Otto-von-Guericke-Universität Magdeburg

Thema:

Modellgetriebenes Customizing von Web-Content-Management-Systemen am Beispiel von OpenCms

Diplomarbeit

Fakultät für Informatik
Arbeitsgruppe Wirtschaftsinformatik

Themensteller:	Prof. Dr. rer. pol. habil. Hans-Knud Arndt
Betreuer:	Dipl.-Wirt.-Inform. André Zwanziger

vorgelegt von:	André Bohna andre.bohna@st.ovgu.de

Abgabetermin:	1. April 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CIM	Computation Independent Model
CLC	Content-Life-Cycle
CMA	Content-Management-Anwendung
CMS	Content-Management-System
Content i. e. S	Content im engeren Sinn
Content i. w. S	Content im weiteren Sinn
CSS	Cascading Style Sheets
DSL	Domain Specific Language
DTD	Document Type Definition
DV	Datenverarbeitung
ERP	Enterprise Resource Planning
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICE	Information-and-Content-Exchange
JDBC	Java Database Connectivity
JSP	Java Server Pages
MDA	Model Driven Architecture
MOF	Meta Object Facility
OCL	Object Constraint Language
OMG	Object Management Group
PDF	Portable Document Format
PIM	Platform Independent Model
PM	Platform Model
PSI	Platform Specific Implementation
PSM	Platform Specific Model
RSS	Really Simple Syndication
SGML	Standard General Markup Language
SOAP	Simple Object Access Protocol (ursprünglich)
UML	Unified Modeling Language
URL	Uniform Resource Locator
VFS	Virtual File System
VLBA	Very Large Business Applications
W3C	World Wirde Web Consortium
WCMA	Web-Content-Management-Anwendung

WCMS	Web-Content-Management-System
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema Definition

Kapitel 1

Einleitung

1.1 Motivation

Wenn Standardsoftware in einer Organisation eingeführt werden soll, muss diese an die organisationsspezifischen Anforderungen angepasst werden (Heinrich et al. (2004), S. 625). Dem Auswahlprozess der einzusetzenden Standardsoftware kommt dabei eine besondere Wichtigkeit zu, da die Entscheidung für ein Standardsoftware-Produkt „nur mit erheblichem Aufwand und hohen Kosten wieder rückgängig gemacht werden kann“ (Alpar et al. (2005), S. 355). Für diesen Auswahlprozess sind die fachlichen Anforderungen zu bestimmen, die durch das Softwaresystem realisiert werden sollen (Strahringer (2009)).

Die Wirtschaftsinformatik bietet mit den Informationsmodellen ein etabliertes Instrument (Sarshar et al. (2006), S. 120), welches die Ableitung eines Kriterienkatalogs für den Auswahlprozess ermöglicht (Becker und Pfeiffer (2007), S. 3). Die innerhalb von Informationsmodellen erfassten fachlichen Anforderungen erfüllen derzeit aber hauptsächlich Beschreibungsaufgaben (Becker und Pfeiffer (2007), S. 4).

Mit der modellgetriebenen Softwareentwicklung existiert ein Ansatz, der die stärkere Einbindung und Nutzung von Modellen in den Software-

entwicklungsprozess verfolgt (Gruhn et al. (2006), S. 20; Stahl et al. (2007), S. 3).

Die Nutzung der Informationsmodelle innerhalb einer solchen modellgetriebenen Softwareentwicklung und damit einhergehend die Ausdehnung des Aufgabenbereichs von der Beschreibung auf die Gestaltung von Softwaresystemen birgt Potential zur Steigerung der Effizienz im Softwareentwicklungsprozess (Becker und Pfeiffer (2007), S. 4).

1.2 Ziel

Allgemein können Untersuchungen in der Wirtschaftsinformatik hinsichtlich der Zielsetzung ein Erkenntnisziel und/oder ein Gestaltungsziel verfolgen (Becker (1995), S. 133). Weiter kann dabei ein methodischer oder inhaltlich-funktionaler Auftrag wahrgenommen werden (Becker (1995), S. 133).

Ziel der Arbeit ist die Entwicklung eines Metamodells, mit dem die fachlichen Anforderungen an eine Web-Content-Management-Anwendung (WCMA) formal spezifiziert werden können. Dieses Metamodell soll die allgemeinen Grundmodule eines Web-Content-Management-System (WCMS) beachten und dadurch die Ausgangsbasis für das modellgetriebene Customizing eines WCMS bilden. Modelle, welche unter Nutzung dieses Metamodells erstellt wurden, sollen (teil-) automatisch in Artefakte für das Customizing eines konkreten WCMS-Produkts transformiert werden können. Die technische Umsetzung der Grundmodule wird am Beispiel eines konkreten Web-Content-Management-System aufgezeigt und auf die Möglichkeiten des Customizing eingegangen.

Für die Erstellung des Metamodells wird ein Profil der Unified Modeling Language (UML) definiert. Die Nutzung des Metamodells sowie die weitere Vorgehensweise wird anhand des Softwareentwicklungsprozess aufgezeigt und innerhalb der Model Driven Architecture (MDA) eingeordnet. Anhand eines Modellierungsbeispiels wird die Anwendung des Metamodells vorgestellt und die Praktikabilität verifiziert.

Diese Arbeit verfolgt mit der Erstellung eines Metamodells für WCMA ein Gestaltungsziel und nimmt durch die Erklärung der Vorgehensweise einen methodischen Auftrag wahr.

1.3 Aufbau der Arbeit

Der Hauptteil der Arbeit ist in folgende drei Kapitel unterteilt: Theoretische Grundlagen, Technische Grundlagen und Modellgetriebenes Customizing von Web-Content-Management-Systemen.

Im Kapitel 2 (Theoretische Grundlagen) wird zunächst der Modell-Begriff allgemein betrachtet. Im Anschluss wird auf die Nutzung von Modellen innerhalb der Softwareentwicklung eingegangen, bevor das Modellverständnis innerhalb der Wirtschaftsinformatik aufgezeigt wird. Im darauf folgenden Abschnitt werden die Begrifflichkeiten im Umfeld von Content-Management-Systemen geklärt. Anschließend wird mit den Web-Content-Management-Systemen die für diese Arbeit relevante Ausprägung der Content-Management-Systeme betrachtet. Es werden die Grundmodule identifiziert und Möglichkeiten des Customizing genannt.

In dem Kapitel 3 (Technische Grundlagen) wird zunächst auf die grundlegenden Standards für die Entwicklung von Web-Anwendungen eingegangen. Anschließend werden die Besonderheiten bei der Modellierung von Web-Anwendungen betrachtet. Im darauf folgenden Abschnitt werden die Ziele und Vorgehensweisen der modellgetriebenen Softwareentwicklung allgemein und innerhalb des Standardisierungsansatzes der Object Management Group, der Model Driven Architecture, aufgezeigt. Mit den Profilen der Unified Modeling Language wird ein Mechanismus für die Erstellung eines Metamodells vorgestellt. Um die allgemeinen Grundmodule eines WCMS anhand einer konkreten Ausprägung aufzuzeigen, wird die technische Umsetzung dieser Grundmodule am Beispiel des Web-Content-Management-System OpenCms beschrieben.

Innerhalb des Kapitel 4 wird das entwickelte Metamodell vorgestellt und die Vorgehensweise der Nutzung des Metamodells innerhalb des Softwareentwicklungsprozess aufgezeigt. Es wird darauf eingegangen, welche Informationen ein OpenCms-spezifisches Modell aufweisen muss, um für die Transfor-

mation in Artefakte für das Customizing geeignet zu sein. Im Anschluss daran wird ausschnittsweise ein Anwendungsbeispiel des Metamodells anhand einer existierenden Web-Content-Management-Anwendung vorgestellt.

Die Arbeit schließt in Kapitel 5 mit der Zusammenfassung der erzielten Ergebnisse und einem Ausblick auf weitere Schritte, die unternommen werden können, um das Ziel des modellgetriebenen Customizing eines Web-Content-Management-System zu ermöglichen.

Kapitel 2

Theoretische Grundlagen

2.1 Modelle und Modellierung

Der Begriff des Modells ist mehrdeutig und wird von verschiedenen Wissenschaftlern selbst innerhalb der selben Disziplin teils unterschiedlich ausgelegt (Fieser und Dowden (2008); Strahringer (1998), S. 2; Thomas (2005), S. 3ff). Der allgemeine Begriff Modell besitzt bereits vier Bedeutungen. Gemein haben alle Bedeutungen, dass es sich bei einem Modell um eine Abstraktion von einem bestimmten Betrachtungsgegenstand handelt. Unter Abstraktion wird dabei das Weglassen von bestimmten, für den Abstraktionszweck nicht relevanten Merkmalen, verstanden. Somit unterscheidet sich das Modell z. B. in Form oder Größe von dem betrachteten Gegenstand.

2.1.1 Allgemeine Modelltheorie

Eine Ausgangsbasis für die Betrachtung von Modellen bildet in der Literatur oftmals (Ludewig und Lichter (2006), S. 5; Thomas (2005), S. 8; vom Brocke (2003), S. 9) Stachowiaks Werk über die „Allgemeine Modelltheorie“. Stachowiak (1973) identifiziert darin drei Hauptmerkmale, die ein Modell ausmachen: das Abbildungsmerkmal, das Verkürzungsmerkmal und das pragmatische Merkmal (Stachowiak (1973), S. 131f). Das *Abbildungsmerkmal* sagt aus, dass ein Modell ein natürliches oder künstliches Original (Modell-

gegenstand) abbildet. Den Begriff des *Originals* definiert Stachowiak (1973) allgemein als „auf natürliche Weise entstanden, technisch hergestellt oder sonstwie gegeben“ (Stachowiak (1973), S. 131). Das *Verkürzungsmerkmal* sagt aus, dass ein Modell „im allgemeinen nicht alle Attribute des durch das Modell repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant erscheinen“ (Stachowiak (1973), S. 132), erfasst. Dem *pragmatischen Merkmal* ist zu entnehmen, dass ein Modell einem Original „nicht per se eindeutig zugeordnet [ist]“ (Stachowiak (1973), S. 132), sondern von einem bestimmten Subjekt (Modellnutzer) „innerhalb bestimmter Zeitintervalle und [..] unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen“ (Stachowiak (1973), S. 133) als Ersatz des Originals dient.

Die zur Erstellung eines Modells genutzten Originale sind dabei selbst schon modellmäßige Konstrukte oder „Vor-Modelle“, deren Wirklichkeitsbezug durch bestimmte Absichten, Zwecke und Ziele von bestimmten Subjekten (Modellersteller) bestimmt ist (Stachowiak (1973), S. 286ff).

Ein *Modell* kann somit allgemein definiert werden, als eine von einem bestimmten Subjekt (*Modellersteller*) vorgenommene, vereinfachte Abbildung eines (inter-) subjektiven Originals (*Modellgegenstand*), welches wiederum ein Modell sein kann. Das so entstandene Abbild kann durch ein bestimmtes Subjekt (*Modellnutzer*) in einem bestimmten Zeitintervall und zu einem bestimmten Zweck (*Modellzweck*) genutzt werden.

Nach dieser allgemeinen Definition des Modellbegriffs werden als Nächstes Abgrenzungs- und Klassifikationsmöglichkeiten von Modellen vorgestellt. Da hiervon in der Literatur eine Vielzahl existiert (Strahringer (1998), S. 2; Thomas (2005), S. 3), werden nur die in der weiteren Arbeit relevanten aufgezeigt.

Vom Entstehungszeitpunkt können Modelle in ein kausales Abhängigkeitsverhältnis gesetzt werden. Ein Modell kann zum einen dazu dienen, ein bereits existierendes Original zu beschreiben (*deskriptiv*), und zum anderen als Vorbild für einen noch zu erzeugenden Modellgegenstand genutzt werden

(*präskriptiv*) (Hesse und Mayr (2008), S. 381; vom Brocke (2003), S. 10). Wenn ein Modell (deskriptiv) aus einem Original erzeugt wurde und dieses (präskriptiv) zur Konstruktion eines neuen Modellgegenstandes genutzt wird, so wird von einem *transienten Modell* gesprochen (Hesse und Mayr (2008), S. 381).

