Einrichtungen gezwungen sind, Verfahren zu nutzen, die den Einbau und das Management vieler Webseiten steuern und unterstützen. Hierbei wird im starken Maße auf die Verfahren des Content-Managements zurückgegriffen, wobei aber der Schwerpunkt auf dem Publishing der Seiten im Internet liegt.
Wenn man die Entwicklung des Webpublishing betrachtet, so erkennt man, daß dieses sehr auf alte, gewachsene Strukturen zurückgeht: Ein Mitarbeiter, meistens ein Techniker, fungierte als Anlaufstelle für viele Autoren mit der Aufgabe, Webseiten von häufig in Papierform vorhandenen oder neuen Informationen zu erstellen.
Mit der Zeit kamen immer weitere Webseiten hinzu, mehr Informationen wurden online abgefragt, und die Firma bzw. die Einrichtung mußte sich Gedanken über ein adäquates Erscheinungsbild zur Öffentlichkeit hin machen. Der vormals einzelne Mitarbeiter erhielt weitere Kollegen oder aber die Autoren gingen dazu über, selbst die Seiten in HTML zu schreiben und sie gleich, in Umgehung des dafür zuständigen Mitarbeiters, zu publizieren. Die sichtbare Folge hiervon war, daß Dokumente Fehler enthielten, sich nicht an das vorgegebene Erscheinungsbild, das Coperate Design, hielten und binnen kurzer Zeit kein Überblick mehr darüber bestand, wer für eine Seite verantwortlich war. Hinzu kam nach einiger Zeit weiter, daß die Seiten „alterten“, das heißt, die Informationen nicht mehr aktuell waren.

Durch die Einführung von Web-Content-Management-Systemen versucht man dieser Situation Herr zu werden und gleichzeitig für die Einhaltung eines Coperate Designs zu sorgen. Der einzelne Autor soll dabei wieder zurück auf seine Kernkompentenz geleitet werden, nämlich das Schreiben der Information, und sich keine Gedanken mehr machen müssen über gestalterische oder technische Aspekte bei der Erstellung einer Webseite. Im Hintergrund steht dabei auch die Kostenersparnis, die dadurch anfällt, daß die Autoren keine Zeit mehr darauf verwenden müssen, HTML und Grundlagen in Webdesign zu lernen.

Diese Arbeit beschäftigt sich mit der Konzeption und der Realisierung eines Web-Content-Management-Systems (WCMS). Dabei wird versucht sich nach den aktuellen Entwicklungen im Internet, also zum Beispiel nach den verwendeten Servertechnologien, zu richten und auch einen besonderen Blick auf die redaktionelle Arbeit bei der Erstellung von Webseiten zu wer-

fen. Die Schwerpunkte dieser Arbeit liegen auf den Möglichkeiten und Techniken zur Realisierung eines WCMS. Nicht jedoch auf der Erklärung, was Content-Management im Detail bedeutet. Dies erfolgte bereits durch diverse Publikationen in Fachzeitschriften (insb. [Art1], [Art2] und [Art3]) und der Arbeit von Oliver Zschau geleistet [Lit1]. Gleichwohl wird ein Einblick in diese Themen insoweit gegeben, als daß es als Grundlage für die späteren Kapitel notwendig ist.

In den ersten Kapiteln finden sich ein Überblick und eine Einführung in Content-Mangement-Systemen: Kapitel 1 „Einführung“ gibt eine Einführung in die Grundlagen des Web-Content-Management. In Kapitel 2 „Komponenten eines WCMS“ werden diese Grundlagen vertieft zur Definition der einzelnen Komponenten eines WCMS.
Daran anschließend wird Kapitel 3 „Systemumgebung und verfügbare Techniken“ besprochen, in welcher Umgebung ein WCMS eingesetzt wird und welche Techniken vorhanden sind, die genutzt werden können.
Mit dieser Vorbereitung wird in Kapitel 4 „Konzeption“ begonnen zu definieren, nach welchen Konzepten die Programmierung verlaufen kann, welche Techniken eingesetzt werden können, deren Vor- und Nachteile, sowie die Wahlmöglichkeiten für Hard- und Software angegeben. In Kapitel 5 „Implementation“ erfolgt dann die Besprechung der eigentlichen Implementation.
Ergebnisse der Implementation werden im Kapitel 6 „Einsatz und Test eines WCMS“ gezeigt. Anhand von Beispielen wird die Funktionsweise des in Kapitel 5 erstellten Basis-WCMS gezeigt. Da das entwickelte System unter Realbedingungen getestet und später eingesetzt werden soll, wird an diesen Stellen stark von der Systemumgebung ausgegangen, wie sie am Regionalem Rechenzentrum Erlangen, inklusive dem angeschlossenen Webserver für die Universität Erlangen-Nürnberg (FAU), gegeben ist. Auch wird der praktische Einsatz in weit verbreiteten Systemen ein wichtiger Punkt sein, so daß die Entwicklung des WCMS sich weniger an Betriebsysteme oder Webserver richtet, die in Bezug auf das Dokumenten-Managements Vorteile bieten (z.B. Lotus Domino) sondern an Systeme, die sich auf dem Markt durchgesetzt und bewährt haben.
Daran anschließend wird in Kapitel 7 „Klassifikation und Vergleich“ eine Klassifikation und ein Vergleich des entwickelten Systems mit anderen vorhandenen Systemen geführt.
Zum Abschluß wird in Kapitel 8 „Zusammenfassung und Ausblick“ aufgezeigt, welch enormes Potential für den weiteren Ausbau und Nutzen durch WCMS möglich und empfehlenswert ist.

# 1. Einführung

## 1.1. Das World Wide Web

Wenn man die Entwicklungszeit des Internets und die, des darin eingebetteten World Wide Webs (WWW) betrachtet, dann fällt zuerst einmal auf, wie jung das WWW eigentlich noch ist: Während das Internet, in Form des ARPANET („Advanced Research Project Agency Network“) schon 1969 ins Leben gerufen wurde [Lit2], folgte das WWW erst ab 1992 mit der Definition der Hypertext Markup Language (HTML) und der Implementation des Hypertext Transfer Protocols (HTTP).

Die Grundsyntax von HTML und deren Semantik sind in der HTML-Norm niedergelegt. Diese und alle anderen webbezogenen Normfragen werden vom World Wide Web Consortium (W3C) behandelt und festgelegt [Web1]. Die HTML-Norm gibt es bereits in mehreren fortlaufenden Versionen, wobei die Version 4.0 als letzte definiert wurde.

Hypertext- (und Hypermedia-) Dateien, sind im Grunde Daten, die Text, Bilder, Grafiken, Audio- und Videosequenzen und vor allem dynamische Verbindungen zu anderen Daten im Internet enthalten können [Lit3]. Diese Daten sind der Hauptgrund, weswegen sich Benutzer ins Internet einwählen und es gerade im kommerziellen und sozialen Sektor so erfolgreich ist. Über diese Verbindungen („Links“) können die Benutzer zwischen miteinander verbundenen Dokumenten hin- und herschalten, egal ob diese sich auf demselben Host befinden oder die Hosts weit voneinander entfernte Rechner sind.

Dateien im Internet die der HTML-Norm unterliegen, werden als Webseiten oder Webdokumente bezeichnet. Um auf diese zugreifen zu können, benötigt man eine spezielle Software, einen Browser. Beispiele sind der „Netscape Navigator“, der „Microsoft Internet Explorer“ oder „Mosaic“, wobei letzterer praktisch nur noch von Webdesignern oder Entwicklern eingesetzt wird. Netscape/AOL und Microsoft nehmen mit ihren Browsern eine marktbeherrschende Stellung ein: 90% aller eingesetzten Webbrowser sind Versionen dieser beiden Firmen [Web2].

Ein Webbrowser nimmt über das Internet Verbindung mit den Webservern (s.u.) anderer Rechner auf und fordert über das HTTP (oder ein anderes Protokoll) die jeweils gewünschten Dokumente ab. Danach stellt er das Dokument entsprechend seiner HTML-Definitionen dar, wobei er gegebenenfalls weitere Dokumente anfordert, die durch das HTML-Protokoll in das Dokument eingebunden sind [Lit4].

Wie oben bereits angedeutet, bedarf es auf dem Host, der die Webseiten bereithält, einer Software, die auf Zugriffe über das HTTP reagiert. Diese Software wird als

Webserver bezeichnet. Im Prinzip handelt es sich dabei um einen dauernd laufenden Prozeß (einen „Dämon“), welcher auf einem bestimmten Port (meist Port 80) nach HTTP-Requests lauscht. Kommt ein HTTP-Request, wird geprüft, ob dieser korrekt ist, ob die angeforderte Datei vorhanden ist und ob die Leseberechtigung vorhanden ist. Danach wird der Inhalt der Datei (oder eine Fehlermeldung) über das HTTP an den anfragenden Browser geschickt und die Verbindung wieder abgebrochen.

Derzeit dominiert der Webserver Apache den Markt und bietet auch im Vergleich zu den kommerziellen Produkten von Microsoft, Netscape oder Lotus eine hohe Qualität und Benutzerfreundlichkeit.

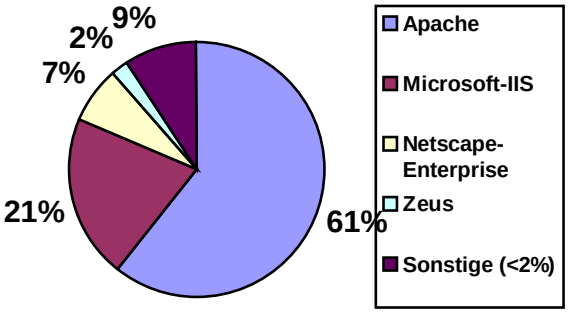



**Abb. 1. Serverstatistik**

Apache belegte einen Marktanteil von 61% von über 14 Millionen gemessenen Webservern im Mai 2000 [Web3].

## 1.2. Traditionelles Webpublishing

Nachdem Tim Berners-Lee 1990 das HTML-Protokoll und den ersten Browser, der damals ebenfalls noch unter dem Namen „World Wide Web“ lief und später in „Nexus“ umgetauft wurde, vorstellte [Web4], erkannten die Universitäten als erste das Entwicklungspotential in dieser Entwicklung. Aufgrund der einfachen und logischen Struktur von HTML-Dokumenten war es ein leichtes, bereits vorhandene Textverwaltungssysteme an dieses Format anzupassen. An den Lehrstühlen der Universitäten, an denen oft und gern das Textverarbeitungssystem LaTeX eingesetzt wurde und wird, war es Dank eines Konvertierungsprogrammes („*latex2html*“) möglich, vorhandene Dokumente in HTML umzuwandeln.

Da jedoch nur ein Teil der Dokumente in LATEX vorlagen und/oder kein geeignetes Portierungsprogramm vorhanden war, mußten Seiten auch von Hand erstellt werden.

Hierzu wurde dann meist ein „Webmaster“ angestellt, bzw. ein Mitarbeiter, beauftragt diese Arbeiten neben den normalen Aufgaben zu erledigen.

Der Webmaster bekam somit von den Autoren Dokumente unterschiedlichster Formate, die dann als Webseite publiziert werden sollten. Hierzu mußte er den Inhalt aus

den Dokumenten heraustrennen und in eine neue Webseite einbauen.

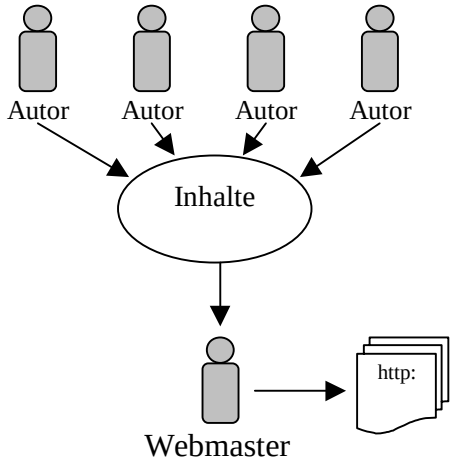



**Abb. 2. Traditionelles Webpublishing**

Abbildung 2 verdeutlicht dies und weist schon auf Kernprobleme dieses Vorgehens hin: Zum einen muß der Webmaster in der Lage sein, die Dokumente von vielen Autoren und damit möglicherweise vielen verschiedenen Textformaten zu verarbeiten, zum anderen kann er in der Regel keine Aussage mehr über die Inhalte treffen und somit auch nicht wissen, wann ein Text veraltet oder schlichtweg falsch ist.

Der einzige Vorteil an diesem Vorgehen ist, daß die Struktur des Webauftritts aus einer Hand kommt und somit Referenzen innerhalb des eigenen Webauftritts bekannt und kontrolliert sind.

Dies trifft jedoch nicht auf Links zu, die aus Dokumenten herausgehend auf andere Server im Internet verweisen. Hier existiert weder Sicherheit noch eine Servicegarantie seitens der Betreiber, daß einmal vorhandene Seiten nach der Publizierung länger vorhanden sind.

Mit der immer größer werdenden Verbreitung und Nutzung des World Wide Webs stieg der Bedarf fremde Informationen schnell abzurufen und gleichzeitig kam der Wunsch auf, eigene Informationen abrufbar zu machen. Für einen einzelnen Webmaster war dies jedoch nicht mehr machbar. So wurden zum einen weitere Personen damit beauftragt, die Informationen ins Web zu bringen, zum anderen taten es die Autoren selbst. Die zwangsläufige Folge war der Verlust der Kontrolle über die Struktur und die Inhalte: Abhängig von der Ausprägung und Effizienz der internen Kommunikation zwischen den Webmastern und Autoren stieg die Unübersichtlichkeit der Webseiten zum Teil stark an.

Die Abbildung 3 versucht dies zu skizzieren. Man erkennt die Problematik die auftritt, wenn die Kommunikations- und Hierarchiewege nicht eingehalten werden. Obwohl hier eine klare hierarchische Struktur vorgegeben ist, bei dem einen Webmaster zwei Assistenten untergeordnet sind, wird diese oft aus unterschiedlichsten Gründen nicht eingehalten.

Dabei müssen die Gründe für das Nicht-Einhalten der Hierarchie nicht einmal schlecht sein: So reicht bspw. ein temporäres Eilbedürfnis oder Kundenfreundlichkeit seitens eines Assistenten, der von einem Autor direkt angesprochen wird aus, um die Ordnung zu zerstören. Da es außerdem

keine besonderen Schutzmechanismen gibt, hindert auch einen Autor nichts daran, –in Umgehung der dafür Zuständigen, seine Seiten selbst zu erstellen.

Da die Struktur der Webseiten nun nicht mehr zentral kontrolliert wird, tritt auch das

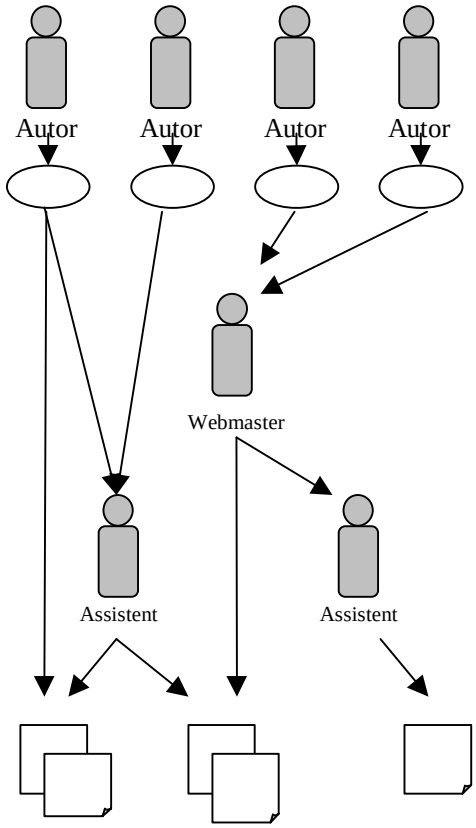



**Abb. 3. Probleme beim Webpublishing**

Problem auf, daß Links innerhalb des eigenen Webservers nicht mehr logisch gesetzt werden: Da der erste Assistent nicht unbedingt weiß, welche Dokumente der zweite erstellt hat, kann er diese auch nicht bei anderen Dokumenten berücksichtigen.

Folge: Die Linktiefen der Dokumente werden immer größer, während etwaige zusammengehörige Seiten, die aufeinander referenzieren sollten, dies nicht tun, da sie von unterschiedlichen Autoren bzw. Webmastern erstellt wurden, die das jeweils andere Dokument nicht kannten.

Mit der Erstellung von immer mehr Seiten wurde nach einiger Zeit ein weiteres großes Problem deutlich: Seiten veralten bzgl. Inhalt und Design. Inhalte, besonders bei wissenschaftlichen Publikationen oder bei aktuellen Nachrichten, sind nur eine gewisse Zeit lang gültig. Erfolgt jedoch von Seiten der Autoren keine Kontrolle, werden die inzwischen veralteten Dokumente nie ersetzt. Im Gegensatz zur Publizierung auf Papier, bei der eine nachträgliche Aktualisierung von bereits veröffentlichten Werken nicht ohne weiteres möglich ist, sind bei der Webseite die technischen Möglichkeiten vorhanden. Außerdem beeinflußt die Publizierung eines Dokumentes im Web beeinflußt auch das Image des beteiligten Unternehmens bzw. des Instituts: Ein Institut oder Unternehmen, das auf ein gutes Erscheinungsbild in der Öffentlichkeit Wert legt, darf nicht den Fehler machen, veraltete und längst überholte Informationen anzubieten.

Entsprechendes gilt auch für das Design, also das optische Erscheinungsbild der Webseiten. Waren Anfang der 90er noch hauptsächlich rein textbasierte Webseiten zu finden, bei denen sich das Aussehen stark an die Papierausgabe hielt, hat sich dies inzwischen durch das Entstehen einer eigenen Branche für Webdesign geändert. So werden die Webseiten in Bezug auf

Layout und Benutzerfreundlichkeit inzwischen ähnlich bewertet wie herkömmliche grafische Programmoberflächen (GUI).

Entsprechend dieser Entwicklung im Bereich des Webpublishing läßt sich ein Schema für den Prozeß der Dokumentenentwicklung festlegen: Der Content-Lifecycle.

## 1.3. Der Content-Lifecycle

Der Content-Lifecycle beschreibt die Lebensphasen des Inhaltes eines Dokumentes.

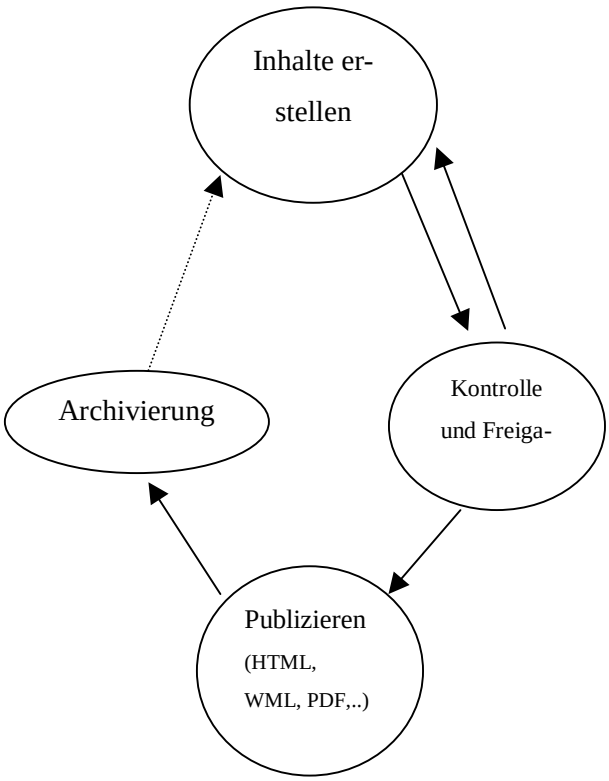



**Abb. 4. Content-Lifecycle**

Die Abbildung 4 zeigt die einzelnen Lebensphasen auf.

### 1.3.1. Inhalte erstellen

In dieser Phase erstellen die Autoren die unterschiedlichen Objekte (Assets) aus denen die spätere Seite bestehen soll. Das können Texte sein, aber auch Grafiken, Videos oder andere Medienformate. Dies bezeichnen wir als den „Content“.

Bei der Erstellung des Contents nutzt man auch die gezielte Recherche nach vorhandenen Informationen. Es ist daher zu hinterfragen, welche Informationsquellen am häufigsten genutzt werden. Enthalten bei diesen Informationsquellen ist auch das eigene Archiv und eigene Publikationen (hier: Publikationen des Unternehmens/Instituts) .

Eine andere Quelle zur Erstellung von Inhalten ist die Nutzung von Angeboten zur Content-Syndication. Unter Content-Syndication wird die Verbindung von Inhalten verschiedener Websites verstanden. Der Begriff Websites bezeichnet hierbei die Gesamtheit aller Seiten des Webauftritts einer Firma bzw. eines Institutes. Besonders interessant ist Content-Syndication für Websites, die ihr Angebot mit business- und branchenrelevanten Informationen aufwerten wollen. So zum Beispiel Börsenkurse, aktuelle Nachrichten, aber auch speziell ausgewählte Inhalte anderer Webseiten.

Grundlage für Content-Syndication ist das "Information-and-Content-Exchange-Protokoll" (ICE). Dieses bidirektionale

Protokoll versendet zur Übertragungssteuerung XML-basierte Nachrichten zwischen den beteiligten Servern. Es ist formatunabhängig, so daß sowohl Texte als auch Bilder oder XML-Daten übertragen werden können. ICE wird derzeit vom W3C für die Aufnahme als Standard geprüft [Web5].

Content-Syndication wird von Content-Providern betrieben. Ein Beispiel dabei ist die Firma iSyndicate [Web6].

### 1.3.2. Kontrollieren und freigeben

In dieser Phase wird der vorher erstellte Content auf inhaltliche und gestalterische Korrektheit (z.B. bei Bildern) geprüft. Dies geschieht im einfachen Fall durch den Ersteller des Contents selbst; besser jedoch ist hier die Einführung einer Hierachie, in der dann die Prüfung durch einen oder mehrere entsprechend autorisierte Mitarbeiter (Redakteure) erfolgen muß.

Bei nur einem überprüfenden Redakteur spricht man von dem 4-Augen-Konzept: Erst wenn der Inhalt durch 2 Personen (hier: Redakteur und Autor) akzeptiert wurde, kann die Seite zur Veröffentlichung weitergereicht werden.

Fällt die Prüfung negativ aus, wird der Content an den Autor zur Nachbearbeitung zurückgereicht.

### 1.3.3. Publizieren

Nachdem ein Dokument freigegeben wurde, wird es in dieser Phase in das gewünschte Format portiert und unter einer festgelegten Adresse gespeichert.

Der wesentliche Schritt in dieser Phase liegt darin, daß die Inhalte nun für die Öffentlichkeit[1] lesbar gemacht werden.

Je nach Art der Technik der Publikation und des verwendeten Verwaltungssystems ist diese Phase aufwendig: Bei einer Publikation im WWW reicht es nicht, allein eine neue Webseite zu genieren und diese irgendwo auf dem Webserver abzulegen; Referenzen müssen zumindest von dem Index der Website gesetzt werden, die einzelnen Teile der Webseite, welche neben dem reinen (HTML-)Text aus Bildern und anderen Objekten bestehen, müssen gespeichert werden und die Zugriffsrechte so gesetzt werden, daß das Zielpublikum die neue Webseite lesen kann.

Benutzt man als Autor oder Redakteur zur Veröffentlichung einer HTML-Seite lediglich einen Web-Editor, wie z.B. Microsoft Frontpage [Web7] oder Allaire Homesite [Web8] werden viele der oben angerissenen Aufgaben durch Site-Management-

[1] Die Öffentlichkeit ist hierbei definiert auf den Ort der Publikation. Es kann sich dabei um das Internet, ein Intranet oder ein anderes Netz oder Verteiler handeln. Zu Beachten ist, daß Inter- und Intranet nicht auf World Wide Web beschränkt sind, sondern auch Dienste wie Usenet und Mail mit einschließen.

Funktionen bewältigt. Jedoch reichen diese Funktionen bei weitem nicht aus, um komplexe Websites, die aus mehreren Tausend bis Millionen Objekten bestehen und spezielle Kontroll- und Freigabezyklen benötigen, zu verwalten.
Diese Aufgabe kann nur von einem Web-Content-Management-System geleistet werden.

### 1.3.4. Archivieren

Da Informationen veralten, muß sichergestellt werden, daß diese nach einer festgelegten Zeit automatisch archiviert oder aber einer Aufarbeitung zugeführt werden. Ebenso wie beim Publizieren muß beim Archivieren dafür gesorgt werden, daß die Inhalte auch später noch zu Recherchezwecken zugreifbar sind. Gleichzeitig, und dies ist einer der wichtigsten Punkte bei der Archivierung, ist darauf zu achten, daß vorhandene Referenzen aus noch aktuellen Seiten durch die Archivierung nicht ungültig gemacht werden, bzw. zu einem Fehler führen.
Bei kleineren Unternehmen oder Instituten spielt die reine Archivierung keine große Rolle, so daß die Seiten unter Umständen nicht gesichert, sondern gelöscht werden.
Dies geschah auch beim traditionellen Webpublishing, was sich in vielen Fällen später als großer Fehler erwies: Durch das Löschen ging Wissen verloren, auch wenn dieses zu dem Zeitpunkt der Löschung als veraltet und unwichtig erscheint.
Doch gerade auf dem Gebiet der wissenschaftlichen Publikationen ist ein Rückblick auf alte Publikationen von großer Bedeutung und Wichtigkeit für die Einhaltung der (natur-)wissenschaftlichen Methode [Lit5].
Ein weiterer Aspekt, der die Wichtigkeit von Archiven betont, ist auch hier die „Gesichtswahrung“ eines Unternehmens im Falle eines Notfalles. Der Imageschaden aufgrund eines Hackereinbruchs bei dem der Internet-Auftritt geschädigt wird, kann durch das Einspielen einer Backup-Sicherung klein gehalten werden.

## 1.4. Content-Management-Systeme

Unter einem Content-Management-System (CMS) versteht man ein System zur Verwaltung und Administration von Inhalten in Bezug auf den Content Lifecycle. Dabei gilt das Grundprinzip der Trennung von Design und Inhalt. CMS benutzen dabei zusätzlich die Mechanismen des Dokumenten-Managements.
Reine CMS müssen dabei nicht unbedingt etwas mit Webpublishing zu tun haben. Aufgrund der gemeinsamen Mechanismen werden Web-Content-Management-Systeme

(WCMS) und CMS oftmals als gleich betrachtet. Bei einem WCMS liegt jedoch, im Gegensatz zu einem reinen CMS, der Schwerpunkt auf dem Webpublishing. In der Praxis wird dieser Unterschied dadurch deutlich, das bei einem CMS der Datenfluß unidirektional ist.

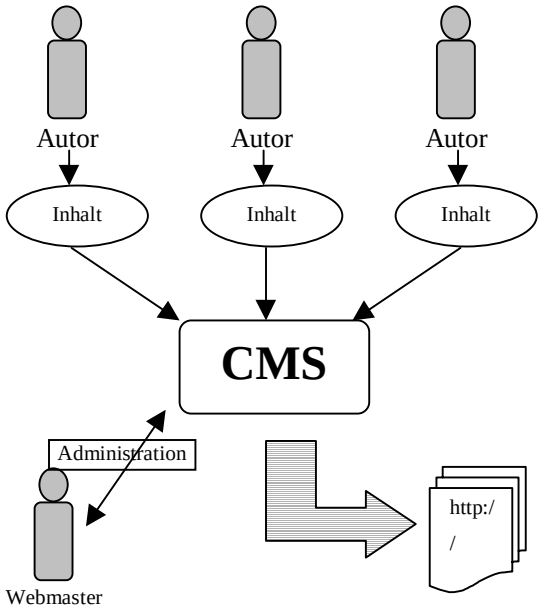



**Abb. 5. CMS**

In anderen Worten: Beim herkömmlichen CMS werden Webseiten zwar erstellt, jedoch erfolgt keine Interaktion mit Inhalten von bereits vorhandenen Webseiten. Die Abbildung 5 zeigt dies. Die Verwaltung und Administration der von den Autoren erstellten Inhalte wird durch das CMS geleistet und daraus Webseiten erstellt. Eine Rückführung und Nutzung von Inhalten von vormals erstellten Webseiten ist jedoch nicht vorgesehen und nur über die persönliche Recherche der Autoren möglich.

Technische Aspekte, wie das Link-Management, werden in der Regel nicht als Gegenstand des CMS bezeichnet, sondern obliegen dem darauf spezialisierten WCMS.

Aufgrund der Tatsache, daß WCMS ein neues Gebiet ist und viele Hersteller von Systemen zum Dokumenten-Management (DM) als erste auf diesem Gebiet Entwicklungen anboten, werden diese Begriffe oft vertauscht.

Zur genauen Begriffsbestimmung führen wir daher an dieser Stelle folgende Definitionen ein:

**Dokumenten-Management (DM):** DM ermöglicht den Dokumentenaustausch. Es bietet sowohl die Archivierung verschiedenster Dokumenttypen (HTML, PDF, ASCII, WORD,...) in einer Netzumgebung, als auch den direkten Zugriff auf diese Dokumente.

**Knowledge Management (KM)**:
KM entspricht dem Wissensaustausch. Es wird auf die Methoden des CM zurückgegriffen, um Wissen zu archivieren und wieder verwendbar zu machen.

**Content-Management (CM):** CM definiert die Art der Informationspräsentation. Als Grundlage gilt das Prinzip der Trennung von Inhalt und Design. CM bezieht

sich auf das Verwalten und Administrieren des Lebenszyklus von Inhalten.

**Redaktionssysteme:** Ursprung von Redaktionssystemen ist die Welt der Printpublikationen und Verlage. Sie bieten daher oft Funktionen des Cross-Media-Publishing, das heißt Funktionen die es erlauben verschiedenste Medientypen zu kombinieren und zusamenzufassen. Da dies auch im starken Maße Teile von WCMS und CMS sind, sind die Übergänge zwischen diesen Systemen fließend.

In vielen Publikationen und Werbebroschüren werden aus oben genannten Gründen CMS, WCMS und Redaktionssysteme als Synonyme für das selbe System verwendet.

## 1.5. Kosten von WCMS

Obwohl durch die Entwicklung von großen Webseiten, wie sie in Kapitel 1.2 aufgezeigt wurde, die Notwendigkeit von WCMS deutlich machte, muß eine Abschätzung darüber gemacht werden, für wen sich ein komplexes WCMS lohnt.
Die Anschaffung eines WCMS ist zuallererst eine Kostenfrage. Die auf dem Markt erhältlichen Systeme zeichnen sich derzeit sowohl durch einen hohen Anschaffungspreis, als auch durch eine unklare Lizenz-Politik aus. Während man bei herkömmlichen Standardanwendungen für ein Produkt einen festen Preis pro Lizenz erhalten kann, ist der Preisvergleich über WCMS schwierig: Fast jeder Anbieter hat zur Berechnung des Preises ein eigenes Modell entworfen. Bei einem Anbieter hängt der Preis von der Anzahl der Benutzer ab, bei einem anderen von der Anzahl der CPUs. Andere wieder berechnen nach dem Traffik auf der Website oder wieder andere über eine Mischkalkulation obiger und anderer Punkte. Bei vielen Systemen kommen zudem neben den eigenen Kosten auch etwaige Lizenzgebühren für die Nutzung von Datenbanken wie z.B. ORACLE hinzu.
Bsp.: Der VIP'ContentManager von Gauss Interprise [Web9] kostet einen Betrag, der sich aus der Summe für das eigentliche Redaktionssystem und einem Betrag für jeden Autor-Arbeitsplatz, sowie einem Betrag für optionale Zusatzmodule, zusammensetzt.

Weiter treten bei einigen WCMS zusätzlich indirekte Kosten auf für:

- Schulung von Mitarbeitern.
- Konvertierung des vorhandenen Bestandes der Website in das WCMS.
- Definition und Organisation einer Hierachie zur Kontrolle und Freigabe von Dokumenten innerhalb der beteiligten Betriebsstruktur.
- Umbau der vorhandenen Serverarchitektur zur Anpassung an das WCMS. Bsp: WCMS, welche den Webserver von Lotus Notes voraus-

setzen, können nicht auf einem vorhandenen Solaris-Apache-Mix eingesetzt werden.

Im Durchschnitt belaufen sich die Preise für ein dem jeweiligen Unternehmen angepaßtes WCMS zwischen 20.000 DM und 150.000 DM ohne Berücksichtigung der Kosten für Hardware und Mitarbeiter[2].
Eine detaillierte Vergleichsstudie über WCMS-Hersteller und CM-Lösungen ist käuflich ab 999 DM (zzgl. Mehrwertsteuer) erhältlich bei New Media Sales [Web10]. Eine weitere findet sich bei der Firma Co-Co Software Engineering und ist für 990 Euro (zzgl. Mehrwertsteuer) zu erwerben [Web11].

Die Entscheidung zum Einsatz eines WCMS läßt sich aufgrund der oben angegebenen Unübersichtlichkeit bei den Kosten besser über einen Umkehrschluß ziehen: Wann braucht man kein WCMS ?
Antworten läßt sich auf diese Frage mit folgenden Punkten:

- Die Website ist klein oder überschaubar.
- Es liegen klare Zuständigkeiten vor und die Hierachiestrukturen werden eingehalten.
- Die Veröffentlichungsrate neuer Dokumente ist gering.
- Das Unternehmen besteht aus wenigen, dafür fachkundigen Mitarbeitern.

Gerade letzter Punkt hat ein großes Gewicht: In einem Unternehmen in dem hochqualifizierte Mitarbeiter arbeiten, werden oft heterogene Softwarelösungen für jeden Mitarbeiter eingesetzt.
Bei WCMS besteht jedoch die Gefahr, daß dieses von den Mitarbeitern abgelehnt wird, da sie oft nur bestimmte Schnittstellen oder Benutzer-Interfaces zulassen welche subjektiv jedoch nicht die selben Vorteile bieten wie die bisher eingesetzten Programme.

## 1.6. Vor- und Nachteile von WCMS

Im Folgenden fassen wir die Vor- und Nachteile von WCMS in Stichworten zusammen:

+ Automatisches Linkmanagement
+ Unterstützung des Content-Lifecycle
+ Trennung von Inhalt und Design: Autoren benötigen keine HTML-

[2] Die Abschätzung dieser Preise beruht auf einem Vergleich von WCMS-Produkten wie sie auf der CeBIT 2000 angeboten wurden.
Darin enthalten sind die Produkte der Firmen: Gauss Interprise, Higher-Order, Vignette, EDM, PANSite, mks – Mortice Kern Systems, Roxen Internet Software, Imperia, Interleaf, PC-Ware, Infopark AG, Media Artists, SchemaText, NetUse Kommunikationstechnologie AG, POET Software, Herrlich & Ramuschkat, dimedis, mindfact.

Kenntnisse mehr

+ Layout für alle Webseiten festgelegt und somit nachträgliche Änderungen des Erscheinungsbild der Webseiten durch das WCMS möglich
+ Festgelegte Kontroll- und Freigabezyklen
+ Zeiteinsparungen bei Wartung und Pflege der Inhalte
- Hoher Anschaffungspreis
- Erfordert festangestellten Mitarbeiter zur Administration und Wartung
- Portierung von alten Webseiten auf das neue System problematisch

Ein Merkmal zur Unterscheidung der Kosten für und ohne WCMS bieten auch Statistiken, die von der Fachpresse erbracht wurden.

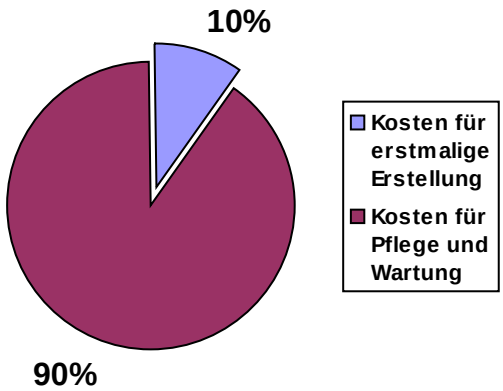



**Abb. 6. Kosten beim herk. Webpublishing**

Die Abbildung 6 zeigt die Kostenverteilung beim herkömmlichen Webpublishing. Sie basiert auf einer europaweiten Umfrage unter 1000-Topmanagern die 1999 von den Firmen Sqribe Technologies und Deloitte & Touche durchgeführt wurde [Art4]. Abbildung 6 zeigt das Ergebnis der Umfrage in Bezug auf die Kosten für Erstellung und Wartung.

Im Vergleich dazu stehen die Kosten für den Betrieb bei einem WCMS. Hierzu führte die LAN Times [Web12] 1999 eine Analyse[3] durch, dessen Ergebnisse für die Bereiche Erstellungskosten und Kosten für Pflege und Wartung in Abbildung 7 zu sehen ist.

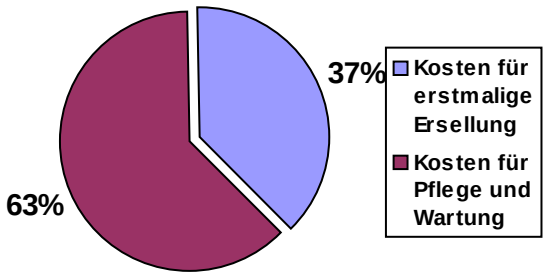



**Abb. 7. Kosten beim WCMS**

Vergleicht man beide Abbildungen, so erkennt man, daß –wie erwünscht- die Kosten für die Wartung stark abgenommen haben.

Da seit der Analyse der Markt für WCMS und Content-Syndication stark gewachsen ist, dürften sich die Kosten noch weiter verschoben haben, so daß der prozentuale

[3] Die Kostenverteilung der Analyse von LAN Times beinhaltete neben den Kosten für Erstellung und Pflege/Wartung noch weitere Kosten für Hard- und Software und Sonstiges. Zum besseren Vergleich der beiden Statistiken beschränkt sich Abb. 7 aber nur auf die anteiligen Kosten für Erstellung und Pflege/Wartung.

Anteil der Kosten für Pflege und Wartung weiter gesunken ist.

# 2. Komponenten eines WCMS

## 2.1. Allgemeines

In Kapitel 1 „Einführung“ wurde beschrieben, was WCMS bedeutet und wozu es eingesetzt wird. In diesem Kapitel wird erklärt, auf welchen Bestandteilen ein WCMS im Detail aufgebaut ist und welche Bedeutung diese haben.

Die wesentlichen Komponenten eines WCMS sind:

- **Benutzer- und Zugriffsverwaltung**
- **Redaktionelle Verwaltung (CM)**
- **Import- und Exportschnittstellen**

Zu beachten ist, daß alle Bestandteile miteinander kooperieren müssen. So ist z.B. die redaktionelle Verwaltung nicht vollständig, wenn sie nicht auf Funktionen der Benutzerverwaltung zugreifen kann.

Im Gegensatz zu der Arbeit von Oliver Zschau [Lit1] in der die 6 Komponenten

- Assetmanagement[4],
- Workflowkomponente,
- Benutzerverwaltung,
- Zugriffsverwaltung,
- Importschnittstellen,
- Exportschnittstellen

aufgeführt sind, werden hier diese Komponenten auf die 3 oben angegebenen verteilt. Dies begründet sich darin, daß in der technischen Umsetzung wesentliche Funktionen dieser Komponenten ineinander übergehen. Beispielsweise ist der Export für eine XML-Datei die inverse Funktion zu dem Import.

Ein anderes Beispiel ist die Zugriffsverwaltung, welche immer Bezug hat zu bestimmten Benutzern. Eine Zusammenführung von Benutzerverwaltung und Zugriffsverwaltung ist somit ein logischer Schritt.

## 2.2. Benutzer- und Zugriffsverwaltung

Da mit Hilfe eines WCMS festgelegt werden soll, wer das Recht hat ein Dokument zu bearbeiten, freizugeben, oder aber Einstellungen zu ändern, muß eine Benutzer- und Zugriffsverwaltung vorhanden sein.

Diese Verwaltung muß folgende Funktionen erfüllen:

- **Rechteverwaltung für Inhalte**. Festlegung, wer das Recht hat Inhalte zu bearbeiten. Dies bedeutet nicht nur das Recht überhaupt Inhalte von Seiten einbringen zu dürfen, sondern auch, vorhandene Sei-

[4] Unter Assetmanagement wird die Verwaltung der einzelnen Objekte aus denen eine Seite besteht verstanden. Obwohl die Bezeichnung „Asset“ ein üblicher Begriff auf diesem Gebiet ist, wird in dieser Arbeit das Wort „Objekt“ verwendet, da sich dieses bei der programmtechnischen Umsetzung durchgesetzt hat.

ten zu bearbeiten oder freizugeben. Gerade in Bezug auf die redaktionelle Verwaltung muß eine klar definierte Benutzerhierachie vorhanden sein, die auf ein entsprechend angepaßtes Rechte- und Dateikonzept aufbaut.

- **Rechteverwaltung für Formatvorlagen**. Festlegung, wer das Recht hat, Formatvorlagen (Design), also die Dateien, die das spätere Aussehen einer Datei bestimmen, zu bearbeiten. Ebenso wie für Inhalte muß auch bei den Formatvorlagen festgelegt werden, wer diese bearbeiten darf.
- **Gruppenverwaltung**. Dies betrifft die Verwaltung von Benutzergruppen und somit die Zugriffsrechte für diese Gruppen. Zur Unterstützung der Rechteverwaltung auf Inhalte und Design bietet die Definition von Benutzergruppen, ähnlich wie bei dem UNIX-Dateisystem, die Möglichkeit die Rechteverwaltung zu vereinfachen.
- **Verwaltung von Benutzer-Aufgaben und Benachrichtigungen**. Durch die redaktionellen Workflows bei denen zum Beispiel im 4-Augen-Konzept die Überprüfung eines neuen Dokumentes durch einen Redakteur erfolgt, muß dieser eine Benachrichtigung erhalten, sobald der Autor das neue Dokument zur Rezension weitergibt. Dies wird durch eine Aufgaben-Verwaltung (auch als „ToDo“-Verwaltung bezeichnet) unterstützt.
- **Administrationsfunktionen**. Durch Administrationszugänge oder spezielle Rechte müssen besonders berechtigte Personen in der Lage sein, neue Benutzer anzulegen, Rechte zu ändern, oder Benutzeraccounts zu löschen.

In der Regel lassen sich Teile dieser Funktionen durch vorhandene Systemfunktionen, bzw. durch den Zugriff auf vorhandene Standardprogramme und Datenbanken abdecken. So gehört es inzwischen zum Standard, das Schnittstellen z.B. zu einem LDAP-Server oder einem X500-Server vorhanden sind, mit dem vorhandene Benutzerrechte übernommen werden.
Schnittstellen zu einer Kalenderverwaltung, wie z.B. Groupwise oder dem CDE CalenderManager sind auch an dieser Stelle hilfreich.

## 2.3. Redaktionelle Verwaltung

In Abbildung 4 wurde anhand des Content-Lifecyles bereits Einblick in die wesentliche Aufgabe der redaktionellen Verwaltung gegeben. Sie besteht daher im Wesentlichen in der technischen Abbildung und Unterstützung des Content-Lifecycles und

somit des Content-Managements in ein WCMS.

Im Folgenden werden wir uns die einzelnen Schritte des Content-Lifecycles genauer anschauen.

### 2.3.1. Seitenerstellung

In der Abbildung 4 begann der Content-Lifecycle damit, daß die Inhalte, das heißt die Objekte, einer Seite von einem Benutzer neu eingegeben oder durch die Referenzierung auf vorhandene Objekte anderer Dokumente angegeben wurden.

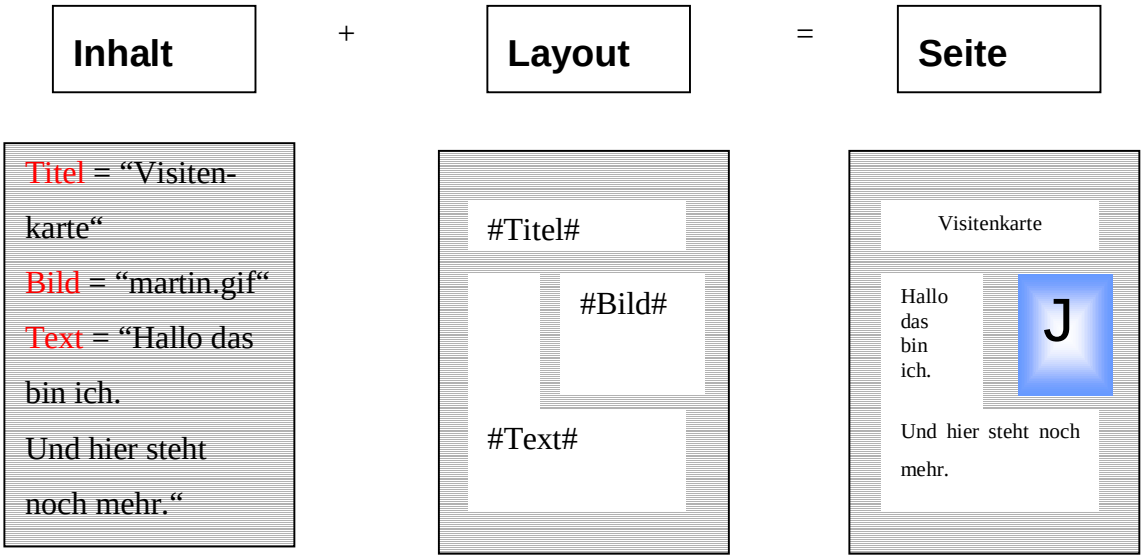



**Abb. 8. Seitenerstellung**

Die Aufgabe des CM besteht im ersten Schritt danach darin, diese Inhalte mit dem Design zusammenzuführen, um daraus eine neue Seite zu machen.

Die Abbildung 8 zeigt dieses Verfahren schematisch. Anhand einer vorgegebenen Formatvorlage (Layout) wird das spätere Aussehen der Seite bestimmt. In dem Layout werden die Stellen, an denen die Inhalte eingefügt werden sollen, durch Schlüsselworte (engl „Keywords“) festgelegt. Damit das spätere Seitenaussehen vollständig ist, müssen alle Keywords durch Inhalte (Objekte) repräsentiert werden[5]. Aus diesem Grund wird bei der Eingabe der Inhalte ebenfalls auf diese Keywords zugegriffen. Ein Satz an Keywords, sowie das dazugehörige Layout definieren wir als **Dokumententyp**.

Ein Dokumententyp kann auch kein, d.h. ein leeres Layout besitzen. Dies ist dann der Fall, wenn die Zielseite nur aus einem Objekt , wie zum Beispiel einem Bild, besteht. (Neben den Keywords und dem Layout enthält der Dokumententyp in der späteren Implementierung auch interne Angaben über die Art und Speicherung der einzelnen Objekte.)

[5] Ausnahme sind Default-Werte, die benutzt werden, wenn Eingaben für Seiteninhalte optional sind.

Die Benutzer- und Zugriffsverwaltung kann den Zugang zur Seitenerstellung über diese Dokumententypen regeln: Die Frage ob ein Benutzer eine Seite erstellen darf, ist davon abhängig, ob er den Dokumententyp für die Seiten nutzen darf.

### 2.3.2. Kontroll- und Freigabe-Zyklus

Wie in Kapitel 1.3.2. „Kontrollieren und freigeben“ erklärt wurde, muß bei der Erstellung einer neuen Seite ein redaktioneller Workflow durchgeführt werden. Dieser Vorgang ist ein Zyklus, der unter Umständen mehrere Male abgearbeitet werden muß, bevor endlich eine Seite zur Veröffentlichung freigegeben wird.

Da bei der Kontrolle einer neuen Seite nicht nur die Überprüfung des Inhaltes eine Rolle spielt, sondern auch, ob das gewählte Design richtig ist, sollte eine Voransicht (engl. „Preview“) der Seite möglich sein. Aus diesem Grund ist die eigentliche Seitenerstellung vor der eigentlichen Seitenfreigabe aufgeführt. Dies bedeutet jedoch nicht, daß eine Seite bereits öffentlich abrufbar ist. Es bleibt Aufgabe des Freigabemechanismus zu bestimmen, wo und wie die endgültige Seite gespeichert wird[6]. Sollte die Voransicht auf eine fertiggestellte Seite zugreifen müssen, so ist diese in einem temporären Bereich abzulegen, der nicht öffentlich referenziert wird.

Die Abbildung 9 zeigt den Kontroll- und Freigabe-Zyklus in schematischer Form.

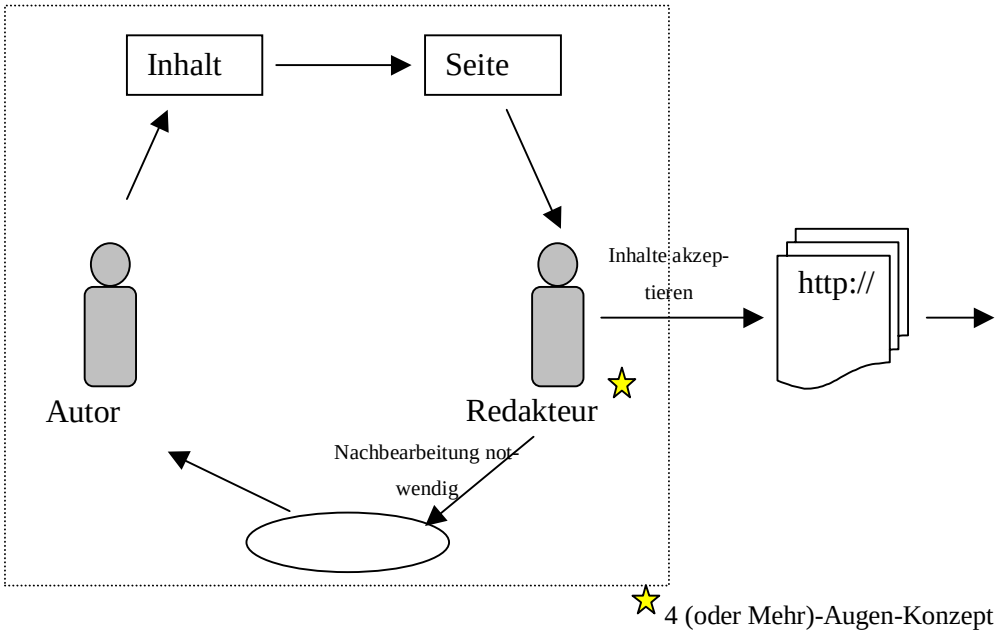



**Abb. 9. Kontroll- und Freigabe-Zyklus**

[6] Beachten Sie, daß die Art wie eine Seite gespeichert wird, hier, wie auch in den folgenden Absätzen noch nicht festgelegt wird. Eine Seite kann sowohl als eigene Datei, als auch in einer Datenbank gespeichert werden.

Man sieht, daß die entgültige Seite erst öffentlich publiziert wird, nachdem sie von dem Redakteur akzeptiert wurde.

Es ist zu beachten, daß die Regeln wie Seiten akzeptiert werden, hier nicht aufgezeichnet sind. So ist es möglich und sinnvoll, daß vor diesem Zyklus noch ein weiterer eingebaut wird, bei dem der Autor und der Redakteur eine Person sind. Der Autor kann in diesem Modus die Seite mehrfach bearbeiten, bevor er sie zum eigentlichen Kontroll- und Freigabe-Zyklus weitergibt.

Der rekursive Einbau weiterer Zyklen ist ebenfalls eine Möglichkeit die nicht außer Acht gelassen werden sollte. Beispiel: Inhalte einer Seite werden von einem Reporter in einem ersten Zyklus erstellt. Die Seite wird daraufhin von einem Redakteur übernommen, in einem zweiten Zyklus kontrolliert, ggf. verändert und dann wiederum an einen Chefredakteur weitergeleitet, der letztendlich die Seite zur Veröffentlichung freigibt.

Eine andere Regel bei Freigabe kann die Berücksichtigung von Gruppenarbeit an einem Dokument einschließen. So könnte die Regel sein, daß ein Dokument erst dann freigegeben wird, wenn mindestens 50% aller Autoren diesem zustimmen. Oder aber die Seite wird abgelehnt und der Korrektur (Revision) zugeführt, wenn nur einer der Redakteure sie ablehnt, obwohl bereits andere der Seite zugestimmt haben.

Die Definition solcher Regeln wird im allgemeinen die Aufgabe des Käufers bzw. des Administrators eines WCMS sein. Vordefinierte Regeln sollten jedoch Bestandteil jedes WCMS sein.

### 2.3.3. Versionskontrolle und Archivierung

Nach der Freigabe einer neuen Seite durch den Kontroll- und Freigabe-Zyklus wird diese veröffentlicht. Dies bedeutet in der Regel, daß die Seite, wie sie für die Voransichtsfunktion bereit gestellt wurde, in ihrer endgültigen Position gespeichert wird.

Es sind jedoch auch weitere Aufgaben nötig: Die neue Seite muß indiziert werden, etwaige Referenzen auf andere Seiten und Objekte müssen gesetzt und ebenfalls indiziert werden, und eine Versionskontrolle muß durchgeführt werden.

Nachdem die Seite gespeichert und veröffentlicht wurde, ist die Arbeit aber nicht getan: Jeder Inhalt veraltet nach einiger Zeit oder Referenzen auf genutzte Objekte oder auf andere Seiten können ungültig werden. Anhand einer Indizierung für alle in der Seite enthaltenen Referenzen ist eine Kontrolle und Änderung dieser automatisch durchführbar. Bei Inhalten jedoch ist dies nicht möglich und eine Nachbearbeitung durch einen Menschen erforderlich.

In Abbildung 10 sehen wir, wie eine Seite der Version 1.0 nach einiger Zeit inhaltlich veraltet. Wie die Veralterung eines Seiteninhaltes vorgeht, ist nicht fest definiert. Üblich ist die Implementierung einer Lebenszeit für eine Seite und/oder von einzelnen Objekten innerhalb der Seite. Ist diese Lebenszeit, die ab Erstellung oder der letzten Änderung gilt, abgelaufen, gilt die Seite als veraltet.