In Abhängigkeit von der Einbeziehung der Dimension der Zeit können Modelle in Statikmodelle und Dynamikmodelle unterschieden werden. *Statikmodelle* beschreiben eine „beobachtbare oder beobachtbar gedachte Konstellation von Gegenständen, Beziehungen und sonstigen beschreibenden Elementen“ (Hesse und Mayr (2008), S. 382) zu einem bestimmten Zeitpunkt. *Dynamikmodelle* hingegen bilden Vorgänge ab, betrachten somit die Veränderung an Gegenständen, Beziehungen oder sonstiger Elemente über die Zeit (Hesse und Mayr (2008), S. 382).

2.1.2 Modelle in der Softwareentwicklung

Die Phasen der Software-Entwicklung gehören neben den Phasen der Software-Anwendung und Software-Wartung zum Software-Lebenszyklus, dem Prozess der Entwicklung von Software-Produkten (Dumke (2003), S. 18). Unter einer *Phase des Software-Lebenszyklus* wird ein zeitlich begrenzter Abschnitt mit relativ eigenständigen Ressourcen sowie einem Anfangs- und Endzustand verstanden (Dumke (2003), S. 18). Zu den Phasen der Software-Entwicklung gehören, wie in Abbildung 2.1 dargestellt, die Problemdefinition, die Anforderungsanalyse und Spezifikation, der Entwurf, die Implementation, der Test sowie Betrieb und Wartung (Dumke (2003), S. 18f; Ludewig und Lichter (2006), S. 335ff; Heinrich et al. (2004), S. 607; Kleppe et al. (2003), S. 3).

Modelle werden dabei, wie in Abbildung 2.1 durch die Ergebnisse der Phasen dargestellt ist, während der ersten drei Phasen und somit vor der eigentlichen Implementierung erstellt (Kleppe et al. (2003), S. 3). Es handelt sich

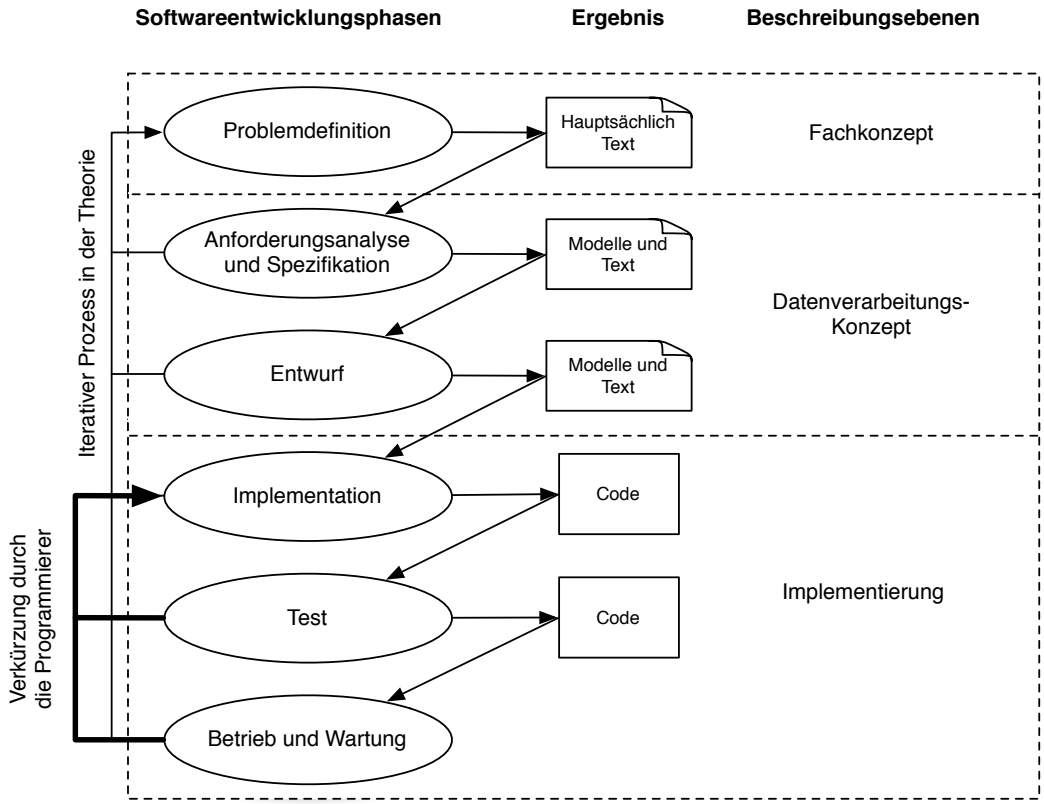

Abbildung 2.1: Modelle innerhalb des Softwareentwicklungsprozesses (Kleppe et al. (2003), S. 3; Scheer (2002), S. 7)

hierbei um präskriptive Modelle, welche das zu realisierende Softwaresystem unter verschiedenen Gesichtspunkten beschreiben. Diese besitzen keine feste Verbindung zu dem implementierten Softwaresystem (Stahl et al. (2007), S. 3). Dadurch entsteht das Problem, dass die Modelle bei Änderungen am Softwaresystem innerhalb späterer Phasen nicht aktualisiert und somit inkonsistent zu dem realisierten Softwaresystem zu werden (Stahl et al. (2007), S. 3; Gruhn et al. (2006), S. 17; Fettke und Loos (2003), S. 557). Dieser Sachverhalt ist innerhalb der Abbildung 2.1 durch die Gegenüberstellung des (mittels dünner Linien dargestellten) iterativen Prozess in der Theorie, der alle Phasen der Softwareentwicklung abdeckt, und der (mittels dicker Linien dargestellten) oft durch die Programmierer vorgenommenen Verkürzung dieses Prozesses auf die Phasen, welche den Code des Softwaresystems als Ergebnis haben, zu erkennen (Kleppe et al. (2003), S. 3). Diese Verwendung

von Modellen wird als *modellbasierte* Softwareentwicklung bezeichnet (Stahl et al. (2007), S. 3).

Die hierbei verwendeten Modelle können hinsichtlich verschiedener Sichten typisiert werden (Rautenstrauch und Schulze (2002), S. 227). Innerhalb der Beschreibungssicht werden die Betrachtungsgegenstände der Modellierung unterschieden. Mögliche Betrachtungsgegenstände sind Daten, Funktionen, Objekte, Leistungen, Organisationen und Prozesse (Rautenstrauch und Schulze (2002), S. 227; Scheer (2002), S. 36).

Hinsichtlich der Art dieser Betrachtungsgegenstände können Domänenmodelle und Systemmodelle unterschieden werden. Während *Domänenmodelle* Ausschnitte des Anwendungsbereichs darstellen, werden bei *Systemmodellen* bestehende oder zu entwickelnde Softwaresysteme betrachtet (Broy und Steinbrüggen (2004), S. vi; Frankel (2003), S. 192). Systemmodelle können darüber hinaus in logische und physische Systemmodelle unterschieden werden (Frankel (2003), S. 192f). Während in *logischen Systemmodellen* die Struktur- und Verhaltensaspekte eines Systems abstrahiert von den Elementen des betrachteten Systems beschrieben werden, sind diese in *physischen Systemmodellen* gerade Gegenstand der Betrachtung (Frankel (2003), S. 192).

Bezüglich der Nähe zur technischen Implementierung können die drei in Abbildung 2.1 im Bezug zu den Phasen der Softwareentwicklung dargestellten Beschreibungsebenen unterschieden werden. Nach der Entfernung von der Implementierung absteigend geordnet sind dies die Ebene des Fachkonzepts, die Ebene des Datenverarbeitungs-Konzepts (DV-Konzept) und die Implementierungsebene (Scheer (2002), S. 7).

Diese Ebenen spiegeln auch die Entwicklungsschritte innerhalb der Softwareentwicklung wieder. Der erste Entwicklungsschritt, der fachliche Entwurf, beinhaltet die Rekonstruktion der Fachbegriffe des zu realisierenden Softwaresystems und geht nicht auf technische Fragestellungen ein (Ortner (1997), S. 20). Die hierbei genutzten Modelle sind demzufol-

ge Domänenmodelle und befinden sich auf der Ebene des Fachkonzepts. Dem fachlichen Entwurf folgt der systemabhängige, logische Entwurf des Softwaresystems (Ortner (1997), S. 20). Die genutzten Modelle sind logische Systemmodelle der Ebene des DV-Konzepts. Auf der Basis des logischen Entwurfs erfolgt dann in den weiteren Entwicklungsschritten die Umsetzung innerhalb der physischen Systemmodelle der Implementierungsebene (Ortner (1997), S. 20).

2.1.3 Modelle in der Wirtschaftsinformatik

Als Gegenstand der Wirtschaftsinformatik werden Informationssysteme in Wirtschaft und Verwaltung angesehen (Ferstl und Sinz (2006), S. 1). Ziel ist es, Methoden und Verfahren für das Verstehen (*Erkenntnisziel*) und Gestalten (*Gestaltungsziel*) dieser Informationssysteme zu entwickeln (Becker (1995), S. 133). Hierfür werden Modelle als Hilfsmittel zur Erklärung (deskriptive Modelle) und Gestaltung (präskriptive Modelle) realer Systeme eingesetzt (Becker (1995), S. 135; Adam (1996), S. 60).

Mit dem abbildungsorientierten und dem konstruktionsorientierten Modellbegriff werden innerhalb der Wirtschaftsinformatik zwei Perspektiven des allgemeinen Modellbegriff unterschieden (vom Brocke (2003), S. 9).

Beim *abbildungsorientierten Modellbegriff* wird das Abbildungsmerkmal des allgemeinen Modellbegriffs von Stachowiak (1973, S. 131) in den Vordergrund gestellt. Die Modelle werden als „immaterielle und abstrakte Abbilder der Realität für Zwecke eines Subjekts“ interpretiert (vom Brocke (2003), S. 10). Im Gegensatz zum Abbildungsmerkmal von Stachowiak (1973) wird hier somit explizit ein Realitätsbezug unterstellt (Thomas (2005), S. 14). Allerdings wird der abbildungsorientierte Modellbegriff, der die objektive Wahrnehmung der Realität (vom Brocke (2003), S. 11) sowie die Existenz von Strukturen in der Realität voraussetzt (Thomas (2005), S. 21), in der Literatur kritisch betrachtet.

Beim *konstruktionsorientierten Modellbegriff* wird gegenüber dem abbildungsorientierten Modellbegriff nicht von einer objektiven, sondern von einer subjektgebundenen Wahrnehmung der Realität ausgegangen. Außerdem wird statt eines realen Systems eine Problemdefinition als Gegenstand der Modellierung betrachtet (Thomas (2005), S. 17f). In der Literatur wird dabei die Definition von Schütte (1998, S. 59) anerkannt, der ein Modell als Konstruktionsergebnis eines Modellierers definiert, der mithilfe einer Sprache eine zu einer Zeit relevante Repräsentation eines Originals deklariert (Schütte (1998), S. 59).