Bevor die Seite jedoch in das Archiv geschoben wird, kann eine manuelle Nachbearbeitung erfolgen in der die Seite auch den Kontroll- und Freigabe-Zyklus neu durchlaufen wird. Mit der neuen Freigabe und somit der Neu-Veröffentlichung wird die Versionsnummer automatisch erhöht.

Auch der neue Inhalt veraltet nach einer gewissen Zeit. Wenn jedoch keine Neubearbeitung erfolgt, wird die Seite ins Archiv geschoben. Die Folgen hiervon sind je nach Implementierung des WCMS unterschiedlich. Allgemein kann man sagen, daß ab dem Zeitpunkt ab dem eine Seite als veraltet gilt, keine neuen oder geänderten Dokumente mehr auf diese Seite referenzieren dürfen – es sei denn als Referenz auf die Position im Archiv.

In den meisten Fällen wird eine Seite, die in das Archiv geschoben werden soll, erst dann wirklich gelöscht, wenn keine Referenz mehr zu der Seite hinführt. Dies wird dadurch erreicht, daß alle Referenzen anderer Seiten auf die veraltete Seite geändert oder gelöscht werden.

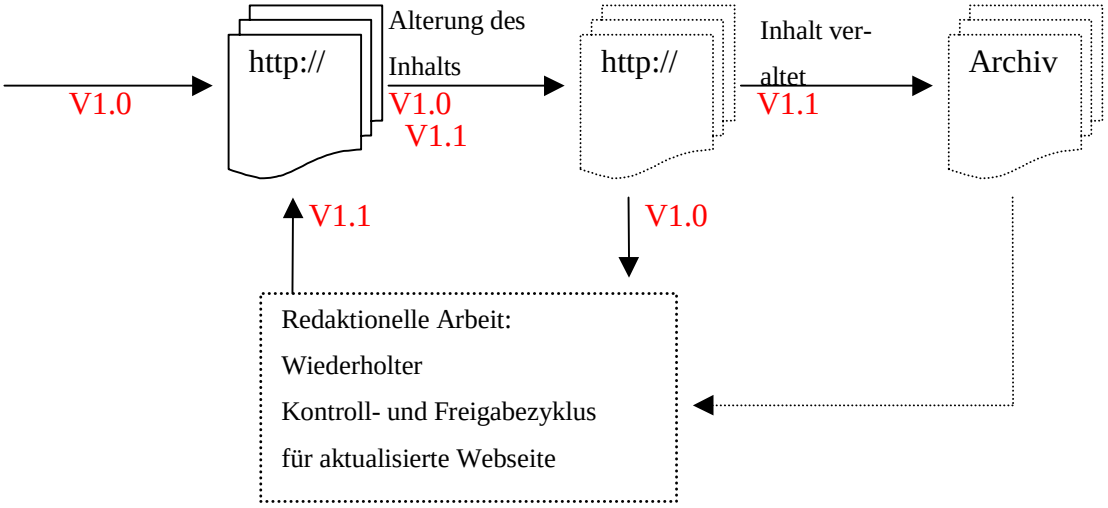



**Abb. 10. Versionskontrolle und Alterung**

In Bezug auf das Verfahren eine feste Lebenszeit für Seiten oder Objekte zu definieren gibt es eine Erweiterung, die nicht unerwähnt bleiben sollte: Die der **Halbwertszeit**.

Im Normalfall, bei dem eine feste Lebenszeit festgelegt wurde, wird mit Ablauf dieser der Autor über die Aufgabenverwaltung informiert. Er hat dann die Aufgabe, die Seite entweder ins Archiv zu schieben oder

sie wieder zu publizieren. Dieses Verfahren hat den Nachteil, daß keine Vorwarnung erfolgt und mit dem Zeitpunkt der Veralterung bis zur Abarbeitung des Autors Zeit vergeht die unter Umständen zu lang sein kann.

Bei dem Konzept der Halbwertszeit wird dieser Nachteil dadurch gemildert, daß der Autor bereits vor Ablauf der Lebenszeit der Seite Nachrichten erhält: Bei der ersten Speicherung einer Seite bzw. einer Änderung wird die Halbwertszeit auf die Hälfte ihrer Lebenszeit gestellt.

Mit Ablauf der Hälfte der Lebenszeit, d.h. der Halbwertszeit, erhält der Autor die erste Erinnerung. Der Autor hat dann noch genug Zeit sich auf die Veralterung einzustellen bzw. kann bereits vor diesem Zeitpunkt Änderungen machen.

Wird nach Ablauf der Halbwertszeit keine Änderung gemacht, wird diese halbiert und die nächste Alarmierung erfolgt nach Ablauf der neuen Halbwertszeit.

Die Halbwertszeit wird so lange verkleinert, bis diese eine Mindestgröße nicht unterschreitet.

## 2.4. Import- und Exportschnittstellen

Wie bereits im einführenden Kapitel 2.2 „Benutzer- und Zugriffsverwaltung“ angedeutet ist die Interaktion mit anderen Programmen und Systemen für ein WCMS in Bezug auf Benutzerfreundlichkeit sehr hilfreich. Ein WCMS wird niemals in seinen einzelnen Funktionalitäten, wie z.B. die Benutzer- oder die Aufgabenverwaltung, so umfangreich sein, wie Programme die nur für diesen Zweck erstellt wurden. Demgemäß sind Schnittstellen und ein modularer Aufbau Kennzeichen für ein gutes WCMS.

Folgende Schnittstellen gehören derzeit zum Standard:

- Import- und Export von HTML- und XML-Dokumenten.
- Import bestehender Sites.
- Import bestehender Datenbestände für Benutzer-, Aufgaben-, Link- und Dokumentenverwaltung.

Folgende Punkte zeichnen bereits komplexere WCMS aus:

- Import und Export bestehender Datenbestände für Inhalte. (Beispiel: Helpdesk-Systeme, Shopsysteme).
- Export-Schnittstellen für vorhandene lokale Suchmaschinen, wie z.B. die HARVEST-Suchmaschine [Web13].
- Schnittstellen zur Steuerung des jeweiligen Webservers.
- Autonome Überwachung und Import von statischen Webseiten auf der Website, die auch von anderen Applikationen (auch anderer WCMS) bearbeitet werden.

- Export von Seiten in Dateiformaten für PDF, Winhelp, Unix Manuals, WORD.
- Unterstützung von Web-Techniken (ASP, JavaScript, Java, CGI, SSI, Embedded Perl, PHP).

Folgende Schnittstellen werden derzeit von keinem WCMS unterstützt, werden voraussichtlich aber in Zukunft ebenfalls Teil komplexer Lösungen sein:

- Automatischer Eintrag neuer Seiten in ausgewählte Suchmaschinen wie Altavista, Google oder HotBot [Web16 – Web18].
- Automatische Verwaltung der vom Webservern und Suchmaschinen-Agenten genutzten Datei „ROBOTS.TXT“.
- Steuerung und Verwaltung von Up- und Downloadbereichen für Dateien beliebiger Medientypen.

Als Standard für Schnittstellen gibt es derzeit Bestrebungen von Softwareherstellern die Sprache XML (Abk. für Extensible Markup Language) einzusetzen [Web14]. Microsoft™ [Web15] geht dabei sogar soweit, dies als Schnittstelle zwischen allen Applikationen innerhalb eines neuen Betriebssystems zu implementieren, welches als Nachfolgeprodukt zu Windows 2000 auf dem Markt kommen soll.

Genaueres zu XML im kommenden Kapitel 3.3.

# 3. Systemumgebung und verfügbare Techniken

## 3.1. Allgemeines

Zur Erstellung eines WCMS benötigt man Wissen darüber, in welcher Umgebung diese Applikation zu laufen hat und welche Techniken zur Verfügung stehen.

Da die Aufgaben eines WCMS im Wesentlichen Verwaltungs- und Steuerungsaufgaben für den Zugriff auf Webseiten darstellen (genauer: Direkte Anwendungsunterstützung und Auslösen von Funktionen), bewegen wir uns im Application Layer des ISO-Referenzmodelles.

Aus diesem Grund beschränken wir uns im Folgenden auf die Erklärung der Techniken und Applikationen aus diesem Layer.

## 3.2. Webserver und Betriebsystem

In Abbildung 1 in Kapitel 1.1 „Das World Wide Web“ wurde bereits eine Statistik angegeben über die weltweite Verteilung der Webserver. Demnach benutzen die meisten, nämlich zusammengezählt 79 Prozent[7], Webserver Apache [Web19], Microsoft-IIS [Web15] oder Netscape Enterprise [Web20].

Interessant an der Statistik ist, daß der Webserver von Lotus in der weltweiten Verteilung auf nur 0.23% kommt [Web21]. Dies ist deswegen von Bedeutung, weil viele WCMS mit Lotus Domino [Web22] laufen. (Beispiele: EDM Domis V3.0 [Web23], das WCMS der Deutschen POST AG). Die Nutzung von Lotus Domino als Plattform für WCM-Lösungen begründet sich vor allem in den guten Zugriffsfunktionen dieses Webservers auf Datenbanken.

Mit Ausnahme von Speziallösungen wie dem ROXEN Webserver [Web24] (Verbreitung: 0.09%) bauen die meisten WCMS auf folgende Server auf: Apache, Microsoft-IIS und Lotus Domino. Basierend auf den Tests von Netcraft und Internet.com folgt ein Überblick über die Vor- und Nachteile dieser Server.

### 3.2.1. Apache

Der Erfolg von Apache im Vergleich zu den kommerziellen Produkten der anderen Anbieter liegt darin begründet, das der Source Code frei verfügbar ist und es einen sehr regen und freien Austausch im Support durch die Benutzer gibt. Die Autoren von Internet.com drücken es so aus:

> *„Apache users have come to rely on the server's rock-solid reliability,*

[7] Stand: Mai 2000. Der Anteil des Apache-Servers ist im Vergleich zu den Vormonaten kontinuierlich gestiegen. Microsoft-IIS hat in den vergangenen Monaten Anteile verloren, während der Anteil für Netscape Enterprise und sonstige Server gleich blieb.

*outstanding performance, and rich set of features. As its two closest competitors have found out, a brand name alone does not necessarily equate to a loyal customer following (Netscape SuiteSpot), nor does a plump pocketbook ensure market share (Microsoft IIS).”*

Die Vor- und Nachteile von Apache in einzelnen:

+ Preis (kostenlos)
+ Performance
+ Hohe Verfügbarkeit
+ Sicherheit
+ Unterstützung des HTTP1.1-Protokolls
+ Schneller Support über Apache-Newsgroups und Foren
+ Einfacher und logischer Aufbau der Konfigurationseinstellungen
- Keine MAC-Version vorhanden
- NT-Version derzeit noch fehlerbehaftet
- Keine grafische Oberfläche zur Administration
- Direkter Support durch Apache erfordert den Abschluß eines Beratungsvertrages

### 3.2.2. Microsoft-IIS

Die starke Verbreitung des Microsoft-IIS Webservers begründet sich vor allen Dingen darin, daß dieser direkt auf Funktionen des Betriebssystems Windows NT oder Windows 2000 zugreifen und entsprechende Schnittstellen nutzen kann.

Ein weiterer Grund liegt darin, daß viele kleinere Unternehmen, die sich einen Webserver anschaffen, meistens in ihrem Intranet Windows NT einsetzten. Der Umstieg, bzw. die Anschaffung eines stabilen UNIX-Servers erscheint daher zu dem Zeitpunkt des ersten Internet-Auftrittes den meisten Unternehmen als unrentabel, zumal diese Technik dann in dem Unternehmen noch weitgehend unbekannt ist.

Zu Vermerken ist auch, daß der neue IIS in Windows 2000 bereits enthalten ist.

Die Vor- und Nachteile lauten:

+ Indizierung von Seiten
+ Eingebautes Administrations-Programm
+ Einfache Konfiguration
- Keine UNIX-Version
- Schlechte Dokumentation neuer Funktionen
- IIS5.0 läuft nur unter Windows2000

### 3.2.3. Lotus Domino

Lotus Domino richtet sich vor allen Dingen an Benutzer von Lotus Notes und verfügt über Funktionen zur Unterstützung von Dokumenten-Management und Groupware-Applikationen, wie z.B. Groupwise.

Die Vor- und Nachteile sind:

+ Sehr gute Unterstützung zur Entwicklung für Groupware- und Dokumenten-Management
+ Optionale Sicherheitseinstellungen zur

Restriktion von Zugriffen

+ Versionen für viele unterschiedliche Betriebssysteme vorhanden
+ NNTP Unterstützung
- Teuer (1700 US$)
- Schlechte Performance bei Websites mit hohem Transfer
- Fehlerhafte Administrationswerkzeuge
- Schwer zu erlernen, und daraus:
- Es existieren nur wenige erfahrene Domino-Programmierer
- Viele Funktionen von Lotus-Notes-Anwendungen, die vom Server unterstützt werden, sind nicht Teil der Standards des W3Cs und erfordern somit einen eigenen Browser.

---

Hinsichtlich der Auslastung und Erreichbarkeit im laufenden Betrieb gibt es derzeit einige Daten, die dafür sprechen, daß der Webserver Apache unter dem Betriebssystem Sun Solaris derzeit das stabilste System ausmachen. Zu diesem Schluß kommt eine Analyse der Zeitschrift c't [Art5] und ein Statusbericht der Verfügbarkeit des Webservers der Universität Erlangen-Nürnberg [Art6].

## 3.3. Webtechniken- und Sprachen

### 3.3.1. Allgemeines

Ein WCMS muß einige Techniken unterstützen, um seinem Namen gerecht zu werden. Als erstes ist dies natürlich HTML. Die Seiten, die von einem WCMS gespeichert werden, liegen in der Regel im HTML-Format vor. Da jedoch inzwischen das Webdesign und interaktive Funktionen einer immer größere Rolle beim publizieren einnehmen, ist eine Unterstützung weiterer Techniken und Sprachen erforderlich. Prof. Gerhard Schub von Bossiazky sagte zu den Ergebnissen einer kürzlich veröffentlichten Studie über die Akzeptanz von Webseiten [Art7]:

> *„Der Einfluß der Informatiker auf die Gestaltung der Sites ist zu groß. Sie arbeiten immer noch mit der logischen und geradlinigen Baumstruktur, aber eine Website braucht wie jede gut gemachte Zeitschrift auch eine emotionale und assoziative Dramaturgie.“*

In den folgenden Kapiteln geben wir daher eine kurze Einführung in die Webtechniken, die derzeit die größte Verbreitung haben.

### 3.3.2. HTML

HTML ist die Abkürzung für „Hypertext Markup Language“. Sie beschreibt, wie der darzustellende Text strukturiert und

formatiert wird, und wie unterschiedliche Medientypen (wie Bilder, Grafiken, Töne usw.) in die Seite eingebunden werden können.

Die Grundsyntax von HTML wird in der HTML-Norm festgelegt, welche vom World Wide Web Consortium definiert wird [Web1].

Eine gewöhnliche HTML-Datei besteht grundsätzlich aus folgenden zwei Teilen:

- Header (Kopf - enthält Angaben zu Titel u.ä.)
- Body (Körper - enthält den eigentlichen Text mit Überschriften, Verweisen, Grafikreferenzen usw.)

Neben dem eigentlichen Text enthalten HTML-Dateien HTML-spezifische Befehle. Alle HTML-Befehle stehen in sog. Tags. Die Tags werden durch spitze Klammern markiert. Fast alle Befehle von HTML bestehen aus einem einleitenden und einem abschließenden Tag. Der Text dazwischen ist der "Gültigkeitsbereich" für die betreffenden Tags.

Beispiel:

```
<html>
<head>
    <title>Text des Titels</title>
</head>
<body>
Text, Verweise, Grafikreferenzen
usw.
</body>
</html>
```

**Quellcode 1. HTML**

Der gesamte Inhalt einer HTML-Datei, wie in Quellcode 1 angegeben, wird in die Tags <html> bzw. </html> eingeschlossen. Hinter dem einleitenden HTML-Tag folgt der einleitende Tag für den Kopf <head>. Zwischen diesem Tag und seinem Gegenstück </head> werden allgemeine Angaben, meist META-Daten, zur HTML-Datei notiert. Die wichtigste dieser Angaben ist der Titel der Webseite, die durch die Tags <title> bzw. </title> markiert wird.

Unterhalb davon folgt der Textkörper, kenntlich gemacht durch die Tags <body> bzw. </body>. Im Textkörper wird der eigentliche Inhalt der Seite notiert, also das, was im Anzeigefenster des Webbrowsers angezeigt werden soll.

Eine umfangreiche deutschsprachige Dokumentation von HTML wurde von Stefan Münz erstellt und ist Online abrufbar [Web25].

### 3.3.3. XML

XML steht für Extensible Markup Language. Mit Hilfe von XML lassen sich Kontext und Inhalt einer Seite fest definieren und maschinell auswertbar machen. Genauer:

> *„Dadurch, daß alle Dokumente den gleichen Regeln folgen, ist es möglich, Anwendungen zu schaffen, welche die Informationen aus den Dokumenten auslesen und automatisch verarbeiten.“* [Web14]

XML entstand aus SGML („Standard Generalized Markup Language“). Im Vergleich zu HTML bietet XML die Möglichkeit eigene HTML-Tags zu definieren und ist somit anpaßbar an die Bedürfnisse der Benutzer, bzw. an die Umgebungsdaten.

Ebenso wie HTML ist XML ein Standard, welcher vom W3C-Konsortium verwaltet wird.

Das folgende Beispiel zeigt eine typische XML-Struktur.

```
<?xml version=„1.0“ ?>
<!DOCTYPE gruss [
     <!ELEMENT gruss #PCDATA>
     <!ELEMENT inhalt #PCDATA>
]>
<gruss>Hallo XML!</gruss>
<inhalt>
Hier schreiben wir einen schönen
Text, der als „inhalt“ klassifiziert wird.
</inhalt>
```

**Quellcode 2. XML-Beispiel 1**

In der ersten Zeile des Beispiels wird die Nummer für die verwendete XML-Version angegeben. Darauf folgt in der nächsten Zeile die Angabe welcher Dokumententyp[8] zu verwenden ist. Dieser kann entweder – wie im Beispiel – direkt angegeben sein, oder aber in einer gesonderten Dokumententyp-Definition-Datei (DTD-Datei) ausgelagert sein.

Im der DTD wird angegeben, wie ein Tag zu interpretieren ist. In diesem Fall werden die beiden neuen Tags <gruss> und <inhalt> mittels der Angabe <!ELEMENT * #PCDATA> als Textfelder definiert.

Die XML-Spezifikation enthält weit mehr Element-Typen. So lassen sich alle HTML-Tags durch XML-Definitionen festlegen: XML ist somit eine Oberklasse von HTML.

Aufgrund der Erweiterbarkeit und Plattformunabhängigkeit von XML bietet es sich als universelles Speicherformat für Datensätze an. So zeigt das folgende Beispiel, wie ein Benutzerdatensatz aussehen könnte:

```
<?xml version=„1.0“ ?>
<!DOCTYPE user SYSTEM
„user.dtd“>
<user id=“1000“>
  <name>Wolfgang Wiese</name>
  <login>unrzc9</login>
  <email>Wolfgang.Wiese@rrze.uni-
erlangen.de</email>
</user>
```

**Quellcode 3. XML-Beispiel 2**

Ein weiterer Vorteil von XML ist der, daß die neuste Generation der Webbrowser (Internet Explorer 5.0, Netscape 6.0[9]) in der Lage sind, XML-Dateien zu lesen und entsprechend der DTD anzuzeigen.

### 3.3.4. CGI

CGI steht für Common Gateway Interface und definiert einen standardisierten Weg, Daten aus einem HTML-Formular an einen Webserver zu übertragen, welcher wieder-

[8] Achtung: Der hier angegebene Dokumententyp ist nicht zu vergleichen mit dem Dokumententyp, welchen wir in Kapitel 2 definierten.

um mit diesen Daten als Argument einen neuen Prozeß (CGI-Programm) startet.

Die Abbildung 11 zeigt dieses Schema. Aus einer HTML-Seite werden die Daten eines Formulars zuerst an den Webserver geschickt. Dieser ruft als nächstes ein CGI-Programm auf und übergibt diesem die eingegebenen Daten.

Das CGI-Programm verarbeitet diese Daten und übergibt dem Webserver als Antwort entweder Statusinformationen oder, im häufigsten Fall, den Content-Type und den Inhalt einer neuen Seite.

Da bei CGI nur die Art der Datenübergabe geregelt ist, ist die Programmiersprache, die für das CGI-Programm verwendet wird, nicht festgelegt. Das CGI ist somit weitgehend plattformunabhängig.

Da ein CGI-Programm auf dem Server ausgeführt wird, auf dem das aufgerufene CGI-Programm liegt, ist es unabhängig davon, welchen Browser oder welches Betriebssystem der Benutzer, also die Person, die es ausführt, verwendet. Ein einführendes Tutorial zu CGI befindet sich auf der Website von XWolf.com [Web26].

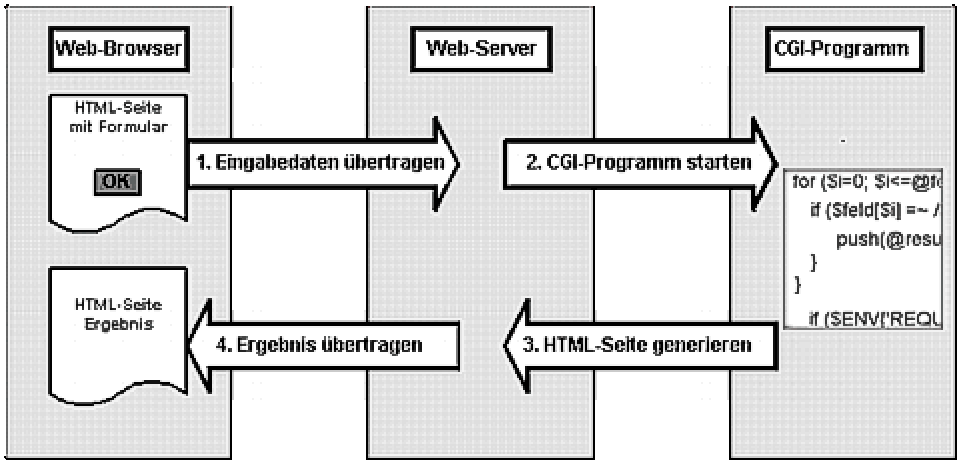



**Abb. 11. CGI**

Mit Hilfe von CGI-Programmen lassen sich komplexe Funktionen auf einer Website realisieren, angefangen bei dem einfachen ausfüllen und speichern eines Formulars, bis zu CGI-Programmsystemen mit denen Server konfiguriert und gesteuert werden können.

### 3.3.5. SSI

Mit Hilfe von SSI, d.h. „Server-Site Includes“ lassen sich Befehle innerhalb einer HTML-Seite einbauen, die vom Webserver ausgeführt werden. Im Normalfall handelt es sich bei diesen Befehlen um dynamische Ausgaben, wie z.B. die Systemzeit oder die Kennung des verwendeten Browsers. Es

[9] Netscape 6 ist bisher nur als Preview erhältlich, unterstützt und verarbeitet aber schon alle Standard-Dokumente.

lassen sich jedoch auch Programme starten und deren Ausgabe in die Datei einbetten.

Da SSI-Befehle vom Webserver ausgeführt werden, „sieht“ der Benutzer diese in der Regel nicht: Der Webserver verarbeitet diese Befehle zuerst und zeigt dann die Ergebnisse innerhalb der vom Benutzer angeforderten Webseite an [Web27].

SSI ist technisch als Ergänzung zu HTML zu verstehen, um den Zugriff auf CGI-Programmen zu vereinfachen. Ebenso wie bei CGI ist SSI unabhängig davon, welcher Browser oder welches Betriebssystem vom Benutzer genutzt wird.

### 3.3.6. PHP

Ähnlich wie SSI arbeitet PHP. PHP steht für „Hypertext Preprocessor“ und ist eine serverseitig interpretierte Skriptsprache, welche in den HTML-Quellcode eingebunden werden kann. PHP hat eine ähnliche Syntax wie C, Java und Perl, hat jedoch zusätzlich einige Features wie z.B. Kommandos zum Zugriff auf Datenbanken.

PHP kann sowohl über ein Modul in die Seite integriert werden als auch als CGI aufgerufen werden [Web28].

Der folgende Quellcode zeigt ein Beispiel, wie ein PHP-Aufruf in HTML eingebunden wird:

```
<html>
<head>
    <title>PHP-Beispiel</title>
</head>
<body>
<?
   echo „Hallo Welt!“;
?>
</body>
</html>
```

**Quellcode 4. PHP-Beispiel**

Bei der Anforderung der obigen Datei durch einen Benutzer wird der Webserver diese zuerst lesen, den PHP-Teil (im Beispiel blau markiert) interpretieren, d.h. den PHP-Teil durch dessen Ausgabe ersetzen und die Seite mit dem interpretierten Inhalt an den Benutzer weiterleiten.

Ein Tutorial, sowie weitere Informationen zu PHP finden sich auf der Webseite vom PHP-Center [Web29].

### 3.3.7. Java

Java wurde seit 1990 von SUN entwickelt und 1995 der Öffentlichkeit vorgestellt. Es handelt sich dabei um eine objektorientierte Programmiersprache mit deren Hilfe es möglich ist, interaktive Programme in Form eines „Applets“ in einem Browser ablaufen zu lassen. Ein Applet ist ein Programm, welches innerhalb der HTML-Seite eingebunden wird [Lit6].

Java ist plattformunabhängig: Das eigentliche Programm wird auf dem Quellrechner vorkompiliert und die Referenz darauf im HTML-Quellcode eingebunden. Bei einem Aufruf wird das Javaprogramm mittels der auf dem Ziel-Rechner befindlichen „Java Virtual Machine“ (JVM) lauffähig gemacht

und gestartet. Die JVM ist auf jedem Betriebssystem installierbar und verarbeitet den vorkompilierten Bytecode[10] des Quellrechners, um daraus einen Maschinencode zu machen, der vom Zielrechner ausgeführt werden kann [Lit7].

### 3.3.8. JavaScript

JavaScript ist eine Entwicklung der Firmen SUN und Netscape. Im Gegensatz zu den anderen hier aufgeführten Techniken, wird JavaScript vollständig Browserseitig ausgeführt. Das bedeutet, daß JavaScript nicht durch den Webserver, auf dem die Seite liegt, ausgeführt wird, sondern durch den Browser auf dem Rechner des Benutzers.
JavaScript lehnt sich in der Syntax an Java an, wird allerdings nicht vorkompiliert und ist eine Skriptsprache. Der JavaSkript-Quellcode wird dabei direkt in die HTML-Seite eingebunden.

```
<html>
<head>
    <title>JavaScript-Beispiel</title>
</head>
<body>
Hallo
<script language="JavaScript">
  document.writeln("Welt!");
</script>
</body>
</html>
```

**Quellcode 5. JavaScript-Beispiel**

Das Beispiel aus Quellcode 5 zeigt das Vorgehen. Im Beispiel wird als Ergebnis der String „Welt!“ mit Hilfe von JavaScript ausgegeben, so daß letztlich im Browser der Text „Hallo Welt!“ zu sehen ist [Lit8].
Damit JavaScript ausgeführt werden kann, muß der Browser des Benutzers dieses verstehen bzw. die JavaScript-Funktion eingeschaltet haben. Da JavaScript jedoch schon weit verbreitet ist und von über 90% aller Browser verstanden wird, hat sich diese Technik weitgehend etabliert.

### 3.3.9. ASP

ASP heißt „Active Server Pages“. Es ist eine Technik vergleichbar mit CGI und PHP, wobei es jedoch eine Eigenentwicklung von Microsoft ist. ASP läßt sich derzeit fehlerfrei nur auf den Webserver Microsoft-IIS ausführen, hat dafür aber auch Zugriff auf Funktionen, die mittels der Skriptsprache ActiveX® implementiert sind [Web30]. Aufgrund von Sicherheitsproblemen, insbesonders bei der Nutzung von ActiveX®, sowie der Beschränkung auf den Webserver, geht die Nutzung von ASP derzeit stark zurück.
Ähnlich wie PHP, kann es in Webseiten durch eine umrandende Markierung einge-

[10] Bei dem Bytecode handelt es sich um einen plattformunabhängigen Sprachkonstrukt, bei dem die Programmiersprache vorkompiliert ist. Der Zielrechner muß nur noch diesen Bytecode entsprechend des eigenen Betriebssystems fertig kompilieren.

bunden werden. Man verwendet hier die Kennzeichnung durch **<% ... %>**.

### 3.3.10. Sonstiges

Weitere Techniken, die benutzt werden können, sind DHTML („Dynamic HTML“), CSS („Cascading Style Sheets“), Shockwave, ActiveX® und VRML („Virtual Reality Modelling Language“).
Diese spielen jedoch bei der Implementierung eines WCMS eine geringe Rolle. Besonders CSS und DHTML werden zwar häufig angewendet, jedoch liegt ihre Bedeutung mehr darin, das Design einer Seite zu bestimmen und nicht aktiv einen Mechanismus zu steuern.

Ein Vergleich der meisten der oben genannten Techniken mit einer Trennung auf serverseitige und clientseitige Techniken findet sich im Artikel „Interaktive Mechanismen für Webseiten“ [Art8].

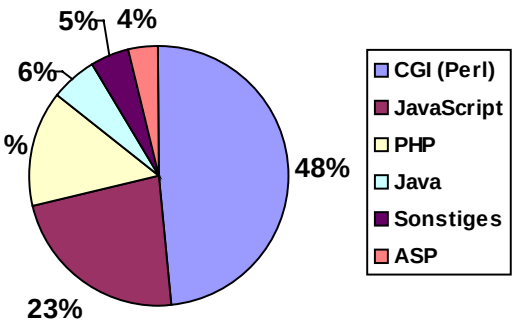
## 3.4. Akzeptanz und Nutzung der Webtechniken

Da ein nicht unerheblicher Zeitaufwand bei der Implementierung eines WCMS für die Unterstützung der oben angesprochenen Techniken benötigt wird, stellt sich die Frage, inwieweit diese überhaupt genutzt werden.

Derzeit gibt es jedoch in dieser Richtung nur wenige Forschungsberichte, bzw. bewerten diese nicht direkt die benutzten Techniken, sondern nur das Surfverhalten der Benutzer [Art7]. Abschätzungen über die Nutzung von JavaScript oder CSS auf publikumsstarken Websites sprechen zwar von einer Nutzung von über 80%, jedoch muß dies nicht für alle Seite gelten. Man hat hier das Problem, daß Websites, die von größeren Unternehmen oder Online-Dienstleistern angeboten werden, eine hohe Änderungs- und Verbesserungsrate haben, während einfache Homepages oder Seiten bei denen nur wenige Benutzer auftreten kaum mit neueren Techniken versehen werden. Auch in diesem Punkt spielen wieder Image-Überlegungen der Unternehmen eine große Rolle und wieviel Geld diese für eine akkurate Pflege der Website ausgeben.

In Form einer Online-Umfrage wurde versucht eine Abschätzung darüber zu erhalten, welche der angegebenen Techniken am häufigsten von Webdesignern und Programmierern eingesetzt wird. Die Umfrage fand vom 22.5.2000 bis zum 5.6.2000 statt und erfaßte die Meinungen von 405 Personen zu der Frage: *„Welches ist Ihre bevorzugte Programmiersprache zur Gestaltung dynamischer Webseiten?“* [Web31].

Aus den Ergebnissen nach Abbildung 12 erkennt man, daß CGI, JavaScript und PHP




**bb. 12. Webtechniken**

am häufigsten genutzt werden, während ASP und sonstige Techniken keine große Rolle mehr spielen.

# 4. Konzeption

## 4.1. Einleitung

Wie in Kapitel 2 „Komponenten eines WCMS“ aufgeführt wurde, muß ein WCMS diverse Komponenten enthalten. In Kapitel 3 “Systemumgebung und verfügbare Techniken“ wurde außerdem gezeigt innerhalb welcher Umgebung sich das System bewegen muß.

Mit Hilfe dieser Rahmenbedingungen ist es nun möglich, eine allgemeine Konzeption für WCM-Lösungen aufzustellen. Aufgabe dieses Kapitels ist es also die technischen Konzepte vorzustellen und zu beschreiben.

## 4.2. Wahl der Plattform

Je nachdem auf welcher Plattform, d.h. auf welcher Kombination von Betriebssystem und Webserver man sich bewegt, ist man in der Wahl der benutzten Programmiersprachen und Applikationen (wie z.B. verfügbare Datenbanken) eingeschränkt. Die Wahl der Plattform bestimmt somit die gesamte zukünftige Entwicklung des WCMS.

Bei der Auswahl der Plattform ergeben sich folgende Wahlmöglichkeiten:

- Webserver:
    1. Apache
    2. Microsoft-IIS
    3. Netscape Enterprise
    4. Lotus Domino[11]
- Betriebssystem
    1. Sun Solaris
    2. Linux-Derivat
    3. Windows NT/2000
- Programmiersprache(n)
    1. Interpretersprachen (Perl, PHP, VisualBasic, usw.)
    2. Compilierte oder vorcompilierte Sprachen (C, C++, Java)
- Vorhandene Datenbanken
    1. LDAP,
    2. XML[12]
    3. X500
    4. SQL
    5. ORACLE
    6. usw…

Meist ist die Wahl der Plattform und der hinzukommenden Systemkomponenten eine Kostenfrage; Sowohl von Seiten der Hardware, als auch von den nötigen Softwarelizenzen und der nötigen Manpower. Jedoch sollte man hier langfristig rechnen und die Skalierbarkeit im Auge behalten.

---

[11] Bei der Wahl des Lotos Domino-Servers sollte man jedoch nicht seine geringe Verbreitung und Probleme vergessen. Siehe hierzu Kapitel 3.2.3.

[12] Wie in Kapitel 3.3 beschrieben ist XML keine Datenbank, eignet sich jedoch für die Speicherung von Datensätzen und den standardisierten Datenaustausch. Datenbanken, die auf reine XML-Strukturen aufbauen sind deswegen eine Möglichkeit die hier in Betracht kommen sollte.

Komplexe Systeme, die Datenbankzugriffe zu bewältigen haben, erfordern in erster Linie große Rechner-Resourcen. Beispiel ist der Auftritt des Henkel-Konzerns bei dem das gesamte Intranet auf mehrere NT- und Domino-Server mit 2 bis 4 Prozessoren verteilt ist und für den Arbeitsspeicher Kapazitäten zwischen 500 MByte und einem GByte benötigt werden [Art9].
Im Vergleich dazu kommen die Seiten der „Hamburg Messe und Congress“ (HMC), die statisch gehalten werden, mit einer Unix Workstation von Sun mit 2 Prozessoren aus.

## 4.3. XML-Integration

Da die Nutzung von XML sowohl als Schnittstelle, als auch als Speicherformat auch über absehbare Zeit hinaus einer der wichtigsten Punkte bei der Erstellung von WCM-Lösungen und Applikationen bleiben wird, haben solche Systeme Vorteile, die das XML-Konzept schon von Anfang an nutzen.
Zwar ist es bei vielen Applikationen ausreichend, die XML-Integration lediglich in Form einer Import-/Export-Schnittstelle anzubieten, jedoch wird der Unterschied spätestens dann deutlich, wenn das WCMS weiter ausgebaut werden soll und/oder interne Bestandteile durch fremde Module ersetzt werden sollen. Die Ablage mittels XML ist zukunftssicher, da nicht abhängig von der gerade benutzten technischen Umsetzung des WCMS: Ein Wechsel der Software oder Austausch von Informationen mit anderen Applikationen kann unkompliziert erfolgen.
Der Datenaustausch zwischen einzelnen Bestandteilen eines WCMS sollte deshalb auf XML aufgebaut sein.

Des weiteren bietet, wie oben bereits erläutert, XML für die Speicherung der Inhalte Vorteile, da es diese von vornherein strukturiert mit Hilfe eines offenen Standards ablegt.

## 4.4. Wahl und Konzeption des Benutzer-Interfaces

Das Benutzer-Interface bezeichnet die Schnittstelle von Autoren, Redakteuren und Administratoren zum WCMS.
Das Benutzer-Interface kann in Form einer grafischen Oberfläche realisiert sein, oder in rein textueller Form.
Es werden zwei verschiedene Verfahren angeboten: Clientseitige und serverseitige Interfaces.

### 4.4.1. Clientseitige Interfaces

Bei einem clientseitigen Interface wird die Bedienoberfläche für das WCMS entweder als Java-Applet aus dem Webbrowser des Benutzers gestartet oder aber durch ein eigenes Programm, einem Client, auf dem Rechner des Benutzers ausgeführt.

Der Benutzer hat dabei vollen Zugriff auf die gesamte Funktionalität des WCMS und dessen Daten. Die interne Kommunikation zwischen Client und Server verläuft für den Benutzer transparent.

Der Vorteil bei der Programmierung von Clients liegt darin, daß diese nicht innerhalb eines Browser ablaufen müssen und somit auch nicht den Beschränkungen von HTML unterordnet sind. Somit ist z.B. es möglich über einen Client die Seiten im einem WYSIWYG-Modus zu bearbeiten.

Der Nachteil von Clients liegt darin, daß diese plattformabhängig sind.

#### 4.4.2. Serverseitige Interfaces

Serverseitige Interfaces benutzen in den Regel einen Webbrowser als Medium. Die Übermittlung von Nachrichten und Befehlen erfolgt über das CGI, also mit Hilfe von Formularen die dynamisch von einem Programm der auf dem Webserver abgerufen wird, erstellt werden.

Da bei serverseitigen Interfaces das Programm durch den Webserver ausgeführt wird, sind diese Interfaces plattformunabhängig.

Serverseitige Interfaces die mit Hilfe einer Kombination von ASP und ActiceX® arbeiten verlangen jedoch meist als Webbrowser den Microsoft Internet Explorer™.

Da jedoch nicht alle Browser auf allen Betriebssystemen vorkommen, ist von der Verwendung von Webtechniken abzuraten, welche einen speziellen Browser benötigen. Dies trifft insbesonders für ActiveX® zu, da der Internet Explorer™ nicht oder nur mangelhaft auf Unix- oder Linux-Systemen vorkommt.

## 4.5. Konzepte zur Daten- und Seitenhaltung

Bei der Erstellung eines WCMS ist einer der wichtigsten Punkte die Wahl der passenden Speichermethode. Derzeit unterscheidet man im wesentlichen zwei Methoden, sowie ihre Mischformen: **Publishing-/Staging Server** und **direktes dynamisches Publishing**.

#### 4.5.1. Publishing-/Staging-Server

Beim Konzept des Publishing-/Staging-Servers benutzt man zwei verschiedene Bereiche zur Ablage und Bearbeitung der Seiten. Der Publishing-Server enthält das eigentliche WCMS und bearbeitet die Requests der Autoren, Redakteure und Administratoren. Dieser Server befindet sich meist im Intranet und ist für die Öffentlichkeit nicht erreichbar. Alle Seiten die von Autoren erstellt und bearbeitet wurden, werden auf diesem Server gespeichert.

Sollen die Inhalte veröffentlicht werden, wird durch das WCMS ein Prozeß gestartet, welches aus allen vorhandenen Seiten

HTML-Dateien erstellt. Diese HTML-Dateien werden dann auf den Staging-Server gelegt und so für die Öffentlichkeit abrufbar gemacht.

Abbildung 13 verdeutlicht dieses Konzept: Autoren, Redakteure und Administratoren greifen zur Bearbeitung der Seiten auf den Publishing-Server zu. Dies geschieht in der Abbildung je nach Wahl durch ein server- oder clientseitiges Interface. Gegebenenfalls muß ein Webserver die Requests zwischen Autoren und Publishing-Server entgegennehmen und weiterleiten.

Auf dem Publishing-Server werden alle Inhalte verwaltet und bearbeitet und bei gewünschter Veröffentlichung die HTML-Dateien erstellt und auf den Staging-Server geschoben.

Die HTML-Dateien auf dem Staging-Server sind dabei statisch (sieht man von eingebundenen Webtechniken ab).

Die Bearbeitung von Seiten kann nur auf dem Publishing-Server erfolgen. Wird eine HTML-Datei auf dem Staging-Server geändert, wird diese bei der nächsten Veröffentlichung durch den Publishing-Server überschrieben. Zwar bringt dies ein organisatorisches Problem mit sich, jedoch bietet dieses Vorgehen den Vorteil der Redundanz: Fällt der Staging-Server aus, kann trotzdem noch auf dem Publishing-Server gearbeitet werden, bzw. fällt der Publishing-Server aus, können immer noch die Webseiten vom Staging-Server gelesen werden.

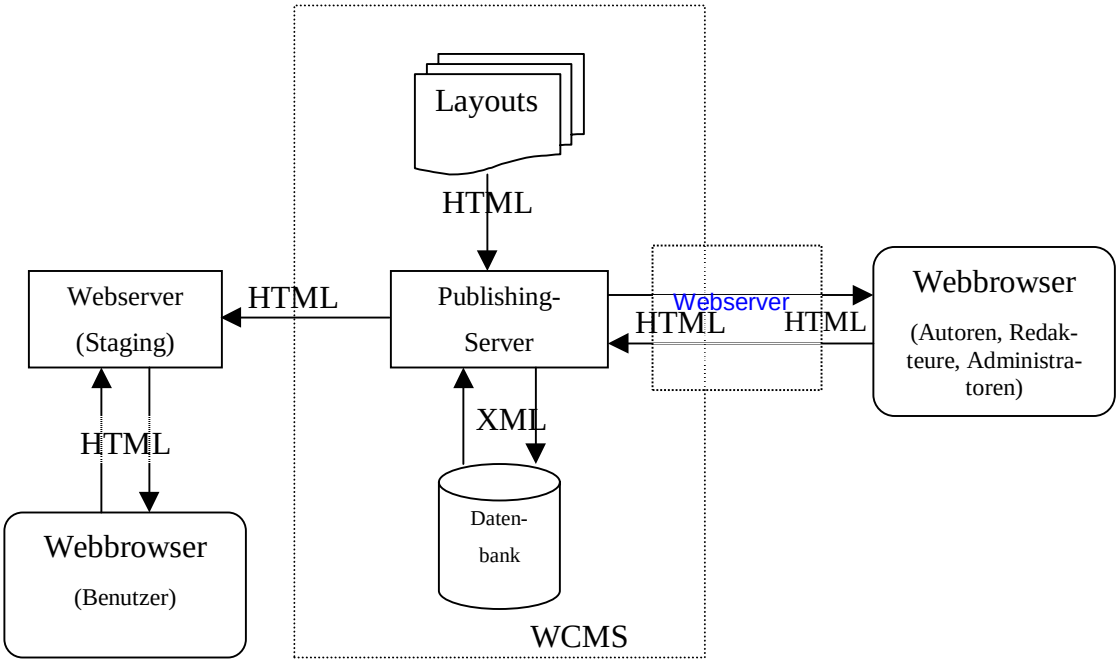



**Abb. 13. Publishing-/Staging-Server**

Die Vor- und Nachteile für die Verwendung von statischen Seiten über den Staging-Server lassen sich wie folgt zusammenfassen:

+ Gute Performance

+ Unkompliziertes Backup
+ Statische Seiten werden von Suchmaschinen erfasst
- Nicht geeignet für häufige Aktualisierungen
-/+ Seitenrecherche für Benutzer durch eigene, aber ggf. spezialisierte, Suchmaschine nötig.
- Layoutänderungen für mehrere Dateien aufwendig

Eingesetzt wird dieses Konzept z.B. in den VIP Content Manager von Gauss Interprise oder von SchemaText.

Die HMC, welche bereits in Kapitel Serverseitige Interfaces erwähnt wurde, benutzt den VIP Content Manager [Art10].

### 4.5.2. Dynamisches Publishing

Beim dynamischen Publishing werden angeforderte Seiten direkt aus der Datenbank publiziert. Abbildung 14 schematisiert dieses Konzept: Beim Aufruf einer Webseite über den Webbrowser des Benutzers gibt der Webserver den Request weiter an das WCMS, welches die Seiteninhalte aus einer Datenbank anfordert, diese mit dem Layout verknüpft und dann die HTML-Ausgabe an den Webserver und damit den Benutzer zurückgibt.

Die verwendete Datenbank des WCMS' kann in diesem Verfahren auch die eines anderen Herstellers, wie Oracle oder Informix sein.

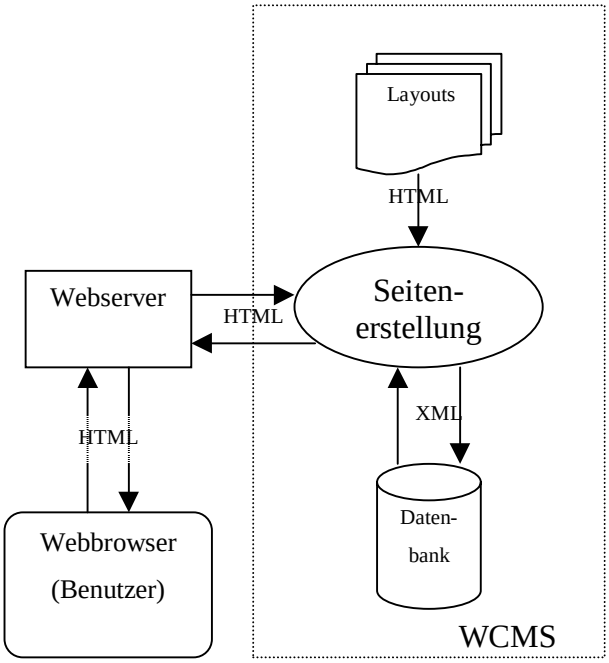



**Abb. 14. Dynamisches Publishing**

Da die eigentlichen HTML-Seiten erst bei der Anfrage durch einen Benutzer erstellt werden, spricht man hier von dynamischen Seiten. Man erkennt diese leicht daran, daß die URL zu diesen Seiten nicht fest ist, sondern meist eine kryptische Gestalt besitzt und Parameter betreffend des Aufrufes enthält. Ein Verzeichnis mit statischen HTML-Dateien existiert nicht mehr.

Die Vor- und Nachteile des dynamischen Publishings sind:

+ Leichte Aktualisierungen
+ Einfache Verwaltung der Seiten
+ Flexible Nutzung der Datenbanken von Drittherstellern
+ Seitenrecherche eingebaut
- Für ausreichend Performance sind hohe Rechner-Ressourcen nötig
- Nachteile bei der Erfassung durch

Suchmaschinen: Viele Suchmaschinen erfassen keine dynamischen Seiten, so daß diese bei einer Suche nicht gefunden werden.

Der in Kapitel 4.2 „Wahl der Plattform“ erwähnte Henkel-Konzern setzt ein WCMS ein, welches dynamisches Publishing benutzt.

#### 4.5.3. Mischformen

Eine Erweiterung des dynamisches Publishing stellt das **Caching** dar. Hier wird dem Fakt, das einige Seiten stärker frequentiert werden als andere, Rechnung getragen. Um lange Ladezeiten für die jeweilige Neuerstellung von Seiten aus der Datenbank zu vermeiden, werden solche Seiten, die häufig angefordert werden entweder statisch an festen Positionen abgelegt, oder aber in ihrer fertigen Form als HTML-Seite in einen Cache-Bereich gespeichert. Der Cache ist dabei entweder ein Speicherbereich auf dem Webserver oder aber ein temporärer Bereich auf der Festplatte des Servers.

Die Frage, welche Seiten gecached werden, kann automatisch über die Anzahl der Zugriffe, als auch über die Administration durch Redakteure beantwortet werden.

Um Aktualität zu garantieren muß der Cache automatisch aktualisiert werden, wenn im WCMS sich Inhalte der betreffenden Seiten geändert haben.

Aufgrund seiner Struktur ist das Caching eine Mischung der Konzepte des dynamischen Publishing und des Publishing-/Staging-Servers.

#### 4.5.4. PSE-Konzept

Ein weiteres Konzept, welches mit dieser Arbeit erstmals vorgestellt wird, ist das des ***Publishing-, Staging- und Extract-Konzept***, kurz PSE-Konzept.

Das Konzept baut auf das des Publishing-/Staging-Servers auf und erstellt auf dem Webserver statische Webseiten. Neu ist jedoch die Art der Speicherung und die Verarbeitung und Verwaltung neuer Seiten.

Abbildung 15 schematisiert das Konzept. In dem Konzept sind die HTML- und Objektdateien, welche von WCMS bearbeitet wurden, für den Webserver voll les- **und schreibbar**. Dies bedeutet nichts anders, als daß das WCMS die vollständige Kontrolle über alle Inhalte zurückgibt und es erlaubt, das andere Applikationen darauf Zugriff nehmen können. Einzige Prämisse dabei ist, daß die Dateien statisch auf dem Webserver liegen.

Für den Benutzer, der eine Seite abruft, verhält es sich daher wie ein normales statisches System.

Ebenso verhält es sich, wenn ein Redakteur eine neue Seite publiziert: Aus den Inhalten und den Layouts wird unmittelbar eine HTML-Datei erstellt und auf dem Webserver gespeichert. Die Inhalte der Datei wer-

den, sofern es sich nicht um eingebundene Mediendateien handelt (Bilder, Grafiken,..), ebenfalls nur in dieser HTML-Datei gespeichert. Einzige Ausnahme: Meta-Angaben wie „Titel“, „Description“, „Keywords“, „Autor“, „Version“, usw. werden in eine Meta-Datenbank geschrieben.

Soll nun eine HTML-Datei verändert werden, wird das WCMS sie vom Webserver laden. Danach wird das Layout, mit dessen Hilfe die Datei erstellt wurde, genutzt um alle Layout-Angaben in der HTML-Datei zu löschen.

Nach diesem Schritt bleiben nur noch die Inhalte übrig und können somit direkt geändert werden.

In anderen Worten: Zur Wiedererlangung der Inhalte wird die Inverse Operation zur Seitenerstellung (siehe Kapitel 2.3.1 „Seitenerstellung“) angewendet.

Die Meta-Datenbank ist nötig um Daten, die nicht in der HTML-Datei gespeichert werden, zu sichern. Darunter auch Angaben zum verwendeten Layout, Kontroll- und Zugriffsdaten.

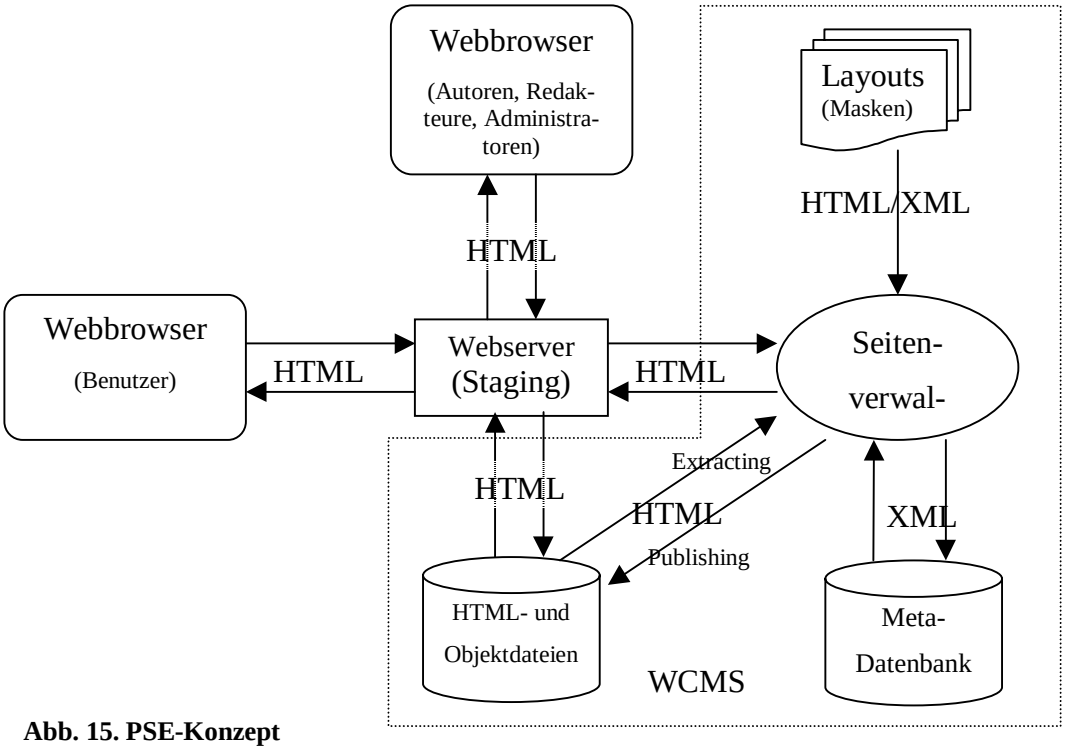



**Abb. 15. PSE-Konzept**

Die in der Abbildung zusätzlich angegebenen Layouts können ebenfalls als HTML-Dateien abgespeichert und somit verwaltet und geändert werden. Basiert das Konzept auf serverseitige Interfaces ist es möglich eine Selbstadministration zu erreichen, indem alle Layout- und Kontrollausgaben zur Verwaltung durch Redakteure und Autoren ebenfalls Teil der änderbaren Seiten sind.

Vor- und Nachteile dieses Konzeptes:

+ Gute Performance
+ Unkomplizierte Backups

- \+ Einfache Aktualisierungen
- \+ Erlaubt die gleichzeitige Verwendung anderer WCM-Lösungen, die nach dem Pushing-/Staging-Konzept arbeiten
- \+ Erlaubt Autoren mittels eigener Editoren direkt in HTML-Dateien zu arbeiten
- \+ Import bestehender Seiten einfach
- \- Benötigt Abbildung des File-Systems innerhalb der Meta-Datenbank
- \- Wenig Praxiserfahrungen, da neu
- -/+ Seitenrecherche für Benutzer durch eigene, aber ggf. spezialisierte, Suchmaschine nötig.
- \- Layoutänderungen für mehrere Dateien aufwendig

# 5. Implementation

Bevor die eigentliche Implementation beschrieben wird, werden die Punkte angegeben, die als Basis in einem solchen System auftauchen müssen, Aussagen über die erforderlichen Leistungen treffen und somit das Lasten- und Pflichtenheft beschreiben. Da es jedoch nicht die Aufgabe dieser Arbeit ist, die genauen Methoden des Software-Engineerings bei der Erstellung von größeren Projekten zu erklären, beschränkt sich dies auf die Angabe der Grundlagen. Techniken, wie zum Beispiel die Anwendungen von Data Dictionary, Jackson Diagramme, das Entity-Relationship-Modell und Automaten werden nicht besprochen. Näheres hierzu findet sich im Werk von Helmut Balzert über Software-Management [Lit9] und darin insbesonders auf die Kapitel 2 bis 4.