Die Informationsmodelle sind ein in der Wirtschaftsinformatik etabliertes Instrument (Becker und Knackstedt (2002), S. 9; Sarshar et al. (2006), S. 120) zur Beschreibung der Struktur und Beziehungen von Informationssystemen. Damit stellen sie den Informationssystem-relevanten Teil des Unternehmensmodells dar (Loos und Scheer (1995), S. 185). Informationsmodelle können anhand der in Abschnitt 2.1.2 vorgestellten Ebenen (Fachkonzept, DV-Konzept, Implementierungsebene) in konzeptionelle, logische und physische Informationsmodelle unterschieden werden (Schmidt (1999), S. 50). Die konzeptionellen Informationsmodelle erfüllen derzeit hauptsächlich Beschreibungsaufgaben. Zur Gestaltung und Prognose werden sie nur selten genutzt (Becker und Pfeiffer (2007), S. 4). Die konzeptionellen Informationsmodelle haben einen Diskursbereich der Realwelt zum Gegenstand, liefern allerdings durch den sozialen Prozess, geprägt „durch die Intentionen und Anschauungen der beteiligten Personen [...] keine exakte Abbildung einer (potentiellen) Realität“ (Becker und Pfeiffer (2007), S. 8). Die Modelle des DV-Konzepts überführen diese konzeptionellen Informationsmodelle in eine alternative, formale Darstellungsform (Becker und Pfeiffer (2007), S. 8). Im Hinblick auf die Softwareentwicklung könnte durch die Wahrnehmung der Gestaltungsfunktion der konzeptionellen Informationsmodelle die Effizienz der derzeit größtenteils manuellen Transformation von Modellen der Ebene des Fachkonzepts in Modelle der Ebene des DV-Konzepts und der Implementierung durch eine (Teil-) Automatisierung erhöht werden (Becker und Pfeiffer (2007), S. 4).

2.2 Content-Management-Systeme

Die Bezeichnung Content-Management-System (CMS) kann als Sammelbegriff für verschiedene funktionale Ansätze genutzt werden, welche die Gemeinsamkeit aufweisen, dass sie dem zielgerichteten, systematischen Umgang bei der Erstellung, Verwaltung und Bereitstellung von Inhalten (Content) dienen (Milleg und Wagner (2001), S. 38). Welche Softwaresysteme unter den Begriff Content-Management-System fallen, ist in der Literatur so unterschiedlich definiert wie der Begriff Content selbst. Laut Milleg und Wagner (2001, S. 38) zählen Redaktionssysteme, Dokumenten-Management-Systeme, Web-Administration/Content Delivery, Portale und das Web-Applikationsmanagement zu den CMS. Kampffmeyer (2003) wiederum grenzt die CMS stärker ein auf das Web-Content-Management, Content-Syndication, Digital- oder Media-Asset-Management sowie das Enterprise-Content-Management (Kampffmeyer (2003), S. 88). Seine Argumentation hierfür besteht in der grundlegenden Verschiedenheit von Dokumenten und Content (Kampffmeyer (2003), S. 88).

Um eine Eingrenzung des Begriffs Content-Management-System vornehmen zu können, wird zunächst der Begriff „Content“ näher betrachtet. Anschließend wird auf den Content-Management-Prozess und den Content-Life-Cycle sowie deren Zusammenhang eingegangen. Auf dieser Basis erfolgt eine Definition des Bezeichnung Content-Management-System. Im Anschluss daran wird auf Unterscheidungsmerkmale von Content-Management-Systeme eingegangen.

2.2.1 Content

In der Literatur wird der Begriff Content als Anglizismus teilweise nur kurz als „Inhalte und Mediendaten aller Art“ (Rothfuss und Ried (2003), S. 60) oder von vielen Nutzern nachgefragte Informationen (Lohr und Deppe (2001),

S. 3) umschrieben. Allgemeiner Konsens besteht in den vielen verschiedenen Ausprägungsmöglichkeiten des Content von statischen Dokumenten über Audio- und Videoinhalte bis zu Daten, die vor dem Zugriff dynamisch aus unterschiedlichen Quellen zusammengestellt werden (Gulbins et al. (2002), S. 150). Um eine Unterscheidung der Begriffe Dokument und Content zu ermöglichen, wird zunächst der Begriff des Dokuments definiert. Im Anschluss wird auf die widersprüchlichen Angaben zu der Struktur von Content in der Literatur eingegangen und folgend der Content-Begriff konkretisiert. Darauf folgend wird auf die Bestimmung der Granularität von Content eingegangen.

Unter einem *Dokument* wird eine Menge an Daten verstanden, die als eine in sich geschlossene, authentische Einheit erstellt, verteilt und bearbeitet wird und für die Wahrnehmung durch Menschen gedacht ist (Heinrich et al. (2004), S. 201; Goik et al. (2007), S. 102; Kampffmeyer (2003), S. 12; Gulbins et al. (2002), S. 19). Diese Definition schließt dabei explizit Dokumente, die dem Datenaustausch zwischen Maschinen dienen, aus. Dokumente können sich dabei aus jeglicher Art von Daten wie Filmen, Bildern oder Text, in analoger wie digitaler Form zusammensetzen (Goik et al. (2007), S. 102). Für die weiteren Betrachtungen werden allerdings Dokumente in digitaler Form vorausgesetzt. Wie in Abbildung 2.2 dargestellt, kann ein Dokument neben dem eigentlichen Inhalt zusätzliche Daten über diese Daten (Metadaten) enthalten. Diese *Metadaten* können unterschieden werden in Angaben zum logischen Aufbau der Daten (Struktur) und Angaben zur Darstellung (Layout) (Schoop und Gersdorf (2001), S. 992; Wilhelm (2002), S. 391).

In der Literatur finden sich teils widersprüchliche Angaben darüber ob Content strukturiert (Rothfuss und Ried (2003), S. 60), unstrukturiert (Gersdorf (2002), S. 75; Stein (2000), S. 310) oder beides (Kampffmeyer (2003), S. 88; Rückel et al. (2007), S. 79) sein kann. Unter *Struktur* wird allgemein die für einen bestimmten Betrachtungszweck vorgenommene Auswahl einer Menge von Beziehungen über einer Menge von Objekten verstanden (Schneider (1998), S. 832). In Bezug auf Daten kann die logische Datenstruktur von

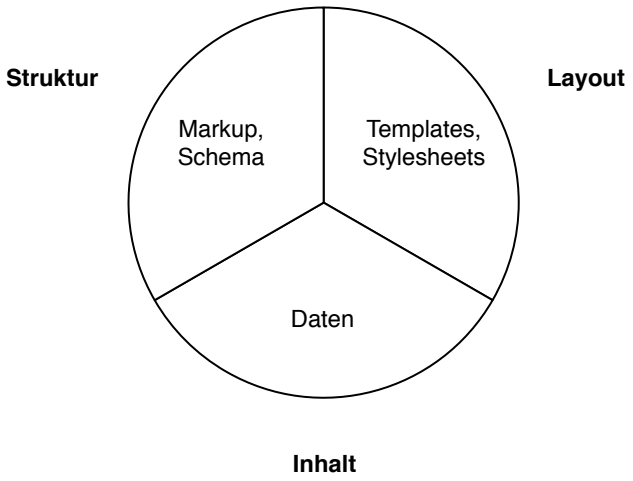

Abbildung 2.2: Trennung der Dokumentbestandteile (Schoop und Gersdorf (2001), S. 992; Wilhelm (2002), S. 392)

der physischen Datenstruktur unterschieden werden (Kretzschmar (2007), S. 211). Der hier betrachtete Strukturbegriff betrifft die *logische Datenstruktur*, welche die Zusammenhänge des Informationsbereichs abbildet (Kretzschmar (2007), S. 212). Daten werden als *strukturiert* bezeichnet, wenn sie nach einem bestimmten, explizit formulierten Schema aufgebaut sind (Großmann und Koschek (2005), S. 13). Unstrukturierte Daten sind nach keinem festen Schema aufgebaut oder legen dieses nicht offen (Großmann und Koschek (2005), S. 13). Um bereits vorliegende Daten zu strukturieren oder neue Daten durch eine Struktur zu erfassen, kann dies zum einen durch eine Fragmentierung und somit durch Angabe der inneren Struktur der Daten geschehen (Kretzschmar (2007), S. 219). Zum anderen kann eine Strukturierung durch die Einordnung der Daten in ein Ordnungssystem wie Hierarchien und Taxonomien und somit eine Assoziation mit bereits existierenden Strukturen erfolgen (Kretzschmar (2007), S. 219). Strukturinformationen können weiterhin über die Angabe von Metadaten hinzugefügt werden (Kretzschmar (2007), S. 219). Um eine Aussage über die Struktureigenschaften von Content treffen zu können, soll als nächstes der Begriff Content näher betrachtet werden.

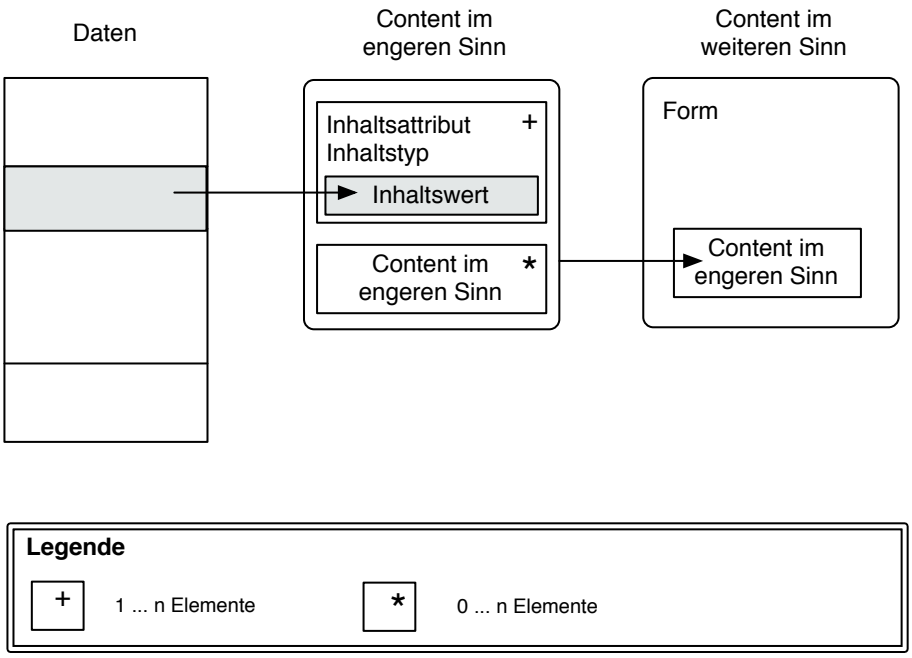

Abbildung 2.3: Content im engeren und Content im weiteren Sinn

Um den Content-Begriff zu konkretisieren unterscheidet Ehlers (2003, S. 26f) zwischen *Content im engeren Sinn* sowie *Content im weiteren Sinn* (siehe Abbildung 2.3). *Content im engeren Sinn* (Content i. e. S.) definiert Content als eine nicht-leere Menge von Inhaltstupeln und einer Menge weiterer Content-Elemente (Ehlers (2003), S. 26). Somit ermöglicht Content i. e. S. eine Strukturierung durch Fragmentierung der zu Grunde liegenden Daten, genau so wie die Bildung von Hierarchien in welche Daten eingeordnet werden können. Als *Inhaltstupel* wird ein Inhaltswert mit einem bestimmten Inhaltstyp und einem Inhaltsattribut bezeichnet (Ehlers (2003), S. 26). Unter *Inhaltsattribut* wird ein, das Inhaltstupel eindeutig identifizierender Name verstanden, der Aufschluss über den Inhaltswert gibt. Der Datentyp des Inhalts wird im *Inhaltstyp* angegeben (Ehlers (2003), S. 26). Inhaltsattribut und Inhaltstyp sind somit Metadaten des Inhaltswertes. Der Inhaltswert dient als Container für jegliche Art von Daten (Ehlers (2003), S. 26). Somit kann sich der Inhaltswert aus strukturierten sowie unstrukturierten Daten

zusammensetzen. Wird im folgenden von Content gesprochen, so ist hierbei Content im engeren Sinn gemeint.

Bei *Content im weiteren Sinn* (Content i. w. S.) wird Content im engeren Sinn um die Präsentationsform erweitert. Content im weiteren Sinn wird aufgrund der Einbeziehung der Präsentationsform und zur Abgrenzung vom allgemeinen Content-Begriff unter Nutzung des Objekt-Begriffs als *Content-Objekt* bezeichnet (Ehlers (2003), S. 27).

Content i. e. S. und somit auch Content i. w. S. erhalten durch die Metadaten eine Explikation der Struktur (Ehlers (2003), S. 29). Der Inhaltswert selbst kann dagegen in Abhängigkeit vom Inhaltstyp sowohl strukturiert als auch schwach strukturiert oder unstrukturiert sein. Der Inhaltswert kann als strukturiert bezeichnet werden, wenn die Struktur durch den Inhaltstypen beschrieben ist und die Daten keine Angaben zum Layout enthalten (z. B. formatierte Daten aus einer Datenbank) (Kampffmeyer (2003), S. 89). Als schwach strukturiert wird der Inhaltswert bezeichnet werden, wenn die Struktur nicht durch den Inhaltstyp bestimmt werden kann, da Struktur- und Layout-Angaben innerhalb der Daten zwar vorhanden, aber nicht offengelegt sind (z. B. Textverarbeitungsdateien in einem proprietären Format) (Kampffmeyer (2003), S. 89). Als unstrukturiert wird der Inhaltswert bezeichnet, falls die Struktur durch den Informationstyp nicht näher bestimmt werden kann (z. B. Audio- und Videodaten, Bilder) (Kampffmeyer (2003), S. 89). Im Gegensatz zu Dokumenten werden beim Content Daten somit „modularisiert und als strukturierte Komponenten verwaltet“ (Schoop und Gersdorf (2001), S. 994).

Von dieser Definition der Struktureigenschaften des Contents ausgehend erschliessen sich auch die Anfangs genannten unterschiedliche Angaben über die Struktur von Content. Auf der einen Seite kann Content als unstrukturiert bezeichnet werden, wenn lediglich der Inhaltswert betrachtet wird. Auf der anderen Seite wird Content durch die Angabe von Metadaten sowie die Zusammensetzung aus Content-Elementen strukturiert.

An die Feststellung der Strukturierungsmöglichkeiten der Daten innerhalb von Content schliesst sich die Fragestellung nach einer möglichst optimalen Granularität des Content an. Für die Bestimmung der Granularität lassen sich zwei Parallelen ziehen. Die erste betrifft die Speicherung von Daten in einem relationalen Datenbanksystem. Bei dem relationalen Datenmodell, welches diesen Datenbanksystemen zugrunde liegt, wird die Struktur der Daten in einem Schema so vollständig wie möglich beschrieben (Rahm und Vossen (2003), S. 3). Die Offenlegung der Struktur wird durch die mittels Algorithmen durchgeführte Methodik der Normalformen erreicht, die der Entfernung von Redundanzen dient. Rautenstrauch und Schulze (2002, S. 132ff) weisen darauf hin, dass bereits die erste Normalform in der Praxis zu „absurden Strukturen“ (Rautenstrauch und Schulze (2002), S. 133) mit Laufzeitproblemen führen kann und deshalb ein Kompromiss zu finden ist. Ein solcher Kompromiss kann durch die Ableitung der benötigten Datenstrukturen aus dem konzeptionellen Datenmodell, auf der Ebene des Fachkonzepts, erfolgen (Rautenstrauch und Schulze (2002), S. 233).

Eine weitere Parallele kann in Anlehnung an die Betrachtung der Granularität von Software-Komponenten in (Turowski (2003), S. 83f) mit folgender Begründung gezogen werden: in beiden Fällen wird die Zusammenführung von Komponenten (Inhalts-Komponenten und Software-Komponenten), die in bestimmten Abhängigkeiten zueinander stehen, zu verschiedenen neuen Ganzen betrachtet (Szyperski (1998), S. 3; Schoop und Gersdorf (2001), S. 992). Wie für die Software-Komponenten gilt somit, wie in Abbildung 2.4 dargestellt, auch für die Granularität von Content, dass eine feine Granularität zu einer leichten Wiederverwendbarkeit mit geringem Nutzen führt (Turowski (2003), S. 84). Eine grobe Granularität dagegen führt zu einer schweren Wiederverwendbarkeit mit hohem Nutzen (Turowski (2003), S. 84).

Die Bestimmung der Granularität des Content darf nicht nur von technischen sondern sollte insbesondere von fachlichen Anforderungen erfolgen (Schoop und Gersdorf (2001), S. 995). Dabei ist zu beachten, dass die Leichtigkeit

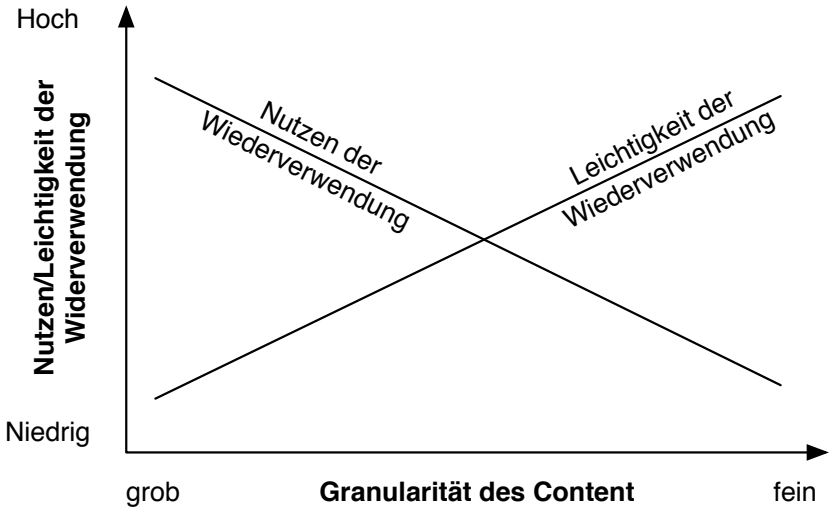

Abbildung 2.4: Abhängigkeit zwischen Nutzen/Leichtigkeit der Wiederverwendung und der Granularität von Content

und der Nutzen der Wiederverwendung in Bezug auf die Granularität des Contents gegenläufig sind (Gersdorf (2002), S. 77).

2.2.2 Content-Management

Mit Content-Management wird der gesamte Prozess der zielgerichteten, systematischen Verwaltung von Content bezeichnet (Gersdorf (2002), S. 75; Rothfuss und Ried (2003), S. 57). Dieser Prozess betrifft alle Aktivitäten von der Planung über die Erstellung respektive Beschaffung, Aufbereitung, Formatierung bis hin zur Publikation von Content (Gersdorf (2002), S. 75; Stein (2000), S. 310; Rothfuss und Ried (2003), S. 52; Kampffmeyer (2003), S. 90).

Es gibt in der Literatur zwei verschiedene Ansätze, den Begriff Management zu definieren. Dabei wird zwischen einem funktionalen[1] und einem institu-

[1] Bei dem funktionalen Managementansatz werden Handlungen, welche „der Steuerung des Leistungsprozesses“ (Steinmann und Schreyögg (2000), S. 6) dienen, betrachtet. Diese Handlungen können beispielsweise planender, organisierender oder kontrollierender Art sein (Steinmann und Schreyögg (2000), S. 6).

tionalen[2] Managementbegriff unterschieden (Weber und Kabst (2006), S. 9; Steinmann und Schreyögg (2000), S. 6). Beim Content-Management wird der funktionale Managementbegriff aufgegriffen und in der Perspektive *Verwaltung* des Content-Management in Form der Querschnittsfunktionen „Gestaltung“, „Steuerung“ und „Kontrolle“ umgesetzt (Ehlers (2003), S. 105).

Die durch das Content-Management verfolgten Ziele können nach der Art des Zielinhalts in Sachziel und Formalziel unterschieden werden (Heinrich et al. (2004), S. 738; Heinrich (2002), S. 95). Zu den *Sachzielen* können neben der Konsistenz, Aktualität und Zuverlässigkeit vor allem die leichte Erschließbarkeit der Inhalte gezählt werden. Das *Formalziel* ist die ökonomische Gestaltung des gesamten Content-Management-Prozesses (Gersdorf (2002), S. 75). Zur Erreichung der Ziele ist eine Mehrfachverwendung von Content nicht nur in Hinsicht verschiedener Ausgabemedien, sondern auch in Hinsicht verschiedener Publikationen anzustreben (Gersdorf (2002), S. 75f).

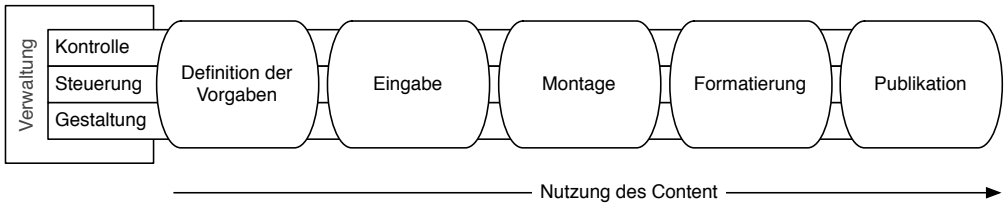

Abbildung 2.5: Idealisierter Content-Management-Prozess (Ehlers (2003), S. 106; Rothfuss und Ried (2003), S. 22; Gersdorf (2002), S. 76)

Wie in Abbildung 2.5 dargestellt, können die *Funktionen* des Content-Management in die Perspektiven *Nutzung des Contents* und *Verwaltung* unterteilt werden. Die Perspektive Nutzung des Content setzt sich aus den horizontal angeordneten Funktionen „Definition der Vorgaben“, „Eingabe“, „Montage“, „Formatierung“, „Publikation“ und „Archivierung/Löschung“ zusammen. Die Verwaltungs-Perspektive umfasst die bereits erwähnten, vertikal angeordneten Querschnittsfunktionen „Kontrolle“, „Steuerung“ und „Gestaltung“ (Gersdorf (2002), S. 77).

[2] Im Gegensatz zum *funktionalen Ansatz* wird beim *institutionellen Ansatz* „die Gruppe von Personen, die in einer Organisation mit Anweisungsbefugnissen betraut ist“ (Steinmann und Schreyögg (2000), S. 6) betrachtet.

Die Querschnittsfunktion *Gestaltung* umfasst die Definition von Strukturen und Metadaten für die einzelnen Content-Elemente sowie die Definition des Layouts spezifisch der Zielgruppen und der Ausgabemedien. Für die Gestaltung des Layouts werden so genannte *Templates* als Vorlagen für das Erscheinungsbild und die Struktur der Dokumente, die aus einzelnen Content-Elementen zusammengesetzt werden, genutzt (Stein (2000), S. 311; Karajannis und Bissel (2000), S. 66). Diese Templates können individuell für ein Dokument oder für eine Gruppe von Dokumenten angelegt sein (Stein (2000), S. 311). Die Managementfunktionen *Steuerung* und *Kontrolle* wirken prüfend und beeinflussend auf alle Funktionen der Perspektive *Nutzung des Contents*.

Bei der *Definition der Vorgaben* werden die Grundlagen für die weiteren Prozessschritte gelegt. Neben organisatorischen Aspekten wie der Berechtigungsstruktur wird hier die Struktur der Inhaltstupel und somit des Content festgelegt (Ehlers (2003), S. 106f).

Die Funktion *Eingabe* setzt die Masken, die entsprechend der festgelegten Struktur des Contents aufgebaut sein müssen, voraus. Diese werden genutzt, um sowohl den Inhaltswert selbst als auch die Metadaten anzulegen. Dabei wird unter der Funktion *Eingabe* nicht nur die erstmalige Erfassung von Content verstanden, sondern auch explizit die spätere Ergänzung, Verfeinerung und Änderung an Content (Ehlers (2003), S. 107).