Da zudem die programmiersprachliche Umsetzung sowie dazugehörige Struktugramme stark von der gewählten Programmiersprache und der Plattform abhängen, wird nur die Definitionsphase angegeben und betreff der Umsetzung nur ein gekürzter Überblick über die für diese Arbeit programmierten Mechanismen.

Die Testphase wird durch das Kapitel 6 „Einsatz und Test eines WCMS“ abgedeckt.

## 5.1. Definitionsphase

### 5.1.1. Lastenheft

Das Lastenheft besteht aus den folgenden Punkten:

**Zielbestimmung**. Das Programmsystem soll es ermöglichen, eine Internetpräsenz umfassend zu verwaltend und zu administrieren.

**Produkteinsatz**. Das System dient im wesentlichen dazu, unterschiedlichen Mitarbeitern eine Möglichkeit zu geben, Dokumente zu publizieren, ohne daß diese dabei durch ein zu strenges Regelwerk oder zu starre Schnittstellen die Produktivität mindern. Das System wird auf einem Unix- bzw. Linux-Server eingesetzt auf dem als Webserver Apache installiert ist.

**Produktfunktionen**. Im Wesentlichen müssen folgende Minimalfunktionen vorhanden sein:

1. Unterstützung von Methoden des Content-Managements zur Trennung von Inhalt und Design.
2. Workflow-Management zur Steuerung von redaktionellen Verfahren.
3. Kontrolle des Content-Lifecycles einzelner Dokumente.
4. Referenzkontrolle (Linküberprüfung).

**Produktdaten/Produktklassen**. Es ist aufgabenbedingt eine Datenbank sowohl über Dokumentinformationen, als auch über Benutzer anzulegen. Des weiteren müssen Schnittstellen derart gestaltet sein, daß das System skalierbar und plattformunabhängig ist.

**Produktleistungen**. Das System muß sowohl mit großen Benutzerzahlen, als auch mit einer großen Zahl an zu verwaltenden Dokumenten zurechtkommen. Im Fallsystem RRZE müssen über 50 Mitarbeiter mit zusammen über 9500 Webseiten verwaltet werden.

**Qualitätsanforderungen.**

| ***Systemqualität*** | Sehr gut | Gut | Normal | Nicht relevant |
|---|---|---|---|---|
| Funktionalität | | X | | |
| Zuverlässigkeit | | X | | |
| Benutzbarkeit | X | | | |
| Effizienz | | | X | |
| Änderbarkeit | | | X | |
| Übertragbarkeit | | X | | |

**Ergänzungen**. Mögliche Ergänzungen, wie zum Beispiel den Einbau eines Helpdesk-Systems, sollen durch modular aufgebaute Programmteile, flexible Schnittstellen und Datenbankfelder ermöglicht werden. So ist es von Vorteil wenn alle Daten so gespeichert werden, daß sie jederzeit zusätzliche Attribute erhalten können, ohne daß dabei vorhandene Funktionen gestört werden bzw. die neuen Daten aufgrund alter Funktionen nicht berücksichtigt werden.

Aus diesen Punkten muß mit den Methoden des Software-Managements und -Engineering, wie sie in [Lit9] beschrieben werden, ein Pflichtenheft erstellt werden.
Das Pflichtenheft enthält die Zusammenfassung aller fachlichen Anforderungen, die das zu entwickelnde System erfüllen muß [Lit10]. In anderen Worten, es beschreibt den gesamten Funktions-, Daten-, Leistungs- und Qualitätsumfang.

### 5.1.2. Pflichtenheft

#### 5.1.2.1. Zielbestimmung

a) Mußkriterien:
   - XML-Schnittstellen.
   - Redaktionelle Workflows.
   - Funktionen des CM: Trennung von Inhalt und Design.
   - Benutzer- und Zugriffsverwaltung.

b) Wunschkriterien
   - Schnittstellen zu gängigen Textverarbeitungsprogrammen, wie WORD, StarOffice, ...

- Schnittstellen zu anderen Protokollen. So zum Beispiel zu WAP, SMTP, ...
- Schnittstellen zu vorhandenen Kalender- und Assistenten-Verwaltungen, wie z.B. Groupwise.
- Assistenten-Funktionen zur Verbesserung der Benutzerfreundlichkeit: Automatische Korrekturhilfen, Zeitplaner, ...
- Embedded HelpDesk[13]-System.
- Kontroll- und Administrationsmöglichkeiten für den Webserver.

c) Abgrenzungskriterien

- Keine Unterstützung für Webbrowser mit einer Marktverbreitung unter 5 Prozent.
- Keine Festlegung des Webservers auf ein Produkt unter einer Verbreitung von 5 Prozent. Damit keine Verwendung von Lotus Domino.

#### 5.1.2.2. Einsatzgebiet und Anforderungen an das System

a) Anwendungsbereiche

- Für Eingabe und Bearbeitung von Inhalten und Layouts: HTML- / Text-Editoren sollen entweder unterstützt werden oder über einen Client implementiert sein.
- Administrationsfunktionen sollen durch einen Webbrowser steuerbar sein.
- Unterstützung zur Link-Verwaltung für Webseiten und Objekte.

b) Zielgruppen

- Autoren ohne tiefere Kenntnisse in HTML.
- Redakteure mit Überblick über die laufenden Projekte ihrer Autoren; Redakteure benötigen ebenfalls keine tiefere Einblicke in HTML.
- Designer mit Kenntnissen in HTML und Webtechniken.
- Administratoren mit guten Kenntnissen in HTML und Webtechniken. Administratoren müssen ebenfalls einen Überblick über die Struktur des Webservers und der enthaltenen Seiten haben.

c) Betriebsbedingungen

- Das System muß mit den gleichzeitigen Zugriff mehrerer Benutzer zurecht kommen.

[13] Derzeit sind HelpDesk-Systeme in Form von Telefon- und Mail-Hotline-Assistenten auf dem Markt erhältlich, die mit den Methoden des Knowledge-Management auf große Wissensdatenbanken zugreifen. Da auch bei diesen die Methoden des Content-Managements benutzt werden und somit eine gemeinsame Basis vorhanden ist, kann die Implementierung eines HelpDesk-Systems Aufgabe zukünftiger Generationen von WCM-Lösungen sein.

- Verwaltung von einer nur durch den Speicher des Betriebssystem begrenzten Zahl an Objekten.
- Abbildung des Unix-File-Systems innerhalb der Seitenstruktur.

#### 5.1.2.3. Architekturentwurf

a) Software
- Server: Unix oder Linux-Derivat. Perl ab Version 5.04, installierte Basismodule für Perl, sowie Module LWP, Data, CGI, XML. HTML. Apache-Webserver. Der Webserver muß CGI, SSI und Zugriffsschutzfunktionen mittels htaccess aktiviert haben.
- Benutzer: Beliebiger PC oder eine Workstation mit einem Webbrowser ab der 4er Generation. Der Webbrowser muß JavaScript ausführen können.

b) Hardware
- Server: Sun Workstation oder PC ab 400 MHz, Mind. 512 MByte RAM.
- Benutzer: Monitor mit einer Auflösung von mindestens 800x600 Punkten.

#### 5.1.2.4. Beschreibung des Funktionsumfangs

a) Benutzerverwaltung
- Einrichten, Ändern und Löschen von Benutzer-Accounts.
- Abruf des Benutzer-Status.
- Schnittstelle zum Aufgaben-Management.
- Kontakt-Funktionen für Benutzer. Benutzer können direkt per E-Mail oder Aufgaben-Verwaltung Nachrichten erhalten.

b) Workflow-Management
- Schnittstellen sowohl zum Aufgaben- und Dokumenten-Management, wie der Benutzerverwaltung.
- Abbildung des Kontroll- und Freigabezyklus.
- Unterstützung eines „privaten“ Kontroll- und Freigabezyklus für Drafts, die vom Autoren selbst rezensiert werden, bevor sie von einem Redakteur geprüft werden.

c) Rechteverwaltung
- Vergabe und Bearbeitung von Zugriffsrechten.
- Schnittstelle zur Benutzerverwaltung.
- Rechtetabelle zur Festlegung von Zugriffsrechten auf einzelne Funktionen des Programms,

sowie zum Zugriff auf Objekten und Dokumententypen.

- Vererbares Rechtekonzept für Benutzer, die weitere Benutzer anlegen können: Ein Benutzer kann die Rechte an andere nur weitergeben, wenn er sie selbst hat.
- Bei Rechten wird unterschieden zwischen Zugriffsrechten auf Programmfunktionen und Zugriffsrechten auf Objekte.
- Ein Benutzer A, der einen Benutzer B erstellt hat, hat immer Zugriff auf die von dem Benutzer B erstellten Seiten, Daten und weitere Benutzer.

d) Aufgaben-Management

- Alarmierung von Benutzern für die Kontrolle von Seiten und Objekten und Verwaltung bestehender ToDo-Listen.
- Schnittstelle zur Benutzerverwaltung.
- Ausgabe von ToDo-Informationen einzelner Benutzer.
- Teilweises automatisches Löschen und Bearbeiten von Aufgaben.

e) Dokumenten-Management

- Kontrolle, Zusammenführung und Trennung von Design und Inhalt.
- Verwaltung der Dokumenten-Layouts.
- Verwaltung von Dokumenten während ihrer verschiedenen Lebensphasen.
- Verwaltung von Meta-Daten.
- Schnittstellen zu Aufgaben- und Workflow-Management.

f) Linküberprüfung

- Verwaltung einer Referenztabelle für alle Objekte.
- Eintrag, Bearbeitung und Austrag von Referenzen.
- Kontrollfunktionen für die Existenz oder Nichtexistenz von Links.
- Schnittstellen zu Dokumenten- und Workflow-Management

#### 5.1.2.5. System-Daten

a) Benutzerdaten. Attribute für Benutzer sind:

- ID.
- Name, Vorname, FreeformName.
- ID-Liste eigener Verzeichnisse.
- URL eigener Homepage.
- ID des Erstellers.
- Dateiangabe zur persönlichen Aufgaben-Verwaltungsdatei.
- E-Mail.
- Abteilung.
- ID-Nummern für Rechte auf Programmfunktionen.

- Erstellungsdatum.
- Datum des letzten Logins.
- Datum der letzten Aktion.

b) Verzeichnisdaten

- ID.
- Datum und Zeit der letzten Änderung.
- Datum und Zeit der Erstellung.
- Name.
- ID des Oberverzeichnisses.
- ID-Liste der Unterverzeichnisses.
- Version.
- Besitzer.
- Pfad.
- Liste aller Links des File-Systems.
- Lebenszeit für Dateien in diesem Verzeichnis[14].
- ID des Autors der letzten Änderung.
- ID des Autors, der dieses Verzeichnis anlegte.
- URL zu diesem Verzeichnis.
- ID-Liste der enthaltenen Dateien.
- ID-Liste der für dieses Verzeichnis erlaubten Dokumententypen.
- Kurzbeschreibung des Verzeichnisses.
- ID-Liste der Benutzer, die Schreibrechte auf dieses Verzeichnis haben.

c) Seitendaten

- ID.
- ID für Dokumententyp.
- Titel.
- Statusflag.
- Meta-Datenfelder.
- ID zum Ziel-Verzeichnis.
- ID zum Rezensions-Verzeichnis[15].
- Anzahl der nötigen freigebenden Redakteure.
- ID für den Benutzer der die letzte Änderung ausführte.
- ID für den Ersteller.
- Aktueller Dateiname.
- Dateiname nach Freigabe.
- ID für den aktuellen Besitzer.
- ID-Liste für die Redakteure.
- Versionsnummer.
- Datum der letzten Änderung.
- Datum der Erstellung.
- Lebenszeit.
- Status der Kontoll- und Freigabe.

[14] Dieser Wert wird durch den des Dokumententypes überschrieben.

[15] Dieses Verzeichnis wird benötigt, wenn die Seite noch nicht freigegeben ist, sondern noch im Kontroll- und Freigabezyklus ist.

d) Dokumententyp-Daten
- Name.
- ID.
- Zugriffstyp.
- Anzahl der nötigen Redakteure zur Freigabe.
- ID-Liste der Redakteure.
- ID des Erstellers.
- Angabe der verwendeten Objekte (Referenzen auf andere Dokumententypen).
- ID zu einem Zielverzeichnis[16].
- Content-Type.
- Lebenszeit für Seiten dieses Typs.
- Angabe der in der Seite zu verwendenden Meta-Daten.
- Flag zur Verwendung des Benutzer- und Freigabezyklus'.
- Dateiangabe zur verwendeten Layoutdatei.
- Kurzbeschreibung.

e) Daten des Aufgaben-Managements
- ID.
- Zeitangabe der letzten Benachrichtigung.
- Kurzbeschreibung der Aufgabe.
- Typ der Aufgabe (Erstellt durch einen anderen Benutzer oder durch das System).
- Aufgabentext.
- Priorität.
- ID des Aufgabenerstellers.
- Lesestatus dieser Aufgabe (wurde sie schon vom Empfänger gelesen?).
- Erstellungsdatum der Aufgabe.

#### 5.1.2.6. Bedienoberfläche

a) Bildschirmlayout: Design und Menüstruktur
- Fenster zum Einloggen in das System.
- Nach dem Einloggen Aufbau eines eigenen Fensters mit einer auf JavaScript basierenden dynamischen Menüstruktur, ähnlich die des MS Explorers™.
- Innerhalb des Bedienfensters Zugriff auf alle Funktionen des Programms mit Hilfe von mit SSI interpretierte Webseiten, welche über CGI Programmfunktionen aufrufen.
- Dynamisch generierte Formularseiten für alle Funktionen. Die Formularseiten basieren auf Layoutdateien, die ebenfalls über das WCMS verwaltet werden können.

[16] Ein Dokumententyp verwendet ein Zielverzeichnis, wenn festgelegt ist, das Seiten dieses Typen in nur einem bestimmten Verzeichnis gespeichert werden sollen.

- Vor dem Aufruf jeder Funktion über das Bedienfenster interne Kontrolle der Benutzerechte.

b) Verwendete Browser
- Netscape Navigator ab Version 4.0.
- Microsoft Internet Explorer™ ab Version 4.0.

## 5.2. Umsetzung

Bei der programmiertechnischen Umsetzung hatten besonders drei Prämissen entscheidenden Einfluß auf die Gestalt des Quellcodes:

- Das System soll auf einem solchem Server laufen können, wie er von den meisten Betreibern von Webservern genutzt wird: Auf einer Plattform mit Unix/Linux und dem Apache Webserver. Speziallösungen oder Programmteile, welche einen bestimmten Browser oder ein spezielles Betriebssystem verlangen, waren strikt zu vermeiden.
- Alle Programmteile sollen modular aufgebaut sein, so daß sie leicht austauschbar oder für die Kommunikation durch Drittprogramme erweiterbar sind.
- Als Speicherformat ist XML zu verwenden.

Als Programmiersprache wurde Perl gewählt. Die Gründe hierzu war im wesentlichen dessen einfacher Aufbau, Vorteile in Bezug auf die Nutzung von regulären Ausdrücken, welche bei der Analyse von Texten notwendig sind, und die plattformübergreifende Verfügbarkeit der Sprache.

Der größte Teil der Funktionen wurde auf Perl-Module verteilt. Diese sind in die einzelnen Systemprogramme dynamisch einbindbar. Der Quellcode kann im Extrateil eingesehen werden.
Als Literatur zu Perl bieten sich die Bände vom O'Reilly-Verlag an, sowie die unter [Art11] aufgeführten Werke.
In den folgenden beiden Kapitel wird ein technischer Überblick über die Module und das Hauptprogramm in ihren Funktionsweisen gegeben. Die Funktionen entsprechen im wesentlichen denen, die im Pflichtenheft, Kapitel 5.1.2.4 „Beschreibung des Funktionsumfangs" aufgeführt wurden.

### 5.2.1. Module

#### 5.2.1.1. XML_DBI

Das XML-Modul ist das zentrale Modul für den gesamten Austausch über XML. Es liest XML-Strukturen ein und konvertiert diese zu Perl-Hashes. Ebenso kann es Perl-Hashes entgegennehmen und als XML-Strukturen speichern.

Bei dem Einlesen von XML-Strukturen wird von „well-formed“ Strukturen ausgegangen. Dies bedeutet, es wird angenommen, daß die in den XML-Daten gemachten Strukturen konform zu der zugehörigen DTD sind. Eine Überprüfung zwischen DTD und XML-Daten findet nicht statt.
Wohl aber werden XML-Konstrukte ignoriert, welche entgegen der Norm sind.
Folgende Funktionen werden exportiert[17]:

- *XML_to_Hash()*
- *XMLfile_to_Hash()*
- *Hash_to_XML()*
- *LineUp_Attributes()*

*XML_to_Hash()* konvertiert ein XML-Konstrukt zu einem Perl-Hash. *Hash_to_XML()* ist die dazu inverse Funktion. Bei *XMLfile_to_Hash()* wird eine XML-Datei gelesen und kein XML-String übergeben.
Die Funktion *Hash_to_XMLfile()* wurde nicht implementiert, da bei dieser Richtungsangabe eine DTD angegeben werden müßte, was aber bei einer automatischen Verarbeitung und Nichtkennen der Gesamtheit aller Strukturen nicht immer möglich ist.
Alle folgenden WCMS-Module greifen auf das XML-Modul zu.

[17] Wenn Funktionen exportiert werden bedeutet es, daß sie anderen Programmen und Modulen zur Verfügung gestellt werden. Neben den exportierten Funktionen kann es beliebig viele weitere Funktionen geben, welche jedoch transparent sind.

#### 5.2.1.2. USER_DBI

USER_DBI enthält alle Funktionen, welche in irgendeiner Form etwas mit den Benutzern zu tun haben.
Exportierte Funktionen sind:

- *SendMailtoUser()*
- *AddNewFollower()*
- *UpdateUserTimes()*
- *newUser()*
- *ReadUser()*
- *ReloadUser()*
- *AddUser()*
- *PopUser()*
- *UpdateUser()*
- *ReadUserIndex()*
- *ReadUserData()*
- *SearchUser()*
- *AllowUserEdit()*
- *GetFollowers()*
- *SubstractFollower()*

Da die Daten aller Benutzer in einer XML-Datei gespeichert sind, wird auf die einzelnen Funktionen mit Hilfe von Methoden der OOP zugegriffen.
So wird mit der Funktion *newUser()* eine Referenz auf ein Benutzer-Objekt generiert. Bis auf Sonderfunktionen, wie *ReadUserIndex()* und *SearchUser()* sind alle Funktionen Methoden auf das Objekt.

#### 5.2.1.3. GROUP_DBI

Dieses Modul enthält die Verwaltung für Seiten und Objekte und managed die Ab-

bildung des Unix File-Systems auf das WCMS. Neben dem XML-Modul handelt es sich hierbei um das wichtigste Module des Systems

Folgende Funktionen werden exportiert:

- *SearchFileID()*
- *globalfileindex()*
- *fileindex()*
- *GetFileID()*
- *addlog()*
- *ParseIndexonOwner()*
- *AddFileTypes()*
- *updatefile()*
- *getfiledata()*
- *newfile()*
- *addfile()*
- *addlink()*
- *removedir()*
- *savedir()*
- *createdir()*
- *haschilds()*
- *hasfather()*
- *newdir()*
- *loaddir()*
- *newindex()*
- *firstdir()*
- *loadindex()*
- *firstdir()*
- *lastdir()*
- *prevdir()*
- *nextdir()*
- *fatherdir()*
- *GlobaleReplaceUserFiles()*
- *ReplaceDirOwner()*
- *ReplaceFileOwner()*
- *FindFilesfromType()*
- *ParseIndexonUser()*
- *updatedir()*
- *loadfile()*
- *CalcFileAge()*

Wie auch bei dem USER_DBI-Modul wird auf die Programmierung mittels OOP zurückgegriffen. Es werden hier jedoch 3 verschiedene Objektreferenzen auf Verzeichnisse und Dateien, sowie Indizes angeboten.

#### 5.2.1.4. TODO_DBI

Das Modul TODO_DBI enthält das im Pflichtenheft definierte Aufgaben-Management. Dabei wird bei den Funktionen zur Alarmierung ebenfalls auf das USER_DBI-Modul zugegriffen.

Die exportierten Funktionen sind:

- *PopTodoByFileID()*
- *AddUserTodo()*
- *CountTodo()*
- *RaisePriority()*
- *LowerPriority()*
- *GetHTMLonPrioName()*
- *CreateTodoFile()*
- *RemoveTodoFile()*
- *RemoveBackups()*
- *new()*
- *ReadTodo()*
- *ReloadTodo()*

- *AddTodo()*
- *PopTodo()*
- *UpdateTodo()*
- *ReadTodoIndex()*
- *ReadTodoData()*
- *SearchTodo()*
- *MarkTodoRead()*
- *GetIndexonPriority()*
- *HighestPriority()*
- *ReadPriorityDefinitions()*

Ebenso wie bei USER_DBI und GROUP_DBI handelt es sich bei den meisten Funktionen um Methoden zur Objektverwaltung. Mit der Funktion *new()* wird eine Objektreferenz für eine Aufgabe erstellt.

#### 5.2.1.5. PAGETYPE_DBI

Dieses Modul verwaltet die Dokumententypen. Da die Dokumententypen für die Seitenerstellung, sowie dessen inverse Operation bei der Implementierung des PSE-Konzeptes zuständig sind, muss dieses Modul ebenfalls auf die Module GROUP_DBI, TODO_DBI und USER_DBI zugreifen.

Es werden folgende Funktionen exportiert:

- *newPageType()*
- *ReadPageType()*
- *PopPageType()*
- *AddPageType()*
- *ReadPTIndex()*
- *ReadPageTypesData()*
- *UpdatePageType()*

Mit Ausnahme von *ReadPTIndex()* und *newPageType()* handelt es sich bei den Funktionen um Methoden auf ein Dokumententyp-Objekt, welches durch die Funktion *newPageType()* generiert wird.

*ReadPTIndex()* gibt ein Index aller vorhandenen Dokumententypen zurück.

#### 5.2.1.6. LINK_DBI

Das Modul LINK_DBI enthält die Verwaltung der Referenzen auf und zu Objekten und Seiten. Über dieses Modul wird die Frage beantwortet, ob eine Seitenreferenz erlaubt und korrekt ist, bzw. wie der Referenzbaum eines Objektes oder einer Seite aussieht.

Folgende Funktionen werden exportiert:

- *GenerateLinkBase()*
- *ParseHTMLLinks()*
- *Allowremove()*
- *uselink()*
- *countlinks()*
- *removerekursion()*
- *wholinksme()*
- *newLink()*
- *loadlink()*
- *addlink()*
- *updatelink()*
- *removelink()*
- *savelink()*
- *reloadlink()*

Auch dieses Modul nutzt die OOP. Mit der Funktion *newLink()* wird eine neue Referenz auf ein Link-Objekt generiert.

#### 5.2.1.7. Weitere Module

Neben den oben genannten Modulen wurden weitere Perl-Module verwendet, welche jedoch nicht explizit für das WCMS erstellt wurden und innerhalb von CPAN [Web32] öffentlich für die Nutzung verschiedenster Programme abrufbar sind. Es handelt sich dabei um die Module:

- web
- LWP
- HTML
- Data

### 5.2.2. Benutzer-Interface / Hauptprogramm

Das Hauptprogramm teilt sich auf in ein mit JavaScript gesteuertes Menü und einem CGI-Programm welches über SSI- und direkte Formular-Aufrufe angesprochen wird. Oberfläche und Programm laufen somit unabhängig voneinander wobei die über JavaScript ausgewählte Funktion entscheidet, wie das CGI-Programm aufgerufen wird.

Jede Aktion des CGI-Programms ist atomar realisiert, so daß bei einem Verlust der Verbindung während der Arbeit kein Datenverlust auftritt.

Die Ausgaben des CGI-Programms werden durch eigene Layout-Dateien realisiert, bei denen die dynamischen Anteile durch das Programm gefüllt werden.

Beim Aufruf des CGI-Programms ist der Ablauf immer gleich:

1. Einlesen der Benutzerkennung über die vom Apache Webservers modifizierte Variable REMOTE_USER. Danach Suchen und Laden der Daten für den Benutzer
2. Einlesen von Rechte-Definitionen und Überprüfung, ob der Benutzer das Recht hat die ausgewählte Funktion auszuführen.
3. Auswahl der auszuführenden Aktion
4. Ausführung der Funktion und damit ggf. Erstellung eines Formulars für den nächsten möglichen Schritt der Funktion.
5. Programmende

Je nachdem welcherart die unter Punkt 5 angegebene Funktion ist, werden zum Teil umfangreiche Operationen gestartet.

Sollte es während der Ausführung zu Fehlern kommen (z.B. wenn ein Benutzer eine Funktion aufruft zu der er kein Recht hat), wird das Programm sofort abgebrochen.

Die entsprechenden Teile zum obigen Vorgehen innerhalb des Quellcodes sehen folgendermaßen aus:

```
my $USER = newUser
USER_DBI($CONFIG{'USER_DB_F
ILE'});
```

```
if (not &isValid_User) {
        Fehlermeldung("Ihre Benut-
zerkennung ist abgelaufen oder nicht
g&uuml;ltig.");
}
```

**Quellcode 6. Einlesen und Prüfen der Benutzerkennung**

```
my %RCR =
&ReadCommandRightDefinitions;

my $PAGE_ACCESS_STATUS =
&AllowedPage;

if ($PAGE_ACCESS_STATUS==0) {
  if ($ENV{'DOCUMENT_NAME'} =~
/WorkPlace\.shtml/i) {
    &LoginWindow;
  } elsif
($ENV{'DOCUMENT_NAME'} =~
/Merkzettel\.shtml/i) {
    &MerkzettelWindow;
  } elsif
($ENV{'DOCUMENT_NAME'} =~
/NewPage\.shtml/i) {
```

**Quellcode 7. Einlesen und Prüfen der Rechte, sowie Funktionsauswahl I**

```
  }
} elsif ($PAGE_ACCESS_STATUS
<0) {
  if ($in{'action'} eq 'Merkzettel') {
    &ModifyMerkzettel;
  } elsif ($in{'action'} eq 'CreateDok-
Type') {
    &DoCreateDokType;
  } elsif ($in{'action'} eq 'DirCreate-
DokType') {
  } else {
```

**Quellcode 8. Funktionsauswahl für von Formularen gewählte Funktionen**

```
$USER->UpdateUserTimes;

exit(1);
```

**Quellcode 9. Programmende**

### 5.2.3. Überwachungsprogramm

Um den Content-Lifecycle abzubilden reicht ein Aufruf des Programms als CGI nicht aus. Es muß ein weiteres Programm im Hintergrund laufen, welches in regelmäßigen aber häufigen Intervallen die einzelnen Seiten und Objekte überprüft: Wenn die Lebenszeit eines Objektes abgelaufen ist, ist es die Aufgabe dieses Überwachungsprogramms über die Aufgaben-Verwaltung die für die Seiten und das Objekt zuständigen Personen zu informieren und Änderungen einzufordern.