Unter *Montage* wird die benutzerabhängige und benutzerunabhängige Selektion weiterer Content-Elemente verstanden Die benutzerabhängige Selektion kann zum einen aufgrund von direkten Angaben des Nutzers zu seinem Informationsbedarf erfolgen oder zum anderen indirekt durch bestimmte Informationen, die über den Nutzer gesammelt wurden und welche Aufschluss über den (potentiellen) Informationsbedarf des Nutzers geben (Ehlers (2003), S. 107). Die Montage ist somit eine vollautomatische Funktion des Content-Management.

Bei der Funktion *Formatierung* erfolgt bei elektronischen Publikationen eine automatische Konvertierung in ein für das Zielmedium geeignetes Format. Nur bei Printmedien ist aufgrund der hohen Ansprüche an eine genaue Positionierung eine Nachbearbeitung erforderlich (Ehlers (2003), S. 108).

Die *Publikation* ist die Funktion der medienspezifischen Bereitstellung der aufbereiteten und eventuell an die Nutzerbedürfnisse angepassten Content-Elemente. Hierfür werden Templates genutzt, um die einzelnen Content-Elemente zu einem Dokument zusammenzusetzen (Ehlers (2003), S. 108).

Um eine Verwendungs- und Medienneutralität des Contents zu erreichen, erfolgt im Content-Management eine explizite Trennung von Inhalt, Layout und Struktur (Gersdorf (2002), S. 75; Ehlers (2003), S. 27; Stein (2000), S. 311; Milleg und Wagner (2001), S. 37). Inhalt bezieht sich dabei auf den in 2.2.1 definierten Inhaltswert als Teil von Content. Hierdurch sowie durch die Verwaltung des Inhalts in modularen Content-Elementen ist der Content an kein Dokument eines spezifischen Ausgabemediums gebunden. Die Zusammensetzung der Content-Elemente zu einem Dokument, welches entsprechend an ein bestimmtes Ausgabemedium gebunden ist, erfolgt erst, wenn ein Informationsbedarf entsteht (Gersdorf (2002), S. 77).

2.2.3 Content-Life-Cycle

Während der Content-Management-Prozess eine idealisierte lineare Folge von Funktionen zur Erstellung einer Publikation darstellt, wird beim *Content-Life-Cycle* (CLC) der gesamte Lebenszyklus eines einzelnen Content-Elements von der Erstellung bis zur Archivierung/Löschung genauer betrachtet (Ehlers (2003), S. 108; Karajannis und Bissel (2000), S. 67).

Zwischen den Funktionen des Content-Management (mit Ausnahme der Funktion *Definition der Vorgaben*, die im Content-Life-Cycle keine Beachtung findet) und des Content-Life-Cycle besteht, wie in Abbildung 2.6 zu

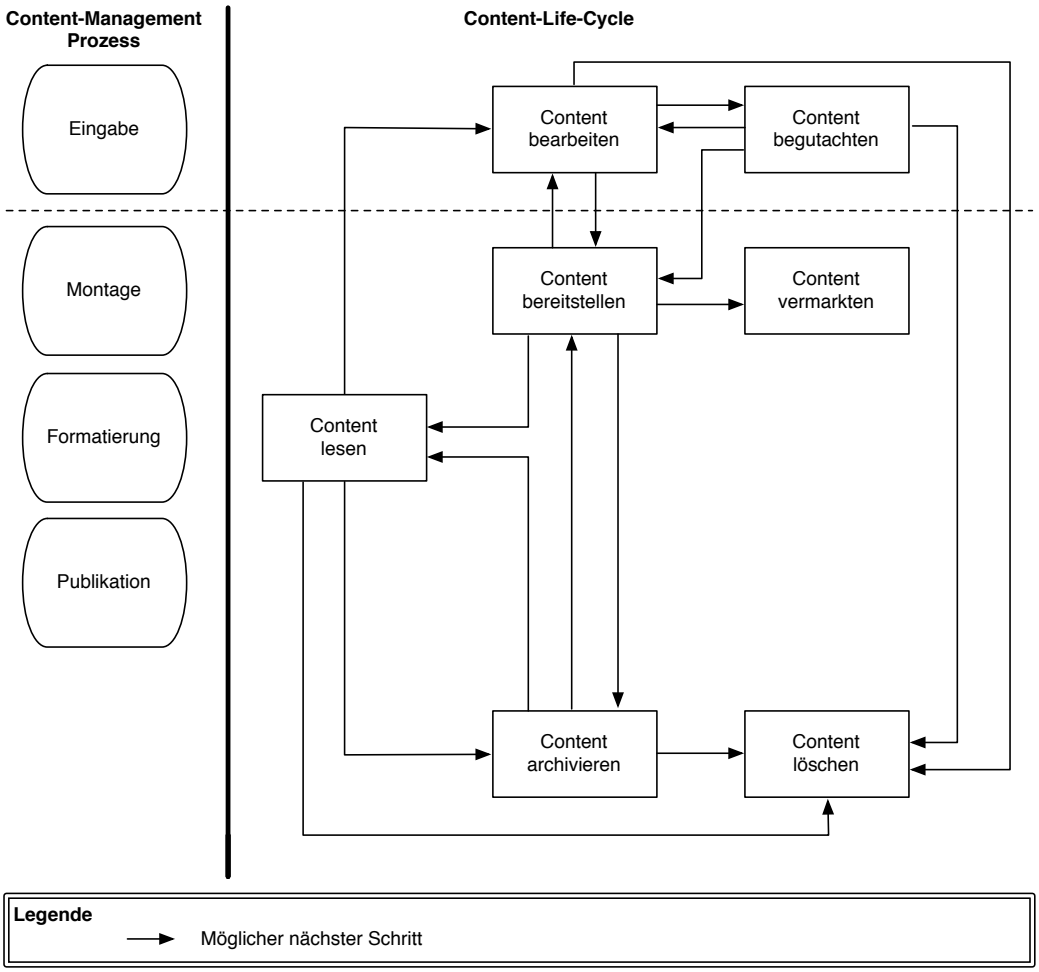

Abbildung 2.6: Abgrenzung von Content-Management und Content-Life-Cycle (Ehlers (2003), S. 109; Gersdorf (2002), S. 76)

sehen, eine m-zu-n Beziehung (Ehlers (2003), S. 109). Die Funktionen Lesen, Vermarkten, Archivieren und Löschen von Content können keiner Funktion im Content-Management-Prozess zugeordnet werden (Ehlers (2003), S. 109). Die Funktion der Bereitstellung von Content im CLC umfasst die Funktionen Montage, Formatierung und Publikation des Content-Management-Prozesses (Ehlers (2003), S. 109).

Der Content-Management Funktion *Eingabe* können die CLC Funktionen *Bearbeitung* sowie *Begutachtung* zugewiesen werden (Ehlers (2003), S. 109).

Die Funktion *Bearbeitung* beinhaltet sowohl die Erstellung von neuem Content, als auch Änderungen an bestehendem Content durch einen Benutzer der Rolle *Redakteur* (Ehlers (2003), S. 110; Büchner et al. (2001), S. 83). Je nach Anforderung kann der Redakteur den erstellten Content selbst an die Funktion *Bereitstellen* oder ihn erst zu der Funktion *Begutachtung* weiterleiten (Ehlers (2003), S. 110; Büchner et al. (2001), S. 83). Die direkte *Löschung* nach der *Bearbeitung* ist ebenfalls möglich (Ehlers (2003), S. 112).

Bei der *Begutachtung* wird der Content von einem Benutzer einer entsprechend qualifizierten Rolle hinsichtlich bestimmter Punkte überprüft (Büchner et al. (2001), S. 83; Hess und Rawolle (2000), S. 62; Karajannis und Bissel (2000), S. 68). Der Content kann dabei durch den Prüfer geändert oder ergänzt und im Anschluss akzeptiert, oder bei Einwänden an den Ersteller zurückgeleitet oder gelöscht werden (Ehlers (2003), S. 110; Büchner et al. (2001), S. 84). Entsprechend den jeweiligen Anforderungen kann der Content, wenn er durch den Prüfer akzeptiert wurde, an die Funktion *Bereitstellung* übergeben werden, oder weitere Iterationen der Funktion *Begutachtung* durch andere Prüfer durchlaufen.

Wird Content an die Funktion *Bereitstellung* übergeben, erfolgt je nach Angabe des Publikationsdatums die sofortige oder zeitverzögerte Publikation (Ehlers (2003), S. 110f; Möllering und Scheube (2001), S. 63). Hierbei werden alle drei zugeordneten Funktionen des Content-Management Montage, Formatierung und Publikation zusammen ausgeführt. Bei der Publikation sind Benutzerrestriktionen sowie das Ausgabemedium zu beachten (Ehlers (2003), S. 111).

Bereitgestellter und archivierter Content kann von Benutzern mit den entsprechenden Rechten innerhalb der Funktion *Lesen* überprüft werden, um gegebenenfalls die Funktionen *Bearbeitung*, *Bereitstellung* oder *Löschung* aufzurufen.

Nach Ablauf des festgelegten Publikationszeitraums erfolgt die *Archivierung* oder *Löschung* des Content (Ehlers (2003), S. 111; Büchner et al. (2001),

S. 84; Möllering und Scheube (2001), S. 63). Bei Publikationen, die kein festgelegtes Publikationsende besitzen, muss der Content durch einen hierzu berechtigten Benutzer manuell archiviert oder gelöscht werden (Möllering und Scheube (2001), S. 63).

Content, der archiviert wurde, kann zu einem späteren Zeitpunkt von Redakteuren nochmals bearbeitet, wieder bereitgestellt oder gelöscht werden. Sollten Änderungen an Content aufgrund von Fehlern oder sonstigen Gründen nötig sein, kann eine Publikation zurückgerufen und nach den entsprechenden Änderungen wieder neu publiziert werden. Content kann darüber hinaus an Dritte weitergegeben werden (Ehlers (2003), S. 112).

Ein Content-Management-System kann zusammenfassend definiert werden als ein Anwendungssystem, welches den Content-Management-Prozess umsetzt, den Content-Life-Cycle der einzelnen Content-Elemente unterstützt und der Erstellung einer Content-Management-Anwendung dient (Ehlers (2003), S. 113). Unter einer *Content-Management-Anwendung* (CMA) wird eine Anwendung verstanden, die durch ein Content-Management-System realisiert wurde (Ehlers (2003), S. 58).

Aufgrund der Freiheitsgrade des Content-Management-Prozess können die Anforderungen, die an ein CMS gestellt werden, deutlich variieren. Aus diesem Grund und, um sich von der Konkurrenz abzuheben, unterscheiden sich die einzelnen Ausprägungen von CMS teils erheblich (Rothfuss und Ried (2003), S. 58). Um verschiedene Ausprägungen von CMS von einander abgrenzen zu können, werden im folgenden Abschnitt Unterscheidungsmerkmale genannt, anhand welcher die CMS voneinander differenziert werden können.

2.2.4 Unterscheidungsmerkmale von Content-Management-Systemen

Content-Management-Systeme können zunächst nach dem Umfang ihrer Funktionalität in CMS im engeren Sinn und CMS im weiteren Sinn unterschieden werden (Rothfuss und Ried (2003), S. 59; Kampffmeyer (2003), S. 90). Ein *CMS im engeren Sinn* ist ein Softwaresystem, welches die programmgestützte, systematische Sammlung und Verwaltung von Inhalten (Content-Elementen) „in einem einzigen (logischen) Bestand“ (Rothfuss und Ried (2003), S. 59) ermöglicht und die sichere Arbeit mehrerer Benutzer unterstützt (Rothfuss und Ried (2003), S. 59f; Kampffmeyer (2003), S. 90). *CMS im weiteren Sinn* bezeichnet ein Softwaresystem, welches den Content-Management-Prozess und den Content-Life-Cycle der einzelnen Content-Elemente ermöglicht (Ehlers (2003), S. 113; Rothfuss und Ried (2003), S. 60; Kampffmeyer (2003), S. 90).

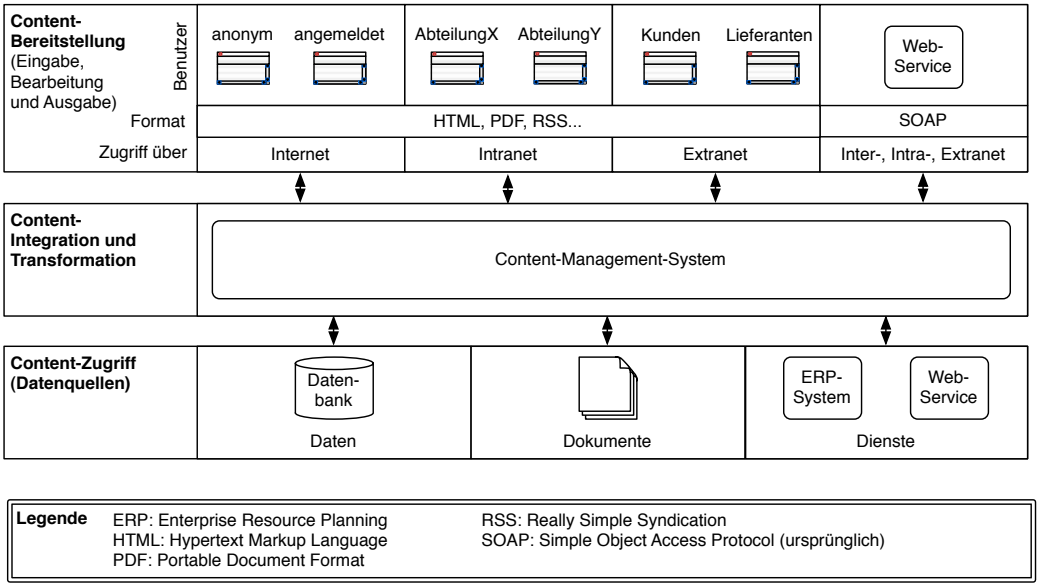

Abbildung 2.7: Schematischer Aufbau eines Content-Management-Systems (Merz (2002), S. 334)

Anhand des in Abbildung 2.7 dargestellten schematischen Aufbaus eines CMS können weitere Unterscheidungen in den dargestellten Ebenen

Content-Bereitstellung und Content-Zugriff getroffen werden. In der Ebene des Content-Zugriffs können die genutzten Datenquellen und in der Ebene der Content-Bereitstellung die Zielmedien, Zugangsmöglichkeiten und Nutzerkreise unterschieden werden.

Der Nutzerkreis einer CMA kann zum einen offen (d. h. prinzipiell kann jede Person auf die CMA zugreifen) oder geschlossen (d. h. nur eine begrenzte Gruppe hat Zugang zu der CMA) sein (Großmann und Koschek (2005), S. 30). Im Bereich der Zielgruppe können sich CMS an organisationsinterne Benutzer (z. B. über ein Intranet), an Kunden und Lieferanten (z. B. über ein Extranet oder das Internet) sowie an die Öffentlichkeit (z. B. über das Internet) richten (Möllering und Scheube (2001), S. 61). Je nach Zielgruppe unterscheiden sich auch die Anforderungen, welche an ein CMS gestellt werden (Möllering und Scheube (2001), S. 61). Organisationsinterne Benutzer sowie Kunden und Lieferanten legen neben der allgemeinen Forderung nach Aktualität der bereitgestellten Informationen besonderen Wert darauf, dass die Informationsflüsse mit den bestehenden Geschäftsprozessen abgestimmt werden (Möllering und Scheube (2001), S. 61).

Auf der Ebene des Content-Zugriffs kann eine Unterscheidung nach den unterstützten Datenquellen vorgenommen werden. Wie in Abbildung 2.7 dargestellt, können dies beispielsweise Datenbanken, Dokumente oder verschiedene Arten von Diensten sein (Merz (2002), S. 334).

Auf der Ebene der Content-Bereitstellung können die CMS anhand der unterstützten Ausgabeformate (beispielsweise die Hypertext Markup Language (HTML) oder das Portable Document Format (PDF)) unterschieden werden.

Der Begriff *Content-Syndication* bezeichnet die Übertragung von Verfügungsrechten an Content mit dem Ziel der kommerziellen Nutzung (Hess (2001), S. 83). Zur Unterstützung von Content-Syndication müssen die CMS die hierfür genutzten Formate (beispielsweise Really Simple Syndication (RSS) oder SOAP) unterstützen und bei einer kommerziellen Nutzung auch eine Abrechnung ermöglichen (Hess (2001), S. 85). Dabei kann sowohl die Ebene

des Content-Zugriffs (Content-Nachfrager) als auch die Ebene der Content-Bereitstellung (Content-Anbieter) betroffen sein.

Bezogen auf den Einsatzbereich eines Content-Management-Systems können die drei Bereiche Wissensmanagement, Electronic Commerce und Electronic Publishing unterschieden werden. Das im Bereich des Wissensmanagements verfolgte Ziel des Aufbaus von Wissensbasen kann von einem CMS durch die Ermöglichung der „Sammlung, Bewertung, Ordnung und Aufbereitung des im Unternehmen vorhandenen expliziten Wissens“ (Schoop und Gersdorf (2001), S. 993) unterstützt werden. Innerhalb des Electronic Commerce kann ein CMS zur Realisierung eines „gemeinsamen Zugriffs aller Partner entlang einer Wertschöpfungskette auf dieselbe Content Base“ (Schoop und Gersdorf (2001), S. 993) genutzt werden. Und im Rahmen des Electronic Publishing ermöglicht ein CMS durch die modulare Verwaltung der Daten, diese flexibel neu zu kombinieren und Dokumente automatisch medienspezifisch zu generieren (Schoop und Gersdorf (2001), S. 993).

2.3 Web-Content-Management-Systeme

Ein *Web-Content-Management-System* (WCMS) ist ein Content-Management-System, welches auf die Distribution von Dokumenten im Internet, Intranet oder Extranet (Winand und Schellhase (2000), S. 1334) ausgerichtet ist. Es unterstützt die hierfür relevanten Ausgabeformate wie beispielsweise die Hypertext Markup Language (Winand und Schellhase (2000), S. 1334; Karajannis und Bissel (2000), S. 66) oder Really-Simple-Syndication (RSS) (Hess (2001), S. 84). Nach den in 2.2.4 genannten Unterscheidungskriterien werden WCMS wie folgt eingeordnet:

Die Funktionalität entspricht dem eines Content-Management-System im weiteren Sinn. Somit unterstützt es nicht nur die Sammlung und Verwaltung von Content, sondern setzt auch den Content-Management-Prozess um und unterstützt den Content-Life-Cycle (Ehlers (2003), S. 113).

In Abhängigkeit des Verwendungszwecks der Web-Content-Management-Anwendung[3] (WCMA) kann sich ein Web-Content-Management-System an eine geschlossene Zielgruppe (Intranet, Extranet), eine offene Zielgruppe (Internet) oder eine Kombination beider richten (Ehlers (2003), S. 185).

Das Einsatzgebiet eines WCMS kann wiederum in Abhängigkeit der WCMA innerhalb der in Abschnitt 2.2.4 genannten Bereiche Electronic Commerce und Electronic Publishing eingeordnet werden (Schoop und Gersdorf (2001), S. 993).

In der Literatur werden die Begriffe WCMS und CMS teilweise gleichgesetzt. Beispielsweise wird der Begriff des Content-Management-Systems benutzt, allerdings als Ausgabemedium nur HTML-Dokumente betrachtet (Christ (2003), S. 15; Karajannis und Bissel (2000), S. 66) oder CMS als Kurzform von WCMS definiert (Merz (2002), S. 330).

[3] Unter einer Web-Content-Management-Anwendung wird in Anlehnung an den Begriff Content-Managment-Anwendung (siehe Abschnitt 2.2.3) eine durch ein Web-Content-Management-System realisierte Anwendung verstanden.

2.3.1 Grundmodule eines Web-Content-Management-System

Der Content-Management-Prozess und der Content-Life-Cycle erfordern bestimmte Funktionalitäten von einem Softwaresystem, welche durch Module realisiert werden. Diese Module bilden somit die Grundfunktionalität, die ein WCMS unabhängig von dem organisationsspezifischen Verwendungszweck implementieren muss. Im Folgenden werden die von Ehlers (2003, S. 113ff) identifizierten Module beschrieben.

Content-Repository

Das *Content-Repository*, das auch als Content Base (Rothfuss und Ried (2003), S.65) und Daten-Repository (Karajannis und Bissel (2000), S. 68) bezeichnet wird, stellt das zentrale Element eines WCMS dar und dient der persistenten Speicherung des Contents (Ehlers (2003), S. 113; Kretzschmar (2007), S. 217). Technisch kann ein Content-Repository die Daten beispielsweise in einer Datenbank, im Dateisystem oder in einem hybriden System aus Datenbank und Dateisystem ablegen (Ehlers (2003), S. 113; Kretzschmar (2007), S. 217). Neben Funktionen zur Speicherung, Suche und Zugriff auf Content stellen die Funktionen Indexierung und Versionierung von Content die besonderen Anforderungen an ein Content-Repository dar (Ehlers (2003), S. 113f).

Die *Sperrfunktionalität* eines Content-Repository ermöglicht den Mehrbenutzerbetrieb von WCMS (Ehlers (2003), S. 114). Hierdurch wird Content, der von einem Nutzer bearbeitet wird, vom Content-Repository bis zur Fertigstellung der Änderungen für andere Benutzer zur Bearbeitung gesperrt. Hierdurch werden Konflikte vorgebeugt, die durch sich zeitlich überschneidende Änderungen am Content durch mehrere Benutzer entstehen. Diese Sperren werden in Anlehnung an die Sperren von Quellcode-

verwaltungssystemen als Check-in und Check-out bezeichnet (Ehlers (2003), S. 114; Kretzschmar (2007) S. 214).

Berechtigungsverwaltung

Durch eine Berechtigungsverwaltung können Benutzern bestimmte Rechte bezüglich der Interaktion mit dem System zugewiesen werden (Ehlers (2003), S. 116). Die Verwaltung der Rechte sollte insbesondere bei größeren WCMS mit vielen Nutzern durch die Zuweisung von *Rollen*[4] und *Gruppen*[5] erfolgen (Rothfuss und Ried (2003), S. 74; Ehlers (2003), S. 116). Dabei muss ein Benutzer nur einmalig einer Gruppe oder Rolle zugewiesen werden und die Verwaltung der Rechte kann sich auf die Gruppen und Rollen beschränken (Rothfuss und Ried (2003), S. 74; Ehlers (2003), S. 117). Bei den Rechten, die vergeben werden können, kann beispielsweise zwischen den Aktionen Anlegen, Lesen, Verändern, Löschen und Publizieren von Content unterschieden werden (Rothfuss und Ried (2003), S. 74; Ehlers (2003), S. 116).

Workflowsteuerung

Unter einem *Workflow* wird eine formale, detaillierte und somit ausführbare Beschreibung eines Teils von einem Geschäftsprozess verstanden, welcher sich aus sequentiell oder parallel angeordneten Tätigkeitsfolgen zusammensetzt (Rautenstrauch und Schulze (2002), S. 267; Heinrich et al. (2004), S. 730). Während ein *Geschäftsprozess*[6] die „erfolgsrelevanten grundlegenden Unternehmenstätigkeiten, die zur Umsetzung der Unternehmensziele und Sicherung des Unternehmenserfolges dienen“ (Rohloff (1995), S. 84), und somit

[4] Durch die Zuweisung zu *Rollen* können gleichberechtigte Benutzer aggregiert werden (Rothfuss und Ried (2003), S. 74; Ehlers (2003), S. 116).

[5] Durch *Gruppen* ist die Bildung von Hierarchien bei der Benutzerverwaltung möglich (Ehlers (2003), S. 116).