Weiterhin ist es die Aufgabe des Programms, die Alarmierung wenn nötig auszudehnen. Über die Prioritäten-Angabe einer Aufgabe sind Benachrichtigungsverfahren definiert, welche nach einer gewissen Zeit die Alarmierung in kürzer werdenden Abständen wiederholen. Welcherart Priorität verwendet wird, ist durch die Wahl des veralteten oder zu ändernden Dokumententyps festgelegt.

```
1    low    -        gering
2    medium          EMAIL: 10 days
     <font color="green">mittel</font>
3    high    EMAIL: 5 days; REPEAT:
3 days       <B><font
color="red">hoch</font></B>
4    highest         EMAIL: 2 days;
REPEAT: 1 day  <B><font
color="red"><blink>h&ouml;chste</b
link></font></B>
```

**Quellcode 10. Prioritäten**

In Quellcode 10 ist der Inhalt der von dem TODO_DBI-Modul genutzten Prioritäten-Datei aufgeführt.
Es handelt sich dabei um eine CSV-Datei, wobei mit dem Tabulatorsprung getrennt wurde.
In der ersten Spalte sieht man die Nummer der Priorität. Die zweite Spalte enthält die interne textuelle Umschreibung dazu. Die Dritte enthält die Angabe in welchen Intervallen eine Alarmierung erfolgt. Die vierte Spalte enthält die HTML-formatierte Ausgabe, mit welcher ein Benutzer über das Web-Interface die Aufgabe deklariert sieht.
Wichtig an den Angaben ist die dritte Spalte: Die Angabe, was bei einer Alarmierung durch eine Aufgabe mit der betreffenden Priorität passieren soll.
Bei einer Priorität „high“ wird 5 Tage nach der ersten Alarmierung eine weitere folgen. Wird auch danach die Aufgabe nicht bearbeitet, wird jetzt die Alarmierung alle 3 Tagen wiederholt.
Optional ist es möglich das Überwachungsprogramm so einzustellen, daß eine Aufgabe nach Nichtbearbeitung nach einer gewissen Zeit in der Priorität steigt.

# 6. Einsatz und Test eines WCMS

## 6.1. Fallbeispiele

In diesem Kapitel wird anhand von Fallbeispielen der Betrieb des durch diese Arbeit implementierten WCMS beschrieben.

Alle folgenden Bildschirmbilder wurden aus dem laufenden Betrieb des Systems, wie es derzeit für das Regionale Rechenzentrum Erlangen (RRZE) eingerichtet ist, entnommen.

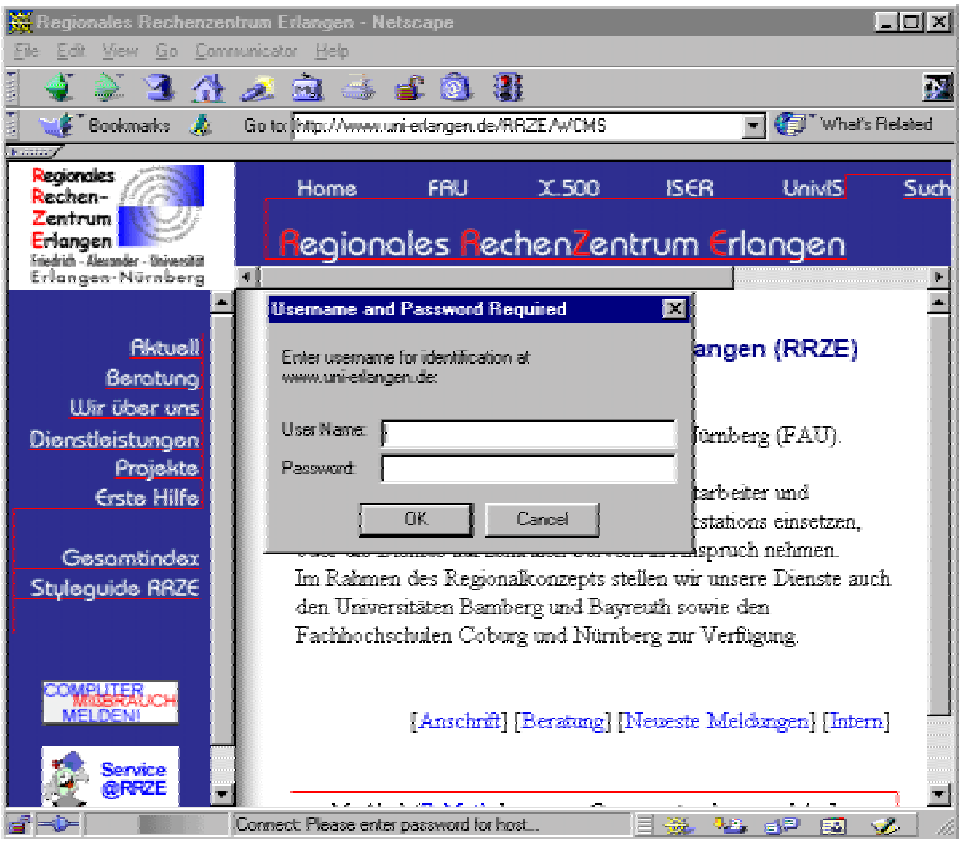



**Abb. 16. Login (1)**

### 6.1.1. Login

Der Login für einen registrierten Benutzer erfolgt über die vom Webserver Apache bereitgestellte Zugriffskontrolle. Das WCMS hat auf diese insoweit Einfluß, daß es eine seiner Aufgaben ist, Benutzer anzulegen oder zu löschen. Nach Eingabe eines gültigen Benutzernamens und Passwortes, wie auf Abbildung 16 gezeigt, erhält man Zugriff auf das eigentliche System. Der Eingangsbildschirm wird in Abbildung 17 gezeigt.

Nach dem Einloggen bleibt man dabei nicht in dem vorher genutzte Browserfenster, sondern es wird ein neues erstellt, worin das WCMS dargestellt wird.

Dieses neue Fenster ist auf die Standard-Größe von 800x600 Pixel festgelegt und besitzt nur einen eingeschränkten Zugriff auf Funktionen des Browsers. (Menü-,

**Abb. 17. Login(2)**

Adreß- und Bookmarkleiste sind abgeschaltet).

Die Bedienoberfläche ist aufgeteilt in 3 Teile: Hauptmenü (oben), Funktionsmenü (links), Arbeitsfenster (großer Bildschirmbereich mit weißem Hintergrund). Das Arbeitsfenster enthält beim Einloggen eine Begrüßung, einen Hinweis über den aktuellen Inhalt der Aufgabenliste, sowie Angaben zu dem eigenen Personenstatus. Unter den Angaben zum Personenstatus befindet sich zusätzlich eine Auflistung von persönlichen Notizen, den Merkzetteln. Die Merkzettel arbeiten unabhängig vom eigentlichen System und dienen nur dem Festhalten von eigenen Notizen.

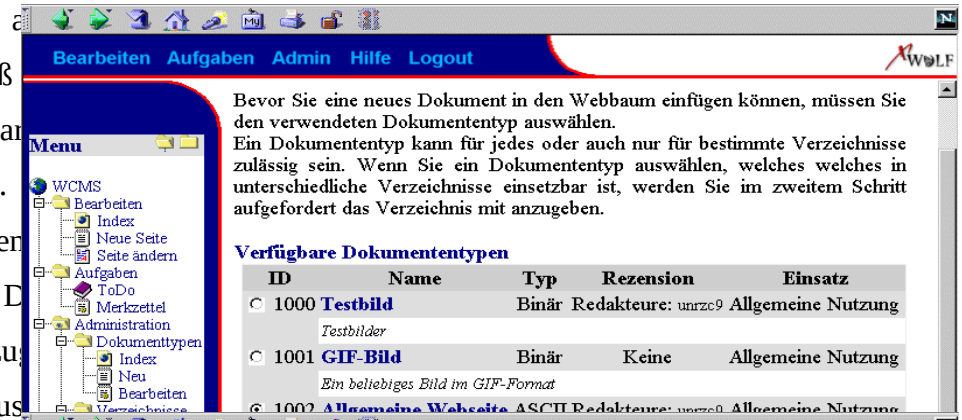
Die Notizen aus dem Merkzettel werden in der Regel bei jeder Aktion des WCMS mit angezeigt, es sei denn, der Bildschirmplatz ist nicht ausreichend.

### 6.1.2. Seite erstellen

Bei der Erstellung eines Seite sind im Vorfeld einige Fragen zu regeln, damit bekannt ist, welche Statusdaten die Seite hat: Wo soll die Seite gespeichert werden, nach welchen Dokumententyp erfolgt die Eingabe, welche anderen Objekte sollen in die Seite eingebunden werden, etc. Aus diesem Grund ist die Eingabe der Inhalte einer Seite der letzte Schritt bei der Erstellung. Die folgenden Abbildungen zeigen das Vorgehen.

Abbildung 18 zeigt den ersten Schritt bei der Erstellung einer neuen Seite: Die Auswahl des Dokumententyps. Man erhält dazu eine Auswahl der Dokumententypen

angezeigt, a
diesen muß
ren des Mar
auswählen.
Im nächsten
durch den D
zeichnis zu
zeichnis aus
eine Auswa
zeichnissen.
gig, in welc
wählten
werden kann
man selbst Z



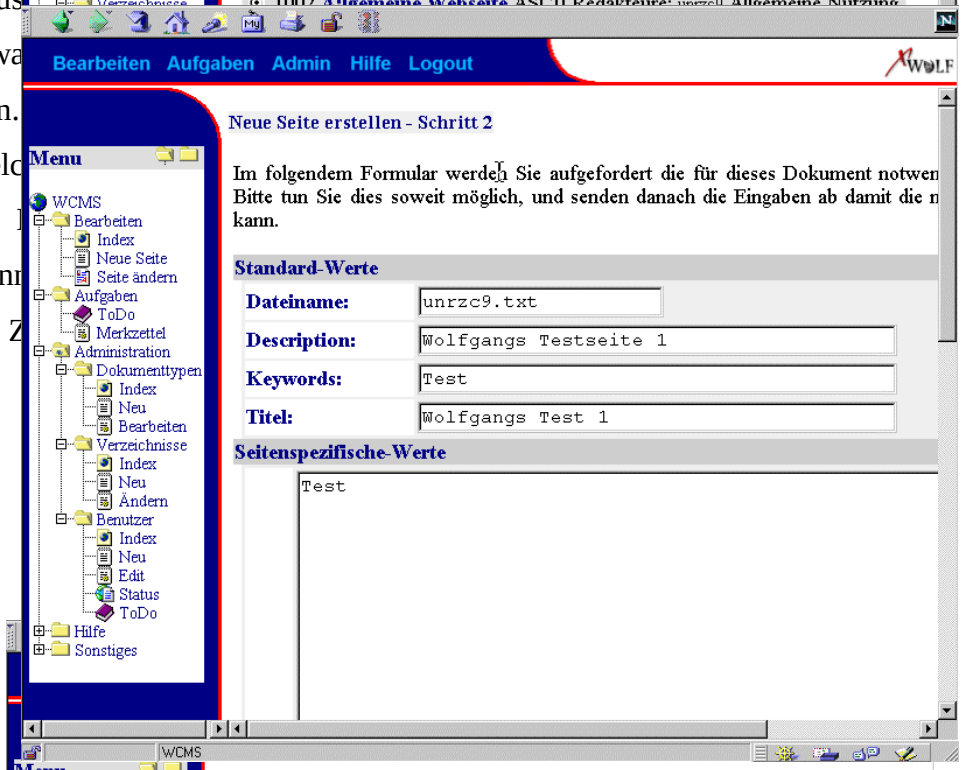



**Abb. 20. Seite erstellen (3)**

**Abb. 19. Seite erstellen (2)**

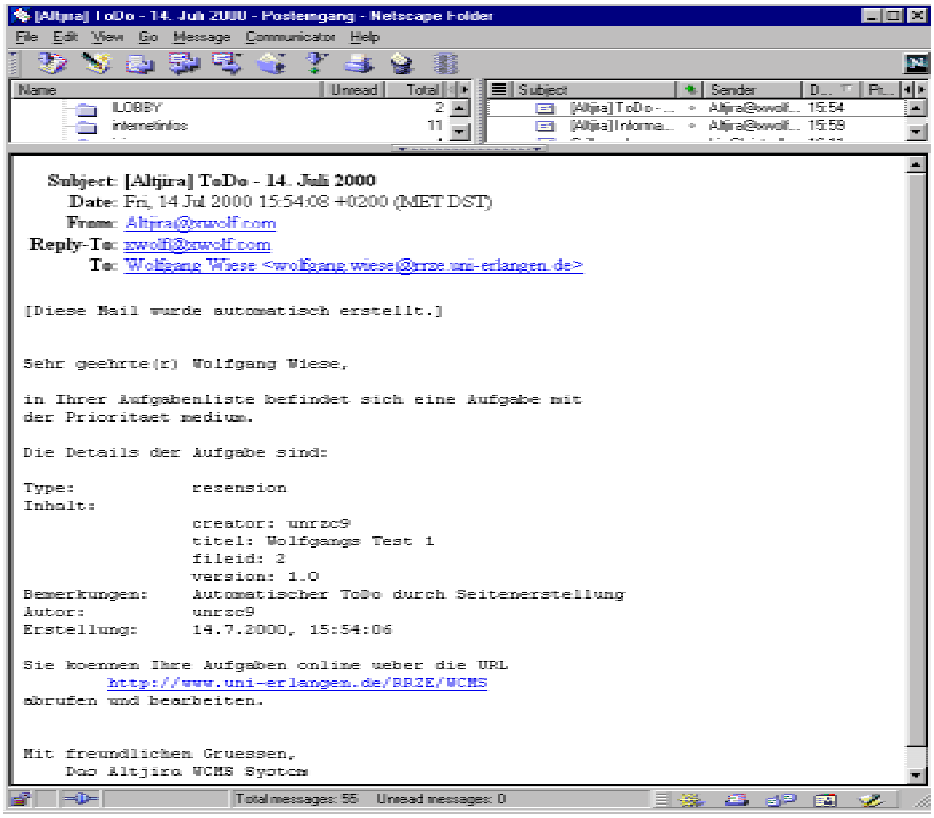



**Abb. 21. ToDo-Alarmierung**

Nachdem Zielverzeichnis und Dokumententyp festgelegt sind, können die Seitendaten eingegeben werden. Im Dokumententyp ist angegeben, welche Eingabefelder zur Verfügung stehen müssen. Diese Eingabefelder werden mit dem gesamten Formular dynamisch erstellt. Nach Eingabe dieser Daten wird die Seite erstellt, bzw. zur Freigabe weitergeleitet.

Soll die Seite von einem Benutzer freigegeben werden, wird dieser durch die Redakteur-Angabe in der Dokumententyp-Definition ermittelt. Es startet also mit dem abschicken des Formulars aus Abb. 20 ebenfalls eine Alarmierung durch das Aufgaben-Management.

Da die Alarmierung mittels E-Mail implementiert ist, erhält der Redakteur für die Seite die in Abbildung 21 gezeigte E-Mail. Wie man in Abb. 21 lesen kann, steht in der E-Mail Status-Angaben zur Aufgabe.

Wichtig an der Mail ist jedoch nur der Hinweis für den Redakteur, daß eine neue Aufgabe vorliegt. Beim Aufruf des Aufgaben-Managements innerhalb des Systems, erhält der Redakteur diese Angaben-Daten in detaillierterer Form präsentiert.

### 6.1.3. Aufgaben-Management abrufen und Seite freigeben

Nachdem der Redakteur eine Alarmierung über eine neue Aufgabe erhält, ist er ge-

halten, das WCMS aufzurufen und das Aufgaben-Management zu kontrollieren. Die Abb. 22 zeigt die Aufgabenliste nachdem die E-Mail aus Abb. 21 einging.

Im Aufgaben-Management kann jetzt die vorhandene Aufgabe bearbeiten und die Seite freigeben werden[18].

Abbildung 23 zeigt die Ausgabe die erscheint, wenn bei der Aufgabenübersicht aus Abb. 22 sich dazu entschlossen wird, die Aufgabe abzuarbeiten und die Seite freizugeben, bzw. zu rezensieren.

Es erscheint eine Angabe der Statusdaten für die freizugebende Seite. Anhand von Buttons ist es jetzt möglich die Seite freizugeben oder aber sie abzulehnen. Wird die Seite ablehnt, ist eine Begründung anzugeben, welche der Autor erhält.

Nach Freigabe erhält der Autor der Seite eine Benachrichtigung mittels E-Mail, die über die Freigabe informiert.

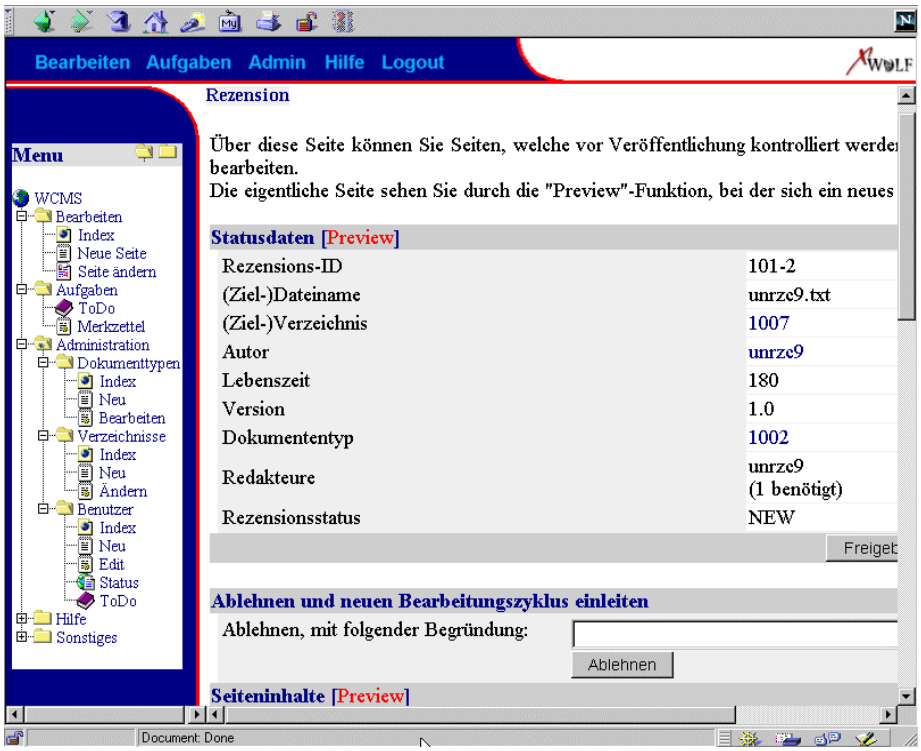



**Abb. 23. Seite freigeben**

[18] Notiz: In dem aktuellen Beispiel sind Ersteller und Redakteur dieselbe Person. Dies widerspricht nicht dem Kontroll- und Freigabe-Zyklus sondern ist eine Eigenschaft des Dokumententypes.

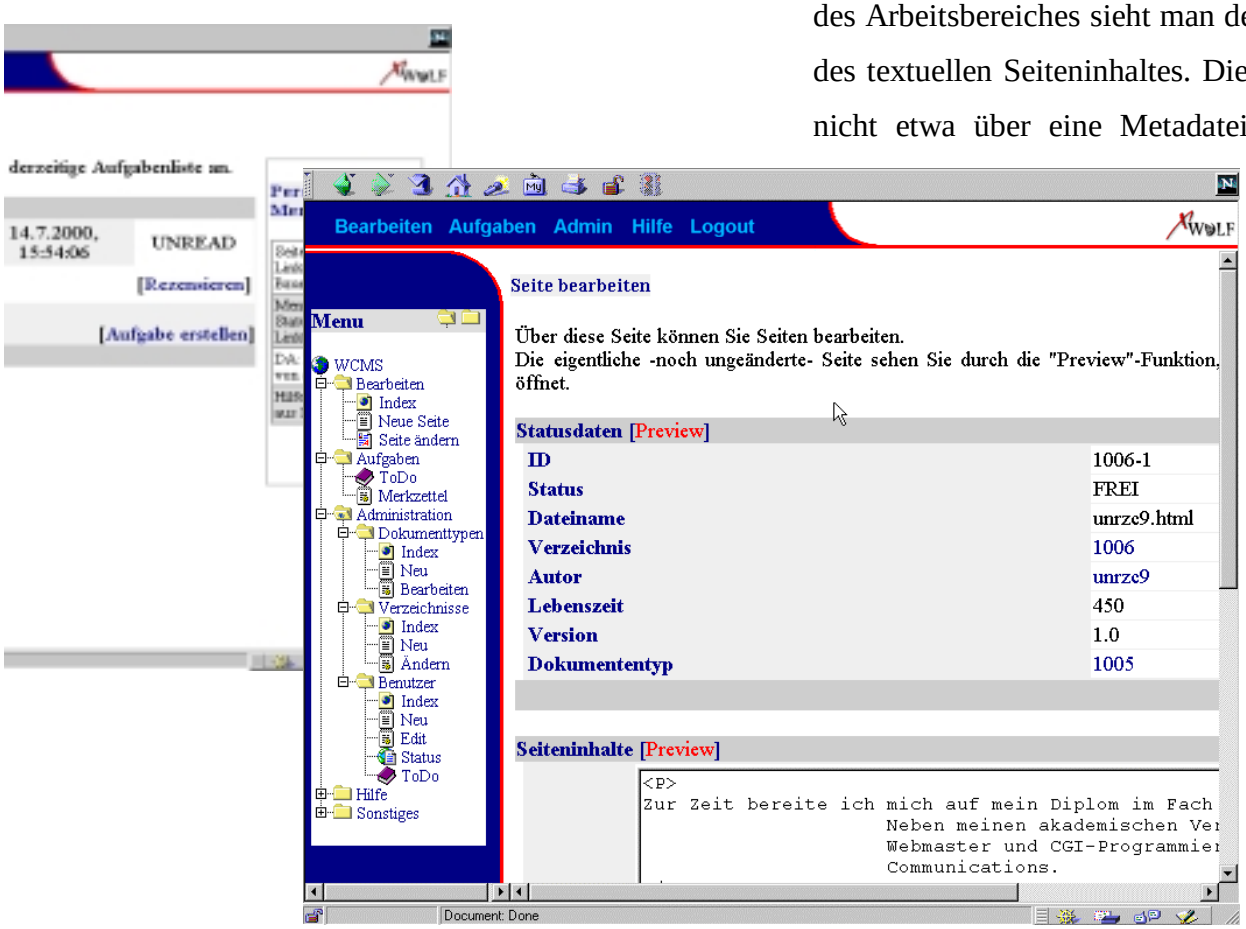



**Abb. 24. Seite bearbeiten**

### 6.1.4. Seite bearbeiten

Der Unterschied in der Anzeige zwischen Freigabe und Bearbeitung einer Seite, ist gering: Da intern bei der Freigabe, wie beim Bearbeiten, auf die Daten der vorhandenen (temporären) Seite zugegriffen werden muß, liegen dieselben Statusdaten vor.

Nachdem eine schon freigegebene Seite bearbeitet wurde, wird die Änderung ebenfalls wieder in den Kontroll- und Freigabezyklus geschoben. Erst nach Freigabe wird die vorherige und veröffentlichte Version überschrieben.

Abbildung 24 zeigt die Ausgabe für die Bearbeitung einer Seite. Im unteren Teil des Arbeitsbereiches sieht man den Anfang des textuellen Seiteninhaltes. Dieser wurde nicht etwa über eine Metadatei, sondern mittels des PSE-Konzeptes erhalten: Vor der Ausgabe des Arbeitsformulars hat das Programm die bestehende Seite gelesen und den Inhalt extrahiert.

Somit ist sicher gestellt, daß auch wirklich die Seiteninhalte bearbeitet werden, die auch veröffentlicht[19] sind.

[19] War die vorherige Seitenversion noch nicht veröffentlich, sondern in einem internen Arbeitszyklus (Draftmodus) befindlich, wird nicht auf die veröffentlichte Version zugegriffen, sondern auf die im Arbeitszyklus.

### 6.1.5. Benutzer-Administration

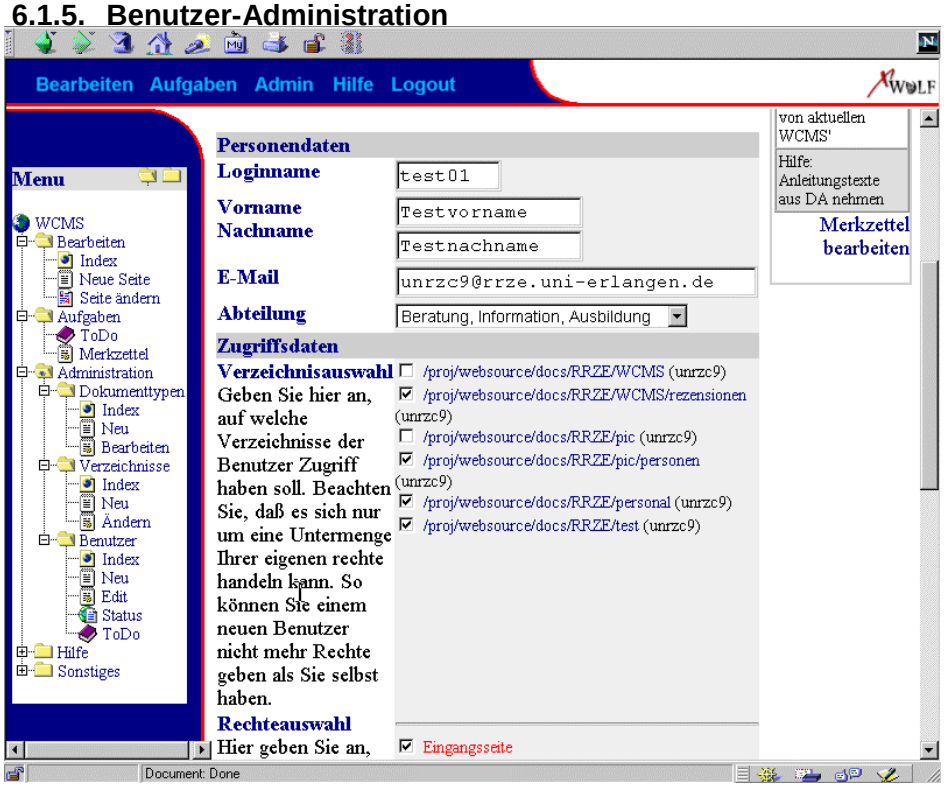



**Abb. 25. Benutzer anlegen(1)**

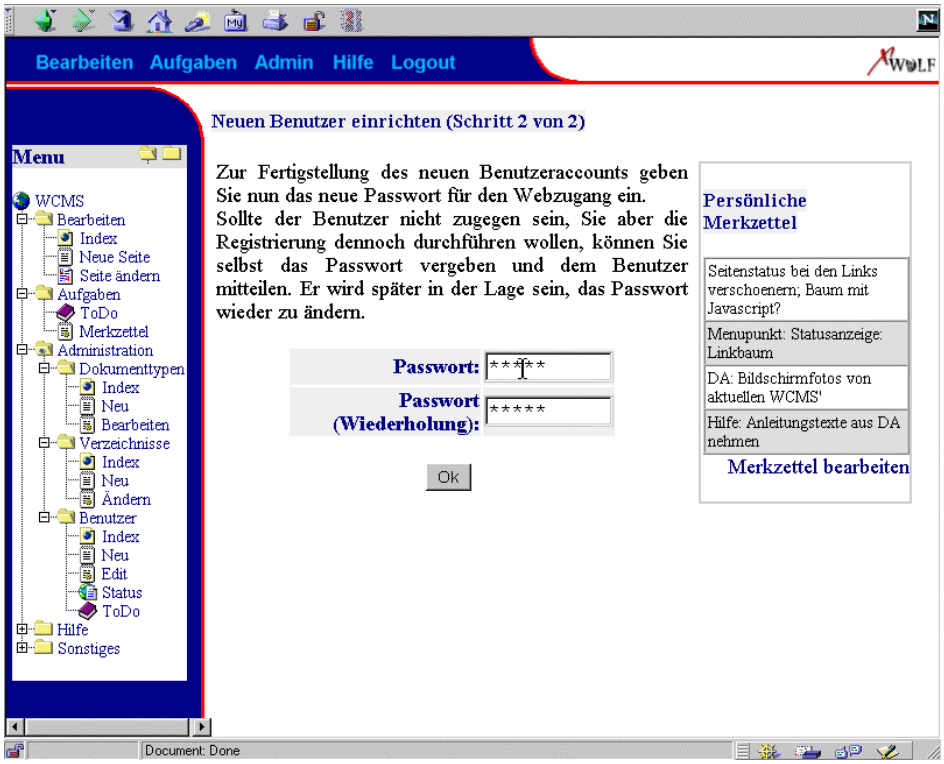



**Abb. 26. Benutzer anlegen(2)**

Nach Eingabe dieser Daten wird noch nach einem Passwort gefragt, mit dem sich der neue Benutzer zukünftig anmelden kann (Abbildung 26).