[6] „Ein Prozeß stellt die inhaltlich abgeschlossene, zeitlich und sachlogische Abfolge der Funktionen dar, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objekts ausgeführt werden.“ (Rosemann (1996), S. 9)

das *was*, definiert, wird bei einem Workflow beschrieben, *wie* dies umzusetzen ist (Heinrich et al. (2004), S. 730).

Die *Workflowfunktionalität* kann entweder im WCMS selbst oder durch ein externes, anwendungsübergreifendes Workflow-Management-System (WFMS) realisiert sein (Ehlers (2003), S. 118; Rautenstrauch und Schulze (2002), S. 267). Ein WFMS hat den Vorteil, dass auch über das WCMS hinausgehende Prozesse abgebildet werden können (Ehlers (2003), S. 118; Büchner et al. (2001), S. 140). Um während eines Workflows auf die Content-Elemente zugreifen zu können, muss die Workflowsteuerung auf das Content-Repository zugreifen (Ehlers (2003), S. 118). Damit während eines Workflows die Benutzer nur die Aktionen an dem Content durchführen können, für die sie autorisiert sind, muss die Workflowsteuerung auch auf die Berechtigungsverwaltung zugreifen können (Ehlers (2003), S. 118).

Import- und Exportverwaltung

Die *Importfunktion* ermöglicht es, externe Daten in das WCMS aufzunehmen. Im idealen Fall liegen die zu importieren Daten schon in Form von Content (im engeren Sinne) vor und benötigen keine weitere Bearbeitung. Falls dem nicht so ist, müssen die Daten durch eine manuelle oder automatische Bearbeitung strukturiert werden. Eine automatische Bearbeitung kann immer dann erfolgen, wenn die Daten zwar unstrukturiert sind, ihre Struktur jedoch nicht variiert. Andernfalls ist eine manuelle Nacherfassung erforderlich (Ehlers (2003), S. 119f).

Zum Austausch von Daten in Form von Content kann der Standard Information-and-Content-Exchange (ICE) genutzt werden (Ehlers (2003), S. 120; Christ (2003), S. 32; Büchner et al. (2001), S. 189). Dieses ist inhaltlich frei definierbar und ermöglicht den Austausch von Daten getrennt in Inhaltswert, Inhaltsattribut und Inhaltstyp (Ehlers (2003), S. 120f).

Mittels der *Exportfunktion* kann Content aus dem Content-Repository in unterschiedlichen Formaten, wie beispielsweise dem PDF oder dem ICE-Format, ausgegeben werden (Ehlers (2003), S. 121ff).

Templateverwaltung

Für die Formatierung und das Layout des Contents der unterschiedlichen Ausgabemedien wird die *Templateverwaltung* benötigt (Ehlers (2003), S. 123). Ein *Template* kann als ein Gerüst von Platzhaltern verstanden werden, welche im Bedarfsfall mit den Inhaltswerten des Contents gefüllt werden und diesen um layoutspezifische Informationen erweitern (Bodendorf (2005), S. 106). Durch diese Anreicherung um layoutspezifische Informationen im Template wird aus dem in Abschnitt 2.2.1 beschriebenen Content im engeren Sinn somit Content im weiteren Sinn. Bedingt dadurch, dass der Content auf mehreren Medien ausgegeben werden kann, besteht zwischen den Templates, die jeweils medienspezifisch sind, und dem medienunabhängigen Content eine m-zu-n Beziehung (Ehlers (2003), S. 124). Ein Template kann somit Inhaltswerte mehrerer Contents aggregieren und der Inhaltswert eines Content kann durch mehrere Templates genutzt werden. Die Ausgabe desselben Contents auf verschiedenen Medien wird in der Literatur als cross-media-publishing sowie single-source-multiple-data-and-multiple-usage und Single-Source-Multiple-Media bezeichnet (Ehlers (2003), S. 124; Lohr und Deppe (2001), S. 12; Milleg und Wagner (2001), S. 39). Innerhalb der einzelnen Templates kann mithilfe von Programmiersprachen oder CMS-spezifischen *Tags* auf den Content zugegriffen werden (Ehlers (2003), S. 124). Templates können weitere Templates einbinden und so zu einer Verringerung der Redundanten Angaben innerhalb der Templates beitragen.

Die Berechtigungsverwaltung wird von der Templateverwaltung benötigt, um nur entsprechend autorisierten Personen Zugriff auf die Templateverwaltung zu gestatten (Ehlers (2003), S. 124). Um innerhalb eines Workflows Dokumente mithilfe von Templates erzeugen zu können, muss die Workflow-

steuerung ebenfalls zugriff auf die Templateverwaltung haben (Ehlers (2003), S. 124).

Hyperlinkverwaltung

Mittels der *Hyperlinkverwaltung* wird die Navigationsstruktur der CMA festgelegt (Ehlers (2003), S. 125). Bei der Publikation werden die Platzhalter für den Content in den Templates durch die entsprechenden Inhaltswerte aus dem Content-Repository ersetzt und mittels der Hyperlinkverwaltung um die Navigation erweitert (Ehlers (2003), S. 125).

Durch die Hyperlinkverwaltung kann auch eine Überprüfung externer Hyperlinks erfolgen. Dabei sind je nach Anforderung bei fehlerhaften Hyperlinks verschiedene Maßnahme wie die Deaktivierung der Hyperlinks und Benachrichtigung der Autoren oder des Administrators möglich (Ehlers (2003), S. 125).

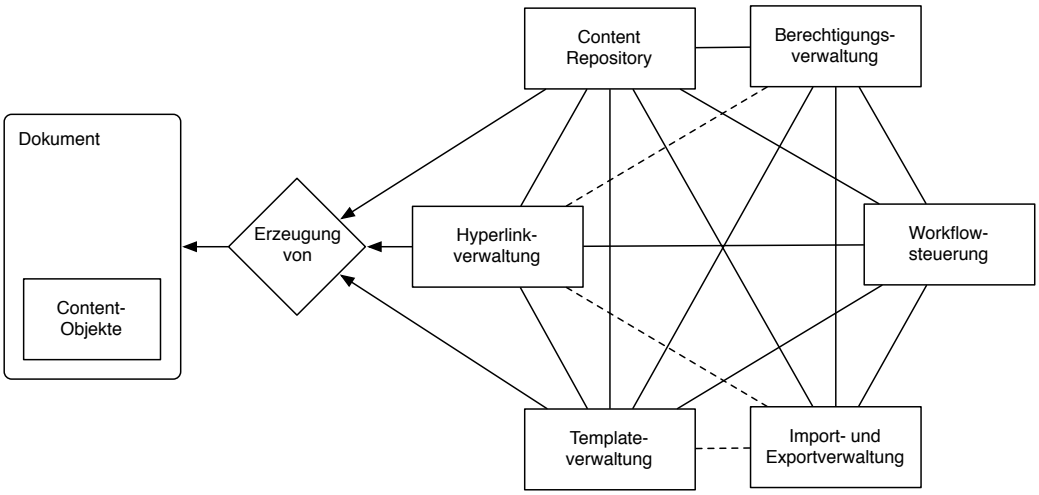

Abbildung 2.8: Grundmodule eines Content-Management-Systems (Ehlers (2003), S. 126)

In Abbildung 2.8 sind die vorgestellten Module eines Content-Management-Systems sowie deren beschriebene Verbindungen dargestellt. Soll ein Doku-

ment erzeugt werden, wird durch die Templateverwaltung zunächst das entsprechende Template ausgewählt. Dann werden die im Template referenzierten Content-Elemente aus dem Content-Repository geladen und durch das Template um das medienspezifische Layout zu Content-Objekten (siehe Abschnitt 2.2.1) erweitert. Durch die Hyperlinkverwaltung wird die Navigation des Dokuments erzeugt und gegebenfalls die Navigation der Seite angepasst.

2.3.2 Customizing eines Web-Content-Management-System

Unter Standardsoftware wird Anwendungssoftware verstanden, die aufgrund prognostizierter Anforderungen für eine große Anzahl von Anwendern entwickelt wurde und als fertiges Produkt am Markt vertrieben wird (Heinrich et al. (2004), S. 625; Rautenstrauch und Schulze (2002), S. 281). Es wird zwischen vertikaler und horizontaler Standardsoftware unterschieden. Während vertikale Standardsoftware explizit aufbauorganisatorischen Einheiten zugeordnet werden kann, ist horizontale Standardsoftware unabhängig davon einsetzbar (Rautenstrauch und Schulze (2002), S. 281). Damit lassen sich die in dieser Arbeit betrachteten Web-Content-Management-System hinsichtlich der Softwareart als vertikale Standardsoftware im Bereich der Anwendungssoftware einordnen.

Wenn Standardsoftware in einer Organisation eingeführt werden soll, muss diese an die jeweiligen Anforderungen angepasst werden. Dabei sind Anforderungslücken zu identifizieren und zu schliessen. Die Entscheidung für den Einsatz und die Auswahl einer spezifischen Standardsoftware sollte dementsprechend nach der Definition der Anforderungen erfolgen (Strahringer (2009)).

Der Begriff *Customizing* fasst Tätigkeiten an einer Standardsoftware zusammen, die erforderlich sind, um diese an die Anforderungen einer spezifischen Organisation anzupassen (Dittrich und Vaucouleur (2008), S. 37; Mertens et al. (2001), S. 5; Thomas und Scheer (2005), S. 688). Dabei kann die von

der Standardsoftware gebotene Funktionalität entweder eingeschränkt oder erweitert werden (Lassmann (2006), S. 324). Die Tätigkeiten können von der Konfiguration von Parametern bis zur Modifikation von Quellcode reichen (Brehm et al. (2001), S. 1). Customizing kann somit in die folgenden drei Kategorien unterteilt werden (Stahlknecht und Hasenkamp (2005), S. 213; Lassmann (2006), S. 324f):

Parametrisierung: Durch das Setzen von Parametern werden gewünschte Programmfunktionen initialisiert.

Konfigurierung: Bei komponentenbasierter Standardsoftware wird nach der Auswahl der benötigten Geschäftsobjekte sowie derer Attribute und Methoden ein Anwendungssystem generiert.

Ergänzungs- oder Individualprogrammierung: Erweiterung der Standardsoftware durch individuelle Programmierung entsprechend den Anforderungen.

Aus der in Abschnitt 2.3 vorgenommenen Einordnung von WCMS in die Unterscheidungsmerkmale von CMS ist ersichtlich, dass WCMS für verschiedene Einsatzzwecke sowie Benutzergruppen mit unterschiedlichen Anforderungen hinsichtlich der Content-Präsentation, des Content-Zugriffs sowie der Integration und Transformation des Contents anpassbar sein müssen (Rothfuss und Ried (2003), S. 58).

Die Anpassungen, die an einem WCMS vorzunehmen sind, können den in Abschnitt 2.3.1 beschriebenen Grundmodulen eines WCMS zugeordnet werden. Das Grundmodul Workflowsteuerung wird nicht weiter betrachtet, da diese Funktionalität wie unter 2.3.1 beschrieben, als anwendungsübergreifendes Workflow-Management-System (unabhängig vom Web-Content-Management-System) realisierbar ist und als solches in mittleren bis größeren Organisationen bereits genutzt wird (Büchner et al. (2001), S. 140). Für die Modellierung der Workflows kann durch die Business Process Modeling

Notation der Object Management Group eine produktunabhängige Modellierung vorgenommen werden (OMG (2009), S. 11; Lankhorst (2005), S. 27). Auch das in Abschnitt 2.3.1 beschriebene Grundmodul der Import- und Exportverwaltung wird nicht weiter betrachtet (für Im- und Exporte kann der in Abschnitt 2.3.1 erwähnte Standard ICE genutzt werden).

In Tabelle 2.1 befindet sich eine Übersicht der Customizing-Möglichkeiten der betrachteten Grundmodule.