### 6.1.6. Dokumententyp-Verwaltung

Da ohne Dokumententyp-Definition keine neuen Seiten erstellt werden können, muß es eine Möglickeit geben, neue Dokumententypen anzulegen.

Abbildung 28 zeigt das Formular bei der Erstellung eines neuen Dokumententyps.
Je nach Wahl des Einsatzortes des Dokumententyps folgt außerdem eine Abfrage in welchen Verzeichnissen dieser gültig sein soll (Abbildung 29).

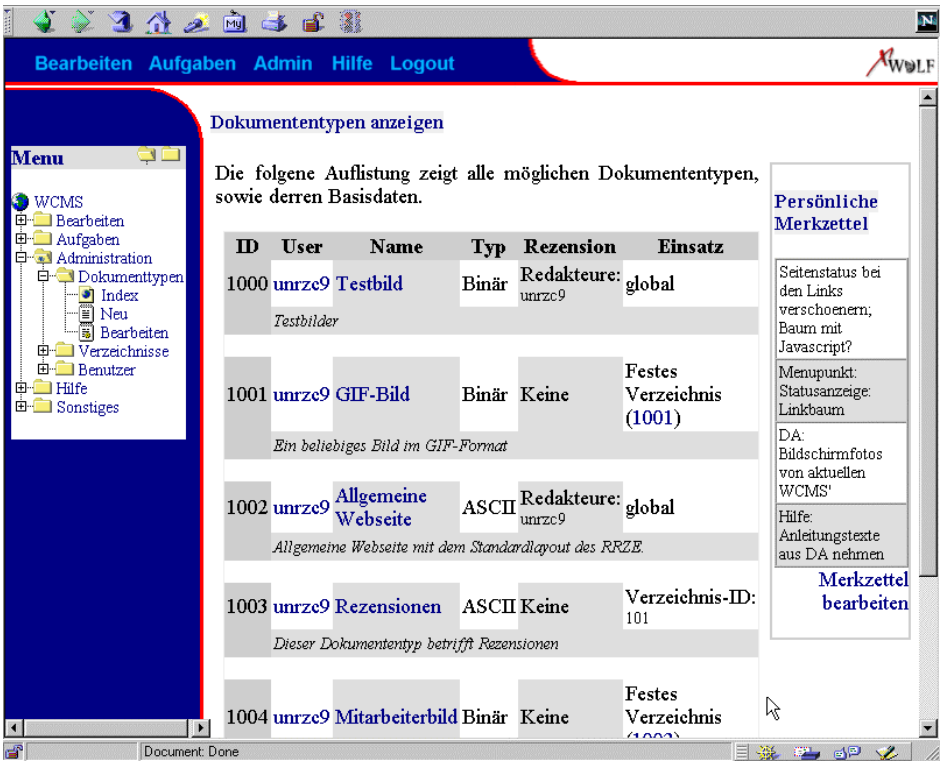



**Abb. 27. Dokumententypen**

Abbildung 27 zeigt den Index der vorhandenen Dokumententypen. Man sieht die ID, den Loginnamen des Erstellers, die Art des Content-Typs, ob ein Kontroll- und Freigabezyklus erfolgen soll und den Ort des Einsatzes.

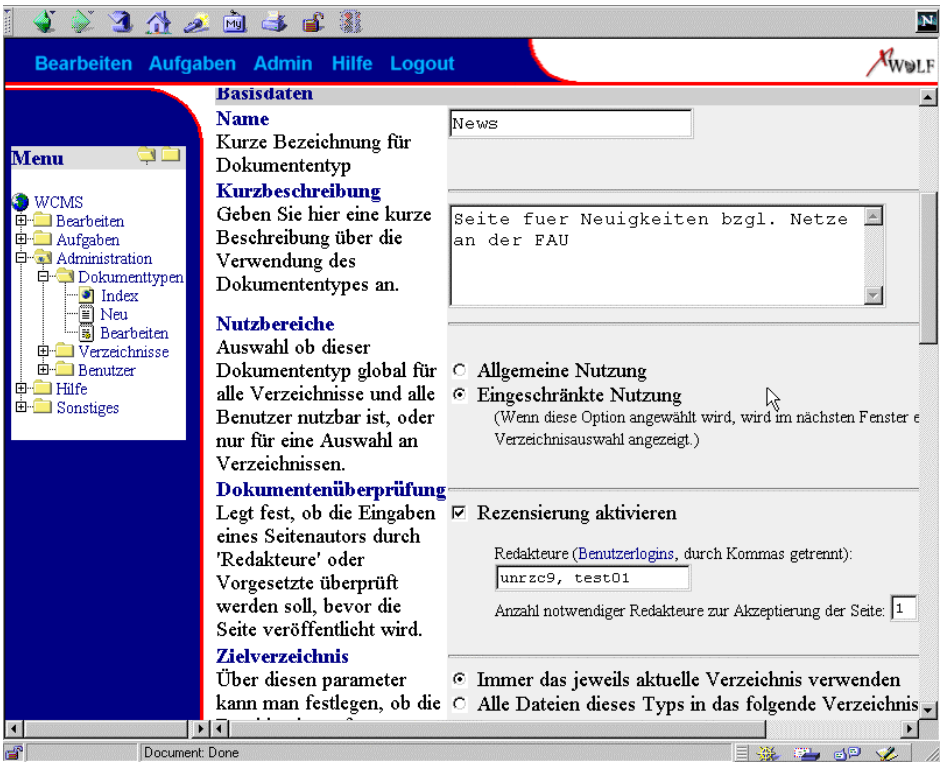



**Abb. 28. Dokumententyp anlegen(1)**

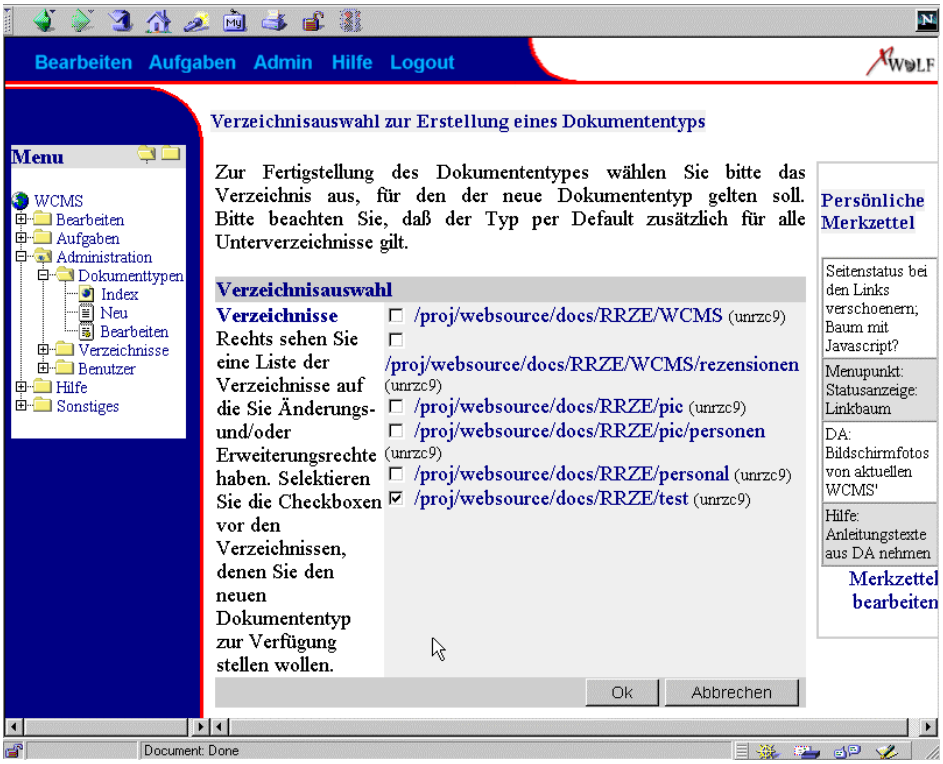



**Abb. 29. Dokumententyp anlegen(2)**

### 6.1.7. Verzeichnisse

Wie in den vorherigen Abbildungen bereits mehrmals gezeigt, können bei den Auswahlfelder für Verzeichnisse, einzelne Verzeichnisse angeklickt werden. Wird dies getan, erscheinen die Statusdaten des Verzeichnisses. Abb. 30 zeigt die Statusdaten

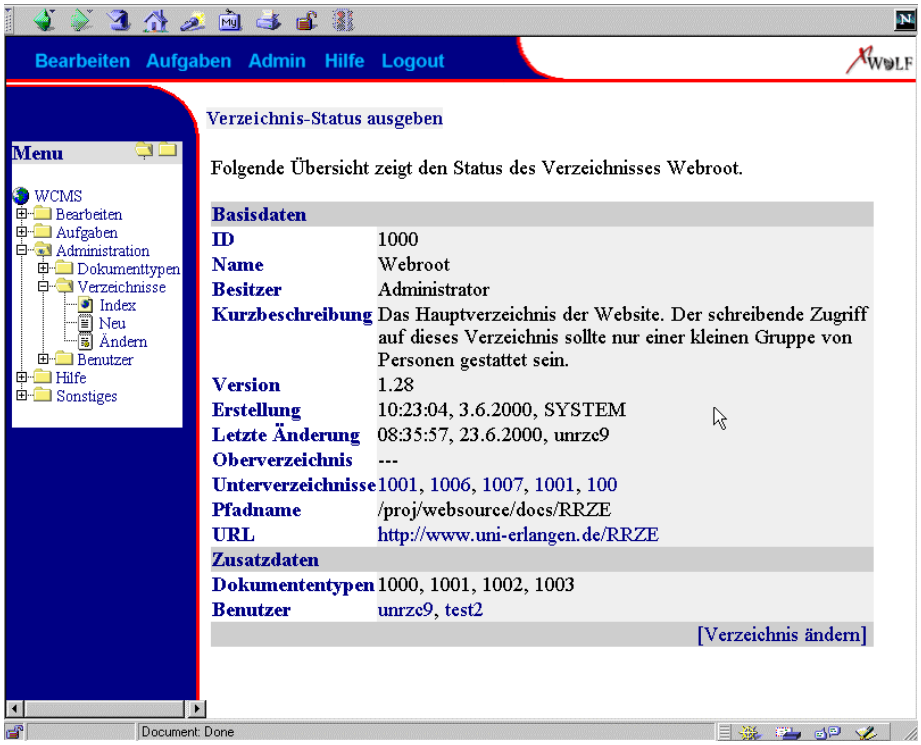



**Abb. 30. Verzeichnis-Status**

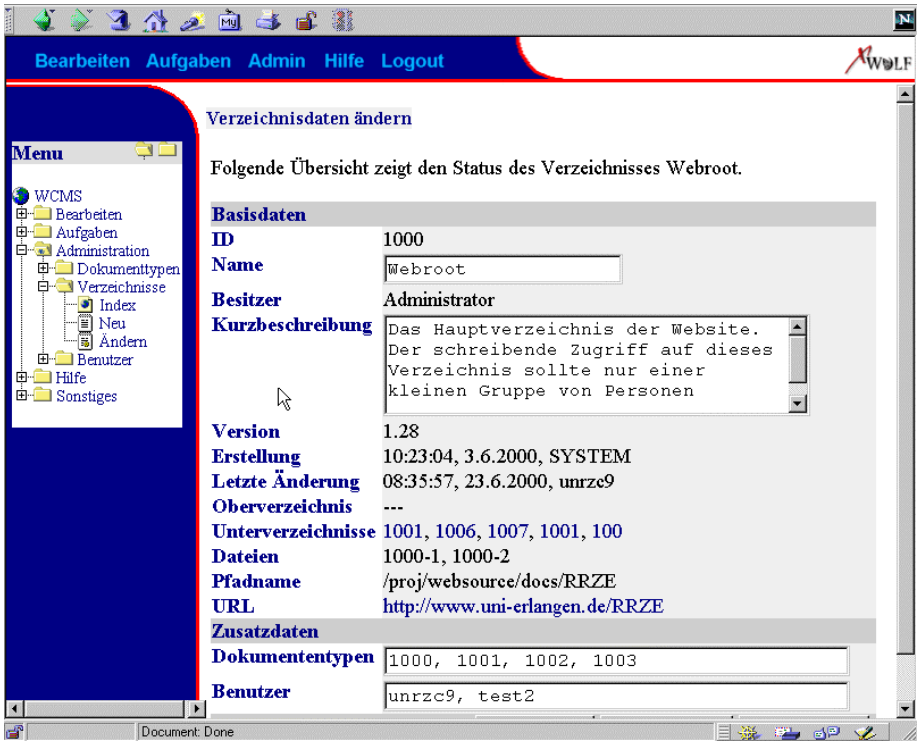



**Abb. 31. Verzeichnis ändern**

für das Root-Verzeichnis. Soll dieses Verzeichnis geändert werden (und besteht das Recht dazu), kann der Link „[Verzeichnis ändern]“ gefolgt werden und erscheint ein Bild wie in Abbildung 31.

## 6.2. Datensätze

Dieses Kapitel gibt die technische Umsetzung ausgewählter Datensätze wieder, wie sie im Pflichtenheft unter Kapitel 5.1.2.5 „System-Daten“ definiert wurden. Alle Dateien sind im XML-Format gespeichert.

Die Angabe der DTD wurde der Übersichtlichkeit halber weggelassen.

### 6.2.1. Benutzer-Daten


```xml
<?xml version="1.0"?>

<file name="last-modification-time" value="19:20:38">
<file name="last-modification-date" value="12.7.2000">
<file name="creation-date" value="1.4.2000">
<file name="creation-time" value="11:00:12">
<comment> In dieser Date finden sich die Personendaten fuer einen laufenden
oder noch kommenden Kurs. </comment>
<author name="last" value="unrzc9">
<author name="created" value="unrza4">
<version>1.105</version>
<user>
    <name>Wiese</name>
    <verzeichnisse>1000, 1006, 1007, 1008</verzeichnisse>
    <url>http://www.xwolf.com</url>
    <todofile>db/todo/unrzc9.xml</todofile>
    <freeformname>Wolfgang Wiese</freeformname>
    <email>unrzc9@rrze.uni-erlangen.de</email>
    <vorname>Wolfgang</vorname>
    <login>unrzc9</login>
    <follower>tester, xwolf, xtest, iwi703, dkbaum</follower>
    <rechte>100, 110, 120, 200, 210, 220, 230, 240, 250, 260, 270, 310, 321, 322,
323, 330, 411, 412, 413, 415, 416, 421, 422, 423, 424, 430, 440, 510,
611</rechte>
    <abteilung>Beratung, Information, Ausbildung</abteilung>
    <time id="creation">3.4.2000</time>
    <time id="lastconnect">12.7.2000</time>
    <time id="lastaction">10.7.2000</time>
</user>
<user>
    <name>-</name>
    <verzeichnisse></verzeichnisse>
    <todofile></todofile>
    <freeformname>Admin -</freeformname>
    <vorname>Admin</vorname>
    <email>wcms@rrze.uni-erlangen.de</email>
    <login>Administrator</login>
    <follower>test2</follower>
```

```
    <rechte>100, 110, 120, 210, 220, 230, 240, 250, 260, 270, 310, 321, 322, 323,
330, 411, 412, 413, 415, 416, 421, 422, 423, 424, 430, 440, 510, 611</rechte>
    <abteilung>Beratung, Information, Ausbildung</abteilung>
    <time id="creation">18.6.2000</time>
    <time id="lastconnect">20.6.2000</time>
    <time id="lastaction">20.6.2000</time>
</user>
```

**Quellcode 11. Auszug aus der Datei „user.xml“**

### 6.2.2. Dokumententyp-Daten

```
<?xml version="1.0"?>

<file name="last-modification-date" value="23.6.2000">
<file name="last-modification-time" value="13:06:52">
<file name="creation-date" value="7.6.2000">
<file name="creation-time" value="14:31:12">
<author name="last" value="unrzc9">
<author name="created" value="SYSTEM">
<page id="1004">
    <creator>unrzc9</creator>
    <name>Mitarbeiterbild</name>
    <type>global</type>
    <targetdir>1002</targetdir>
    <check_ok_num>1</check_ok_num>
    <content-type>bin_gif</content-type>
    <CREATE_INDEX>0</CREATE_INDEX>
    <verfallszeit>450</verfallszeit>
    <check_user>unrzc9</check_user>
    <id>1004</id>
    <layout_file>NO-LAYOUT</layout_file>
    <rezension>OFF</rezension>
    <require>Description, Keywords, Titel</require>
    <desc>GIF-Bild von einem Mitarbeiter.</desc>
</page>
<page id="1005">
    <linkfelder>Homepage</linkfelder>
    <name>Homepage</name>
    <type>global</type>
    <linkattribute></linkattribute>
    <targetdir>1006</targetdir>
    <check_ok_num>1</check_ok_num>
    <content-type>nf_html</content-type>
    <CREATE_INDEX>0</CREATE_INDEX>
    <binfelder>Bild1:1004</binfelder>
    <id>1005</id>
    <binattribute>width=90 height=120</binattribute>
    <creator>unrzc9</creator>
    <textfelder>Text1, Text2</textfelder>
    <textattribute>cols=70 rows=10</textattribute>
    <verfallszeit>450</verfallszeit>
```

```
    <check_user>unrzc9</check_user>
    <require>Description, Keywords, Titel</require>
    <rezension>OFF</rezension>
    <layout_file>/proj/webbin/cgi-
bin/wcms/db/pagelayout/homepages.html</layout_file>
    <desc>Layout fuer eine Standard-Homepage eines Mitarbeiters.  </desc>
</page>
```

**Quellcode 12. Auszug aus der Datei „pagetype.xml“**

### 6.2.3. Verzeichnis-Daten

```
<?xml version="1.0"?>

<file name="last-modification-date" value="10.7.2000">
<file name="last-modification-time" value="22:05:29">
<file name="creation-date" value="18.6.2000">
<file name="creation-time" value="19:18:10">
<name>Testverzeichnis</name>
<updir>1000</updir>
<version>1.15</version>
<owner>unrzc9</owner>
<id>1007</id>
<path>/proj/websource/docs/RRZE/test</path>
<link name="tadefo" type="dir">
<lifetime>180</lifetime>
<files>1007-1</files>
<comment>Verzeichnis</comment>
<url>http://www.uni-erlangen.de/RRZE/test</url>
<author name="last" value="unrzc9">
<author name="created" value="unrzc9">
<filetypes>1000, 1001, 1002, 1003</filetypes>
<dirmodes>755</dirmodes>
<filemodes>644</filemodes>
<rechte>ALLOWLINKS, READ, WRITE, MKDIR, SETUSER, SETFILETYPES,
RMDIR, CHANGEOWNER, CREATESUBFILETYPES</rechte>
<subdirs></subdirs>
<user>unrzc9, tester, Administrator, test2, xtest, iwi703, dkbaum</user>
<desc>Test - Bitte ignorieren   </desc>
```

**Quellcode 12. Verzeichnis-Datei “1007.xml”**

### 6.2.4. Seiten-Daten

```
<?xml version="1.0"?>

<file name="last-modification-date" value="29.6.2000">
<file name="last-modification-time" value="14:18:41">
<file name="creation-date" value="27.6.2000">
<file name="creation-time" value="09:55:07">
<author name="last" value="unrzc9">
<author name="created" value="unrzc9">
```

```
<version>1.11</version>
<datei id="1">
    <log size="10">
            <entry id="1" text="(29.6.2000 - 13:50:14) Erstellung durch unrzc9">
            <entry id="2" text="(29.6.2000 - 14:18:41) Rezensionsfreigabe durch
unrzc9">
    </log>

    <type>1002</type>
    <titel>Aenderungsprotokoll</titel>
    <status>REZENSION</status>
    <Description>protokoll</Description>
    <editor name="lastchange" value="unrzc9">
    <editor name="creation" value="unrzc9">
    <dateiname>protokoll.html</dateiname>
    <Keywords>protokoll</Keywords>
    <owner>unrzc9</owner>
    <id>1</id>
    <dateiversion>1.0</dateiversion>
    <time name="lastchange" value="29.6.2000">
    <time name="creation" value="29.6.2000">
    <lifetime>180</lifetime>
</datei>
```

**Quellcode 13. Seitendaten-Datei „1007-files.xml“ zum Verzeichnis mit der ID 1007**

# 7. Klassifikation und Vergleich

## 7.1. Klassifikation

Wenn ein Klassifikationsschema für WCM-Lösungen aufgestellt werden soll, ergibt sich zunächst die Frage, wonach sich diese richten soll: Sollen mehr technische Aspekte im Vordergrund stehen, organisatorische, die Art der Benutzeroberfläche oder der Zeitaufwand für das Erlernen der Bedienung, oder ganz andere? Lösen läßt sich die Frage nur schwer. Ein Ansatz ist, benutzte Standards und Grundlagen als Elemente für die Klassifikation zu definieren.

Diese sind bei einem WCMS folgende:

- Unterstützte Plattformen.
- Art der Speicherung von Dokumenten.
- Art der Publizierung von Dokumenten.
- Unterstützung des XML-Standards.
- Skalierbarkeit.

Für Unternehmen sind außerdem folgende Punkte interessant:

- Preis-Leistungs-Verhältnis
- Zahl der benötigten Vollzeit-Mitarbeiter zur Bedienung (Administration)
- Schulungskosten

Ein anderer Ansatz zur Klassifikation besteht in der Definition von Entwicklungsphasen von WCM-Lösungen. Diese können wie folgt aussehen:

| Generation | Funktionalität |
|---|---|
| 1. | Editoren mit Site-Management. Klassisches CM mit Web-Export. |
| 2. | WCMS ohne Import bestehender oder geänderter Webseiten vom Webserver oder anderen Editoren. Seiten können nur durch das WCMS erstellt werden. |
| 3. | WCMS mit einmaligen Import bestehender Seiten. Danach nur noch Export durch das WCMS. |
| 4. | WCMS bei dem jederzeit sowohl Import, als auch Export von Seiten erlaubt ist und somit auch andere Produkte auf die Seiten ändernd zugreifen dürfen. |
| 5. | WCMS der Stufe 4, jedoch zusätzlich mit voller Funktionalität eines HelpDesk-Systems. |

Neben diesen beiden Ansätzen sind auch andere möglich, bei der die Bewertung mit Hilfe eines Punktesystems das Ziel ist.

Ein solches Klassifikationsschema läßt sich jedoch nur schwer erstellen, da eine Bewertung anhand von Punkten große Erfah-

rungen in den bestehenden Systemen und deren technische Möglichkeiten voraussetzt.
Da jedoch der Markt für WCM-Lösungen derzeit erst im Entstehen ist, bzw. sich die ersten größeren Hersteller erst mit der CeBIT 2000 in diesem Marktsegment zu etablieren begannen, ist derzeit eine objektive Festsetzung für ein Punktesystem noch nicht möglich.

Aus denselben Gründen ist eine Klassifikation nur unter Einschränkungen möglich. Als Beispiel kann das Produkt SchemaText von der Schema GmbH [Web33] genannt werden. Ausgehend von den Prinzipien des reinen Content- und Dokumenten-Managements konnte dieses noch vor Kurzem innerhalb der oben definierten Entwicklungsphase 2 eingeordnet werden.
Durch ein sehr aktives Entwicklerteam ist SchemaText aber inzwischen auf Phase 3 gelangt. Auch sollte es keine größeren technischen Probleme machen, statt nur einen einmaligen manuellen Import von Webseiten anzubieten, auch eine automatische Übernahme und Kontrolle von durch andere Produkte bearbeitete Webseiten einzubauen.
Der Aufstieg in die Entwicklungsphase 4 ist damit eher eine Frage der Unternehmensentscheidung. Im Vergleich dazu ist das diese Arbeit erstellte Produkt zwar Teil der Entwicklungsphase 4, hat jedoch auf der anderen Seite nicht die Features, die SchemaText (und andere Produkte) in Bezug auf Benutzerfreundlichkeit anbieten. (Beispiel: WYSIWYG-Modus).
Die Entwicklungsphase 5 wurde bei der Niederschrift dieser Arbeit von keinem der verfügbaren Produkte erreicht.

Die folgende Klassifizierung beschränkt sich auf die Einordnung anhand des verwendeten Speicherkonzeptes (Publishing-/Staging-Server, dynamisches Publishing, PSE-Konzept):

**Produkte mit Publishing-/Staging-Server:**

- SchemaText
- Imperia
- MKS Web Integrity

**Produkte mit dynamisches Publishing:**

- NPS Productivity System 4
- POET Content Management Suite
- Vignette Storyserver
- VIP’Content Manager 3.0

**Produkte mit PSE-Konzept:**

- Altjira WCMS (das für diese Arbeit erstellte System)

## 7.2. Trendanalyse

Bei 11 Herstellern von WCMS konnte durch eine Umfrage die Funktionalität ihres Produktes anhand von vorgegebenen „Ja“/“Nein“/“Weiß nicht“-Fragen ermittelt werden. Je nach Qualifikation des Ansprechpartners bzw. weil das Produkt in dem jeweiligen Punkt noch ausgebaut werden sollte, kam es auch zu einigen „Weiß nicht“-Antworten.

Bei den Anbietern bzw. Produkten handelt es sich im einzelnen um ActiveWeb, Roxen, MKS, Imperia, PC-Ware, Bladerunner, iRacer, Media Artists, POET, Community Engine, NPS. (Vergleiche Anhang A1.4 „Weitere Quellen“ und A3 „Anbieter von WCMS auf der CeBIT 2000“). Zusätzlich wurde das in Kapitel 5 entwickelte Produkt berücksichtigt.

Die folgende Tabelle zeigt die Ergebnisse der Umfragen.

| **Frage** | **Ja** | **Nein** | **Weiß nicht** |
|---|---|---|---|
| Sind Benutzer-Accounts durch Abteilungsleiter eintragbar? | 50% | 8% | 42% |
| Benutzer nur durch zentrale Benutzer-verwaltung eintragbar? | 16% | 42% | 42% |
| Eingaben über Client? | 25% | 75% | 0% |
| Eingaben über Browser? | 83% | 0% | 17% |
| System ist plattformunabhängig? | 58% | 42% | 0% |
| SQL-Schnittstelle vorhanden? | 67% | 0% | 33% |
| XML-Ausgaben möglich? | 75% | 0% | 25% |
| Linküberprüfung vorhanden? | 67% | 0% | 33% |
| Seiten nur im WCMS änderbar? | 50% | 33% | 17% |
| Können Seiten vorhanden sein, die vollständig durch andere Produkte erstellt werden? | 33% | 25% | 42% |
| Werden Seiten nach Freigabe sofort publiziert? | 58% | 17% | 25% |
| Ist eine Versionskontrolle für Webseiten vorhanden? | 83% | 0% | 17% |
| Können mehrere Personen oder Gruppen an einer Seite arbeiten? | 58% | 0% | 42% |
| Sind Benutzergruppen anlegbar? | 58% | 0% | 42% |
| Enthält die Benutzer-verwaltung Alarmierungsfunktionen? | 67% | 0% | 33% |
| Sind Importfunktionen für WORD, etc. vorhanden? | 50% | 42% | 8% |

Abgesehen von Detailfragen, bei denen die Ansprechpartner oft nicht gut genug informiert waren, lassen sich folgende Schlüsse darüber aufstellen, was ein WCMS heute enthalten und können muß:

- Plattformunabhängigkeit.
- Benutzeradministration, welche die organisatorische Struktur des Betriebes wiederspiegelt (Gruppen, Abteilungsleiter, Autoren, ...).

- Unterstützung des Betriebs-Workflows.
- Zugriff über einen Webbrowser.
- SQL-Schnittstellen.
- XML.
- Unterstützung „fremder“ Seiten.
- Alarmierungsfunktionen für die Benutzer- und Aufgabenverwaltung.

Bei einer Festigung des Marktes für WCM-Lösungen wird zukünftig eine Punktwertung die obigen Angaben enthalten müssen.

# 8. Zusammenfassung und Ausblick

Wie in den vorherigen Kapiteln angesprochen wurde, ist die Entwicklung von WCM-Lösungen ein neues Produktgebiet. Zwar werden auf vorhandene und bewährte Techniken wie der des Dokumenten-Managements zurückgegriffen, jedoch bedeutet gerade die zielgerichtete Kombination dieser Techniken eine Herausforderung für die Entwicklung von spezialisierten Lösungen.
In den ersten Kapiteln dieser Arbeit wurden die Grundlagen für die Entwicklung einer WCM-Lösung aufgezeigt. Mit der anschließenden Definition eines Lasten- und Pflichtenheftes wurde zusätzlich ein Ansatz für den Aufbau einer allgemeinen Lösung gegeben, die unabhängig von der zu dieser Arbeit erstellten Variante einzusetzen wäre.

Bis es zu einer Konsolidierung von Standards bei der Produktpalette von Web-Content-Management-Systeme kommt, ist es leider nur schwer möglich einen Vergleich zwischen den einzelnen Systemen aufzustellen: Zu unterschiedlich sind die Techniken, zu unterschiedlich ist die verwendete Architektur der Plattformen. Grundlegend lassen sich daher nur die in den Kapiteln 1 und 2 beschriebenen Komponenten angeben.

Da sich jedoch bereits einige kommerzielle Anbieter mit ihren Lösungen innerhalb des betroffenen Marktsegmentes zu etablieren beginnen, ist eine Verbesserung der Marktsituation in Aussicht.
Zudem ist mit der wachsenden Bedeutung von XML eine Festigung von Standards für das Dokumenten-Management zu erwarten, was ebenfalls starken Einfluß auf WCM-Lösungen hat.

Neben der Etablierung von Standards bietet das Gebiet des Web-Content-Management noch mannigfaltige Möglichkeiten für den zukünftigen Ausbau. Folgende Punkte sind besonders bemerkenswert:

- Kombination von WCM- mit Help-desk-Lösungen. WCM-Lösungen bieten das Publishing von Wissen in Form von Internetseiten. Bei Help-Desk-Lösungen liegt der Schwerpunkt darin, Wissen zu archivieren und zu verwalten. Eine Kombination dieser beiden Produktgruppen könnte innovative Anwendungen für die Wissenssuche über das WWW bewirken.
- Unterstützung anderer Internet-Dienste.
- Kombination von WCM-Lösungen mit Anwendungen des Site-Managements.
- Automatischer Eintrag neuer Seiten in Suchmaschinen.

- Direkte Suche nach Inhalten: Durch die Funktion der inversen Seitenerstellung ist es möglich, den Inhalt von Seiten zu ermitteln. Mit Hilfe eines genügend großen Satzes an Standard-Layouts, wie sie zum Beispiel bei HTML-Editoren beigelegt sind, wäre es „Wissens-Suchmaschinen“ möglich, nur auf die Teile der Seite zuzugreifen, die wirklich gesucht werden.
- Bewertung von Seiten: Ebenso wie beim obigen Punkt kann über die inverse Seitenerstellung und einen Satz an Layouts nicht nur „Wissen“ gefunden werden, sondern es ist auch möglich, eine Wertung für Webseiten abzugeben: Sind spezielle Meta-Tags vorhanden? Wie sieht das Layout aus? Entspricht das Layout dem Corporate-Design eines Unternehmens? Gerade für die nachträgliche Kontrolle von Webseiten, die von mehreren Benutzern mit unterschiedlichen Werkzeugen erstellt und bearbeitet werden, könnte dieser Punkt wertvoll sein.

Der Ausbau von WCM-Lösungen tangiert themenbedingt die Entwicklung größerer Websites. Die Frage hieraus ist, inwieweit zukünftige Entwicklungen von WCM-Lösungen auch Eingang in kommende Standards des W3C’s nehmen werden. Das diese Möglichkeit nicht von der Hand zu weisen ist, zeigt der ICE-Standard zum Austausch von Content.

So sollte nicht außer Acht gelassen werden, daß erfolgreiche WCM-Lösungen auf Webserver und Webbrowser aufbauen, die weit verbreitet sind. Zwar gibt es dabei Ausnahmen, doch zeigen die Statistiken ([Web2] und [Web3]), daß diese Angebote es in Zukunft schwer haben werden.

Fazit: Die Entwicklung von Lösungen zum Web-Content-Management wird auch in Zukunft ein Gebiet sein, bei dem auf eine breite Basis an Grundlagen zurückgegriffen werden muß. Unterschiedliche Konzepte und ein unterschiedlicher Funktionsumfang bei der Realisierung, werden zwar auch in Zukunft aus Wettbewerbsgründen vorhanden bleiben, jedoch werden sich diese Unterschiede und Konzepte auf einheitlichere Standards zurückführen lassen.

# A1. Literatur- und Webverzeichnis

## A1.1. Buch- und Arbeitsliteratur

[Lit1] „Web Content Management – Einführung, Konzepte, Software“, Oliver Zschau, Diplomarbeit an der University of Applied Sciences (FH), Fachbereich Medien und Elektrotechnik, Januar 2000

[Lit2] „Das Internet: Weltweit vernetzt“, IWT Verlag, Peter Klau, ISBN 3-88322-467-7

[Lit3] „Internet in a Nutshell“, O’Reilly, Valerie Querica, 1998, ISBN 3-89721-113-0

[Lit4] „Webmaster in a Nutshell“, O’Reilly, Spainhour & Querica, 1997, ISBN 3-930673-55-X

[Lit5] “Die wissenschaftliche Methode”, W. Wiese, 1998, http://home.bereich.com/xwolf/doc/wissmeth.doc

[Lit6] „Contentmanagement für das Intranet des Klinikums der Universität Erlangen-Nürnberg“, Stephan Raabe, Studienarbeit, Lehrstuhl Betriebssysteme

[Lit7] „Web Professional“, Rainer Kolbeck, Interest Verlag, 1998, ISBN 3-8245-0402-2

[Lit8] „JavaScript – 7-Tage-Crashkurs“, Arman Danesh, Markt&Technik, 1996, ISBN 3-8272-5176-1

[Lit9] „Lehrbuch der Software-Technik“, Spektrum Akademischer Verlag, Balzert, 1998, ISBN 3-8274-0065-1

[Lit10] Vorlesungsmaterial: „Software-Engineering“, 1998, Peter Wilke, LS Programmiersprachen, Univ. Erlangen-Nürnberg

## A1.2. Informationsseiten im WWW

[Web1] W3C, http://www.w3.org

[Web2] Zugriffsstatistik der Universität Erlangen-Nürnberg, http://www.uni-erlangen.de/webadm/stats/fau/

[Web3] Netcraft Web Server Survey (05/2000) , http://serverwatch.internet.com

[Web4] Tim Berners-Lee, http://www.w3.org/People/Berners-Lee/

[Web5] W3C: ICE-Standard, http://www.w3.org/TR/NOTE-ice

[Web6] iSyndicate – The Content Marketplace, http://www.isyndication.com

[Web7] Microsoft Frontpage 2000, http://www.microsoft.com

[Web8] Allaire Homesite 4.5, http://www.allaire.com

[Web9] Gauss Interprise AG, *http://www.gauss-interprise.com*

[Web10] New Media Sales, *http://www.newmediasales.com*

[Web11] CoCo Software Engineering, *http://www.coco.co.at*

[Web12] LAN Times, *http://www.lantimes.com*

[Web13] Harvest Suchmaschine, *http://harvest.rrze.uni-erlangen.de/*

[Web14] XML, *http://www.w3.org/TR/REC-xml*

[Web15] Microsoft, *http://www.microsoft.de*

[Web16] Altavista, *http://www.altavista.com*

[Web17] Google, *http://www.google.com*

[Web18] HotBot, *http://www.hotbot.com*

[Web19] Apache Webserver, *http://www.apache.de*

[Web20] Netscape Enterprise Webserver, *http://www.netscape.com*

[Web21] Netcraft Net Survey, *http://www.netcraft.com/survey*

[Web22] Lotus Domino, *http://domino.lotus.com*

[Web23] EDM Domis, *http://www.edm.de*

[Web24] Roxen, *http://www.roxen.com*

[Web25] TeamOne SelfHTML, *http://www.teamone.de/selfhtml*

[Web26] CGI-Tutorial, *http://cgi.xwolf.com/faq/cgitutor.shtml*

[Web27] SSI-Beispiele und Erklärungen, *http://cgi.xwolf.com/faq/ssi-sample1.shtml*

[Web28] PHP-Projekt, *http://www.php.net*

[Web29] PHP-Center, *http://www.phpcenter.de*

[Web30] ASP-Forum, *http://www.asp-forum.de*

[Web31] XWolf CGI & Webworking: Umfragen, *http://www.xwolf.com/cgi-bin/xpoll/xpoll.pl?viewpolls=1*

[Web32] CPAN, *http://www.cpan.org*

[Web33] Schema GmbH, *http://www.schema.de*

## A1.3. Artikel aus Fachzeitschriften

[Art1] „Web-Inhalte mühelos betreuen“, Oliver Zschau, NetworkWorld, Ausgabe ½ vom 21. Januar 2000

[Art2] „Web-Content-Management zerschlägt gordischen Kommunikationsknoten“, Robert Gammel, Computerwoche (CeBIT-Ausgabe), Februar 2000.

[Art3] „Content Management“, Anja Weinstein, Internet Professionell, Juli 2000

[Art4] "Content-Management-Systeme", H. Behme, Screen Business Online, Februar 1999

[Art5] „Dasein oder Nicht-Dasein - Analyse der Ausfallzeiten von Web-Servern", Jürgen Schmidt, c't, Ausgabe 8/2000

[Art6] „Die Webstruktur des RRZE", Wolfgang Wiese, Benutzerinformation RRZE 63, April 2000.

[Art7] „Studie. Websites vergraulen Surfer", c't Bericht zur Studie des Forschungsschwerpunkts Kommunikation an der FH Düsseldorf, Juli 2000.

[Art8] „Interaktive Mechanismen für Webseiten", Wolfgang Wiese, Dezember 1999, *http://www.xwolf.com/artikel/web-interaktiv.shtml*

[Art9] „Content-Management – Bewegende Inhalte", Noelani Maria Afif, InformationWeek, Ausgabe 11/2000

[Art10] „Statische HTML-Seiten – Die Arbeit auf viele Schultern verteilt", Noelani Maria Afif, InformationWeek, Ausgabe 11/2000

[Art11] „Buchtips und Leseempfehlungen" zur Programmierung mit Perl sowie für CGI, Wolfgang Wiese, Juni 2000, *http://www.xwolf.com/artikel/buchtips.shtml*

## A1.4. Weitere Quellen

Weitere Informationen, die teilweise in die Arbeit eingingen, kamen aus Presse- und Produkt- und Werbeinformationen zu WCM-Lösungen von den folgenden Firmen:

- 2CK Software, *http://www.2ck.com*
- Berens & Partner, *http://www.berensp.com*
- COIN Corporate Interactive, *http://www.coin.de*
- Dimedis, *http://www.dimedis.de*
- E-Business Technologies, *http://www.ebt.com*
- EDM Software, *http://www.edm.de*
- Feldmann, *http://www.feldmannmediagroup.com*
- Flying Dog Software, *http://www.powerslave.de*
- Gauss Interprise, *http://www.gauss-interprise.com*
- i/o Publishing Group, *http://www.iconmedialab.no*
- Imperia Software Solutions, *http://www.imperia.net*
- Info Office, *http://www.infooffice.de*
- Infopark Online Service, *http://www.infooffice.de*

- Innovation Gate, *http://www.innovationgate.de*
- Klahold, *http://www.interred.de*
- Media Artists, *http://www.media-artists.de*
- MKS, *http://www.mks.com*
- Netestate, *http://www.netestate.de*
- OMS Open Mind Systems, *http://www.openmindsystems.com*
- Pansite, *http://www.pansite.de*
- PC Ware Information Technologies, *http://www.pc-ware.de*
- Pironet, *http://www.pironet.de*
- Schema, *http://www.schema.de*
- Sicom Informatik, *http://www.sicom.de*
- Site Park, *http://www.sitepark.de*
- Six Offene Systeme, *http://www.six.de*
- Webfair, *http://www.webfair.com*

## A2. Abkürzungsverzeichnis

ARPANET - Advanced Research Project Agency Network
CGI - Common Gateway Interface
CMS - Content-Management-System
CSS - Cascading Style Sheets
DHTML - Dynamic Hypertext Markup Language
FAU - Friedrich-Alexander-Universität (Erlangen-Nürnberg)
GUI - Graphical User Interface / Grafische Benutzeroberfläche
HTML - Hypertext Markup Language
HTTP - Hypertext Transfer Protocol
ICE - Information and Content Exchange
JVM - Java Virtual Machine
KM - Knowledge Management
OOP - Objektorientierte Programmierung
PHP - Hypertext Preprocessor
PSE - Publishing-, Staging- und Extract-Konzept
RRZE - Regionales Rechenzentrum Erlangen
SSI - Server-Side Includes
VRML - Virtual Reality Modelling Language
W3C - World Wide Web Consortium
WWW - World Wide Web
WCM - Web-Content-Management
WCMS - Web-Content-Management-System
WYSIWYG - „What you see is what you get“. (Kennzeichnung für grafische Editoren.)
XML - Extensible Markup Language

# A3. Anbieter von WCMS auf der CeBIT 2000

| Produkt | Hersteller | Halle | Stand |
|---|---|---|---|
| STORYSERVER | Vignette | 1 | 7K4 |
| Internews | Media Artists | 2 | D02 |
| In4meta | PC-Ware | 2 | D02/Nr. 9 |
| Roxen | Roxen | 2 | B54 |
| InfoOffice RedDot | InfoOffice | 3 | C55 |
| Web Integrity | MKS | 3 | A45 |
| Poet CMS | Poet | 3 | D03 |
| @it | Dimedis | 4 | A04 |
| ActiveWeb | MIS-IT | 4 | C04 |
| Authoring Server Suite | NetObjects | 4 | B12 |
| Core Media | Higher Order | 4 | A04 |
| MLC Webstar | Deutsche Post Com | 4 | A16 |
| Bladerunner | Interleaf | 6 | A48 |
| Community Engine | WebFair | 6 | D32 |
| EDM Domis | EDM | 6 | C55 |
| New media system | Feldmann | 6 | H20/L2 |
| OpenCMS | mindfact interaktive medien | 6 | F07 |
| Imperia | Imperia | 6 | A34 |
| COIN | Interact | 6 | C20 |
| IRacer | HERRLICH & RAMUSCHKAT | 6 | 836-837 |
| NPS | Infopark | 6 | F48 |
| Pansite | Pansite | 6 | A34 |
| Pirobase | Pironet | 6 | E05 |
| VIP’ContentManager | Gauss Interprise | 6 | C48 |
| Six CMS | Six | 10 | 425 |
| InfoSite | Sitepark | 11 | ENAC-392 |
| InSite | InSite | 16 | B54 |
| Communique | Day Interactive | 16 | D45 |

| | | | |
|---|---|---|---|
| Publicity | Altavier | 16 | D23 |
| NetFicient | Deutsche Bank/Lotus | 18 | B08 |

## A4. Hinweise

Die Informationen in dieser Publikation werden ohne Rücksicht auf einen eventuellen Marken- und Patentschutz veröffentlicht. Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Fast alle Hardware- und Softwarebezeichnungen, die in dieser Publikation genannt werden, sind gleichzeitig eingetragene Warenzeichen und sollten als solche betrachtet werden.

Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Weder der Autor, noch die Betreuer oder der beteiligte Lehrstuhl können für fehlerhafte Angaben und deren Folgen eine juristische Verantwortung noch irgendwelche Haftung übernehmen.

# B1. Index


***4***

4-Augen-Konzept · 14, 22

***A***

ActiveX · 36, 37, 41
Administration · 15, 16, 19, 30, 44, 67
Alarmierung · 27, 51, 56, 59, 60, 64, 65
Apache · 10, 18, 29, 30, 31, 39, 47, 50, 54, 58, 61, 84
Applet · 35
Application Layer · 29
Applikationen · 27, 28, 29, 30, 39, 40, 44
Archivierung · 15, 16
ARPANET · 9, 87
ASP · 28, 36, 38, 41, 84
Assetmanagement · 21
Assets · *Siehe* Objekte
Assistenten · 11, 49
Aufgaben-Management · 50, 51, 56, 64, 65
Auslastung · 31
Autor · 7, 11, 14, 17, 22, 25, 26, 27, 45, 65, 90

***B***

Backups · 45
**Benutzer- und Zugriffsverwaltung** · 21, 24, 48
Benutzerkennung · 58, 59
Benutzerverwaltung · 21, 50, 51
Betriebssystem · 31, 34, 35, 36, 39, 50, 54
Betriebsysteme · 8

***C***

**Caching** · 44
CeBIT · 18, 84
CGI · 28, 33, 34, 35, 36, 38, 41, 50, 53, 58, 59, 84, 85, 87
CGI-Programm · 34
CM · 16, 18, 21, 23, 48
CMS · 15, 16, 17, 87
Content · 1, 7, 8, 13, 14, 15, 16, 18, 19, 22, 23, 34, 43, 47, 49, 53, 59, 69, 83, 84, 85, 87
Content Lifecycle · 15
Content-Lifecycle · 13, 18, 23, 59
Content-Management · 1, 7, 8, 15, 16, 84, 85, 87
Content-Mangement-Systemen · 8
Content-Syndication · 13
Coperate Design · 7

***D***

Datenbank · 24, 39, 43, 44, 45, 46, 48
Datenverlust · 58
Definitionsphase · 47
Design · 12, 15, 16, 18, 22, 23, 24, 37, 47, 48, 51, 53
Dokument · 9, 12, 14, 21, 22, 25
Dokumenten-Management · 16, 30, 50, 51
Dokumenten-Managements · 8, 15

Dokumententyp · 23, 24, 33, 52, 53, 57, 62, 63, 64, 69, 74
Domino-Server · 40
DTD · 33, 55, 73

***E***

Entwicklungszeit · 9
Exportschnittstellen · 21, 27

***F***

Fehlermeldung · 10, 59
Formular · 33, 58, 64, 69

***G***

Gauss Interprise · 17, 18, 43, 84, 85
Grafiken · 9, 13, 32, 45
Groupwise · 22, 30, 49

***H***

**Halbwertszeit** · 26, 27
Hardware · 18, 39, 50, 90
Hash · 55
Helmut Balzert · 47
HelpDesk · 49
Henkel · 40, 44
Hierachie · 17
HTML · 7, 9, 10, 14, 16, 18, 27, 31, 32, 33, 34, 35, 36, 37, 41, 42, 43, 44, 45, 46, 49, 50, 58, 60, 85, 87
HTML-Norm · 9, 32
HTTP · 9, 87

***I***

ICE · 13, 83, 87
Implementation · 8, 9, 47
Importschnittstellen · 21
Inhalt · 10, 12, 14, 15, 16, 18, 25, 26, 32, 34, 35, 47, 48, 51, 60, 62, 67
Inhalte · 11, 12, 13, 14, 15, 16, 19, 21, 22, 23, 25, 27, 40, 41, 42, 44, 45, 62, 84, 85
Instituten · 15
Internet · 7, 9, 11, 14, 15, 18, 29, 33, 41, 54, 83, 84
Internet Explorer · 9

***J***

Java · 28, 35, 36, 39, 40, 87
JavaScript · 28, 36, 37, 38, 50, 53, 58, 83

***K***

Keywords · 23, 45, 74, 75, 76
**KM** · 16, 87
Kontrollfunktionen · 51

***L***

Lastenheft · 47
LATEX · 10
Layout · 13, 19, 23, 43, 45, 58, 75
LDAP · 22, 39
Lebensphasen · 13, 51
Lebenszeit · 26, 27, 52, 53, 59
Lifecyles · 22
Linkmanagement · 18
Link-Management · 16

Links · 9, 11, 12, 51, 52
Linktiefen · 12
Lotus Domino · 8, 29, 30, 39, 49, 84

***M***

Medienformate · 13
Microsoft · 9, 10, 14, 28, 29, 30, 36, 39, 41, 54, 83, 84
Microsoft-IIS · 36
Modul · 35, 54, 55, 56, 57, 60

***N***

Netscape · 9, 10, 29, 30, 33, 34, 36, 39, 54, 84
Netscape Navigator · 9, 54
Nexus · 10
Normfragen · 9

***O***

Objekte · 13, 21, 23, 25, 26, 49, 51, 53, 55, 59, 62

***Ö***

Öffentlichkeit · 12, 14, 35, 41

***O***

Oliver Zschau · 8, 21, 83, 84
ORACLE · 17, 39

***P***

Performance · 30, 31, 42, 43, 45
Perl · 28, 35, 39, 50, 54, 55, 58, 85
Pflichtenheft · 47, 48, 54, 56, 73
PHP · 28, 35, 36, 38, 39, 84, 87
Preis · 17, 30
Preview · 24, 34
Priorität · 53, 59, 60
**Produktdaten** · 48
**Produkteinsatz** · 47
**Produktfunktionen** · 47
**Produktklassen** · 48
**Produktleistungen** · 48
Programmierung · 8, 41, 56, 85, 87
Protokoll · 9, 10, 13
PSE-Konzept · 44, 67
Publikationen · 8, 12, 13, 15, 17
Publishing · 7, 17, 41, 42, 43, 44, 85, 87
Publizieren · 14, 15
Publizierung · 11, 12

***R***

Rechtekonzept · 51
Redakteur · 14, 22, 25, 44, 50, 64, 65
Redakteure · 14, 25, 41, 42, 44, 53
redaktionelle Arbeit · 7
**Redaktionssysteme** · 17
Referenzen · 11, 14, 15, 25, 26, 51, 53, 57
REMOTE_USER · 58
Request · 10, 43

***S***

Schnittstellen · 18, 22, 27, 28, 30, 47, 48, 49, 50, 51
Serverarchitektur · 17

Sicherheit · 11, 30
Site-Management · 14
Skalierbarkeit · 39
Software · 8, 9, 18, 19, 40, 47, 48, 50, 83, 84, 85
Software-Management · 47
Solaris · 18, 31, 39
SQL · 39
SSI · 28, 34, 35, 50, 53, 58, 84, 87
Staging · 41, 42, 44, 46, 87
StarOffice · 48
Stefan Münz · 32
Struktugramme · 47
Suchmaschinen · 27, 28, 43
Support · 29, 30
Syndication · 13, 14, 19

***T***

Tag · 32, 33
Techniken · 8, 28, 29, 31, 36, 37, 38, 47
Tim Berners-Lee · 10, 83
ToDo · 22, 51
TODO_DBI · 56, 57, 60

***Ü***

Überwachungsprogramm · 60

***U***

Universität Erlangen-Nürnberg · 1, 8, 31, 83
UNIX-Dateisystem · 22
Unternehmen · 12, 15, 18, 30, 37

***V***

Verfügbarkeit · 30, 31, 54
Version · 9, 26, 30, 33, 45, 50, 52, 54, 67
Verwaltung · 15, 16, 21, 22, 28, 43, 44, 45, 49, 50, 51, 55, 57, 59

***W***

W3C · 9, 14, 33, 83, 87
WCM-Hersteller · 18
WCM-Lösungen · 29, 39, 40, 46, 49, 85
WCMS · 7, 8, 16, 17, 18, 19, 21, 23, 25, 26, 27, 28, 29, 31, 37, 39, 40, 41, 43, 44, 45, 53, 55, 56, 58, 61, 62, 65, 87
Web · *Siehe* WWW
Webauftritts · 11, 13
Webbrowser · 9, 33, 40, 41, 43, 49, 50
Webdesign · 7, 12, 31
Webdesignern · 9, 37
Webdokumente · 9
Webmaster · 10, 11, 83
Webpublishing · 7, 10, 13, 15, 16, 19
Webseiten · 7, 9, 11, 12, 13, 16, 17, 19, 27, 29, 31, 36, 37, 42, 44, 48, 49, 53, 85
Webserver · 8, 10, 14, 17, 29, 30, 31, 33, 34, 35, 36, 39, 41, 42, 43, 44, 45, 47, 49, 50, 54, 61, 84
Website · 14, 17, 18, 27, 31, 34
Websites · 13, 15, 31, 37, 85
Windows NT · 30, 39
WORD · 16, 28, 48
Workflowkomponente · 21
Workstation · 40, 50
World Wide Web · 9, 10, 14, 32, 87

World Wide Webs · 7, 9, 11
WWW · *Siehe* World Wide Web

***X***

X500 · 22, 39
XML · 14, 21, 27, 28, 32, 33, 39, 40, 48, 50, 54, 55, 56, 73, 84, 87

***Z***

Zugriffsrechte · 14, 68
Zugriffsrechten · 50, 51
Zugriffsverwaltung · 21
Zyklus · 24, 25, 26, 65