Grundmodul	Customizing
Content-Repository	Definition von Content gemäß Content i. e. S. (siehe Abschnitt 2.2.1)
Berechtigungsverwaltung	Definition von Gruppen und Rollen, denen einzelne Benutzer zugewiesen werden können
	Zuweisen der Rechte (Anlegen, Lesen, Ändern, Löschen, Publizieren) zwischen Gruppen/Rollen und Elementen des WCMS
Templateverwaltung	Definition der Templates mit Zugriff auf die Content-Inhaltswerte und Navigationsstrukturen
	Festlegen verschiedener Ausgabeformate für den Content
Hyperlinkverwaltung	Definition der Navigationsstruktur der WCMA und der Autoren-Navigationsstruktur des WCMS

Tabelle 2.1: Zusammenfassung der Customizing-Möglichkeiten eines WCMS

Aus diesen erforderlichen Anpassungen, insbesondere bei der Template- und Hyperlinkverwaltung, ergibt sich, dass bei der Einführung eines WCMS in einer Organisation neben einer Parametrisierung und Konfigurierung immer eine Ergänzungsprogrammierung vorzunehmen ist.

2.4 Implikationen

Bevor die Auswahl einer Standardsoftware getroffen werden kann, müssen die fachlichen Anforderungen durch Domänenmodelle (siehe Abschnitt 2.1.2) festgelegt werden (siehe Abschnitt 2.3.2). Bei der Einführung von Standardsoftware in eine Organisation muss diese dann, wie in Abschnitt 2.3.2 beschrieben, an die organisationsspezifischen Anforderungen angepasst werden. Das hierfür benötigte Customizing kann den in Abschnitt 2.3.2 genannten Kategorien Parametrisierung, Konfigurierung und Ergänzungs- oder Individualprogrammierung zugeordnet werden. Erfordert das Customizing eine Ergänzungs- oder Individualprogrammierung (wie im Fall der Web-Content-Management-Syteme), erstreckt sich das Customizing auf die in Abschnitt 2.1.2 genannten Phasen der Softwareentwicklung. Entsprechend werden auch beim Customizing Modelle der Ebenen des Fachkonzepts, des DV-Konzepts und der Implementierung genutzt, um das Softwaresystem auf verschiedenen Abstraktionsstufen zu spezifizieren. Dabei sind die Modelle bei der modellbasierten Softwareentwicklung nur lose miteinander sowie mit der späteren Implementierung verbunden (siehe Abschnitt 2.1.2).

Wie in Abschnitt 2.1.3 angeführt, besteht das Potential, durch eine (teil-) automatische Transformation der Modelle der verschiedenen Abstraktionsstufen und somit der Nutzung bereits modellierter Sachverhalte die Effizienz im Softwareentwicklungsprozess zu steigern.

Das somit formulierte Ziel des modellgetriebenen Customizing kann durch die Nutzung von produktunabhängigen und produktabhängigen Modellierungssprachen realisiert werden. Hierbei sind im ersten Schritt, nachdem sich die Organisation für den Einsatz eines WCMS entschieden hat, die fachlichen Anforderungen unter Nutzung einer produktunabhängigen Modellierungssprache in Systemmodelle zu überführen und zu formalisieren. Durch die Produktunabhängigkeit der Modellierungssprache wird dem potentiellen Problem vorgebeugt, dass die fachlichen Anforderungen an die technischen Möglichkeiten eines spezifischen Produkts angepasst werden (Ehlers (2003),

S. 152f). Diese produktunabhängigen Modelle können dann um produktspezifische Angaben ergänzt und für die Erzeugung von Artefakten für das Customizing des WCMS genutzt werden. Die grundlegenden Funktionen eines WCMS, wie sie durch die in Abschnitt 2.3.1 beschriebenen Grundmodule bereitgestellt werden, sollten somit ohne konkreten Produktbezug in die produktunabhängigen Modelle aufgenommen werden (Ehlers (2003), S. 152). Um dies zu ermöglichen, ist eine produktunabhängige Modellierungssprache für Web-Content-Management-Anwendungen erforderlich, welche diese Funktionalitäten beachtet.

Da eine Web-Content-Management-Anwendung eine durch ein Web-Content-Management-System realisierte Web-Anwendung darstellt, wird im folgenden Kapitel zunächst auf die Grundlagen der Entwicklung von Web-Anwendungen eingegangen. Die geforderte Transformation von Modellen wird im darauf folgenden Abschnitt anhand der modellgetriebene Softwareentwicklung betrachtet. Mit der Model Driven Architecture wird im gleichen Abschnitt ein Standardisierungsansatz für die modellgetriebene Softwareentwicklung von der Object Management Group vorgestellt. Für die Erstellung einer Modellierungssprache werden die Unified Modeling Language Profile betrachtet. Um die technische Umsetzung der in Abschnitt 2.3.1 aufgezeigten allgemeinen Grundmodule aufzuzeigen und diese damit praktisch zu verifizieren, wird im letzten Abschnitt das Web-Content-Management-System OpenCms anhand dieser Grundmodule vorgestellt.

Kapitel 3

Technische Grundlagen

3.1 Entwicklung von Anwendungen für das World Wide Web

Im folgenden Kapitel werden zunächst die technischen Grundlagen zur Realisierung von Web-Anwendungen vorgestellt. Darauf folgend wird die Nutzung von Modellen bei der Modellierung von Web-Anwendungen näher betrachtet.

3.1.1 Grundlagen der Entwicklung von Web-Anwendungen

Unter *Web-Anwendungen* werden Softwaresysteme verstanden, die auf Technologien und Standards des World Wide Web Konsortiums (W3C) basieren und über einen Webclient erreichbar sind (Schwinger und Koch (2006a), S. 2).

Als Grundlage für Web-Anwendungen kann das zwischen 1989 und 1990 von Sir Timothy John Berners-Lee entwickelte Konzept eines Hypertextsystems genannt werden (Walter (2007), S. 3; Dumke et al. (2003) S. 14; Büchner et al. (2001), S. 16). Ziel war es, eine Möglichkeit zu schaffen, um Dokumente leicht auszutauschen und bearbeiten zu können (Büchner et al. (2001),

S. 16). Dieses Konzept basiert auf den Komponenten Webserver, Webclient, dem Vermittlungsprotokoll Hypertext Transfer Protocol (HTTP), dem Adressierungsschema Uniform Resource Locator (URL) sowie der Auszeichnungssprache Hypertext Markup Language (HTML) (Walter (2007), S. 3; Dumke et al. (2003) S. 14).

Der Webclient (auch Webbrowser oder kurz Browser genannt) leitet zunächst die Anfrage des Nutzers nach einem bestimmten Dokument durch die Angabe einer URL oder durch einen Mausklick auf einen Hyperlink mittels HTTP an den Webserver (Rautenstrauch und Schulze (2002), S. 190ff). Das von dem Webserver über HTTP zurückgegebene HTML-Dokument wird dann vom Browser dargestellt (Walter (2007), S. 5).

Im folgenden Abschnitt wird zunächst die Auszeichnungssprache Hypertext Markup Language vorgestellt. Mit den Cascading Style Sheets wird auf eine Möglichkeit zur Trennung der layoutspezifischen Angaben aus den HTML-Dokumenten eingegangen. Im Anschluss wird mit der Extensible Markup Language eine Metasprache für die Definition von Auszeichnungssprachen betrachtet, die Grundlage für eine Vielzahl von Ausgabe- und Austauschformaten wie die Extensible Hypertext Markup Language, Really Simple Syndication und SOAP ist.

Die Hypertext Markup Language

Das Konzept des Hypertext allgemein geht zurück auf den Artikel „As we may think“ von Vannevar Bush aus dem Jahre 1945 (Bush (1996), S. 43; Berners-Lee und Fischetti (1999), S. 17f; Ehlers (2003), S. 36). Hierin wird die fiktive Maschine „memex“ als Erweiterung des menschlichen Gehirns („MEM“ory „EX“tension) beschrieben, die einzelne Elemente über Verbindungen miteinander in Beziehung setzt und so eine nichtlineare Navigation zwischen diesen ermöglicht (Berners-Lee und Fischetti (1999), S. 17f; Ehlers (2003), S. 36). Innerhalb der HTML wird dies durch Verweise (Hyper-

links) auf andere Dokumente ermöglicht (Wöhr (2004), S. 10). Des Weiteren ist die Hypertext Markup Language eine deskriptive Auszeichnungssprache, d. h. die Auszeichnungen, die mittels HTML vorgenommen werden, fügen dem Dokument zusätzliche Informationen über die Bedeutungen bestimmter Teile des Dokuments hinzu (Bormann und Bormann (2002) S. 234). Das Konzept der Auszeichnungssprachen kommt aus dem Verlagswesen. Hierbei werden typografische Anweisungen innerhalb eines Dokuments mittels bestimmter Zeichen angegeben (Nussbaumer und Gaedke (2006), S. 112). Diese bestimmte Zeichen, die Sprachelemente von HTML, werden Tags genannt und werden mittels spitzer Klammern markiert (Ehlers (2003), S. 65). In den Anfängen der HTML existierten nur Tags, welche auf die Beschreibung der Struktur der Dokumente ausgerichtet waren. Im Laufe der Zeit wurden diese um Tags für das Layout erweitert (Wöhr (2004), S. 42f; Bormann und Bormann (2002) S. 525ff).

Die HTML-Spezifikation liegt aktuell in der Version 4.01 vor und ist eine Anwendung der Standard Generalized Markup Language (SGML). Die SGML kann genutzt werden, um Grammatiken von Auszeichnungssprachen durch eine Dokumenttypdefinition (DTD) zu definieren (Raggett et al. (1999); Nussbaumer und Gaedke (2006), S. 117; Wöhr (2004), S. 44). Während Text direkt in HTML eingebettet und durch verschiedene Tags um Layout- und Strukturinformationen angereichert werden kann, werden andere diskrete Informationsarten (beispielsweise Grafiken und Bilder) und auch kontinuierliche Informationsarten (z. B. Animationen und Filme) durch einen Verweis (Hyperlink) referenziert (Dumke et al. (2003), S. 36; Ehlers (2003), S. 37). Die Möglichkeiten zur Auszeichnung mittels HTML sind durch die fest vorgegebene Menge an HTML-Tags beschränkt. Einzelne HTML-Tags bieten auch die Möglichkeit der Angabe von bestimmten Attributwerten, wobei die Namen der Attribute wie die der Tags innerhalb der DTD vorgeben sind.

Cascading Style Sheets

Aufgrund der Erweiterung der Sprachelemente von HTML um layoutspezifische Tags sowie durch die Nutzung der Elemente für Zwecke des Layouts kam es zu einer Mischung von Struktur- und Layoutinformationen in den HTML-Dokumenten (Bormann und Bormann (2002) S. 527). Die Cascading Style Sheets (CSS) ermöglichen im Gegenzug die Trennung des Inhalts vom Layout (Wöhr (2004), S. 86f). Somit ist es möglich, HTML ausschließlich für die Strukturbeschreibung des Dokuments zu nutzen und die Layoutinformationen mittels CSS hinzuzufügen (Wöhr (2004), S. 86f; Bormann und Bormann (2002) S. 528). Dies kann mittels CSS auf die folgenden drei, nach Flexibilität geordneten Ansätzen geschehen (Wöhr (2004), S. 86f):

1. Einbindung der Layoutinformationen im generischen Attribut „style“
2. Definition der Layoutinformationen im „head“-Bereich der Seite
3. Definition der Layoutinformationen in einer separaten Datei

Die Referenzierung einzelner Elemente kann bei der Definition der Layoutangaben im „head“-Bereich sowie in einer externen Datei unter Hilfe der generischen Attribute „style“ und „class“ erfolgen (Wöhr (2004), S. 87). Während der erste Ansatz hinsichtlich der Trennung des Layouts noch keine Vorteile bringt, können beim zweiten Ansatz die Layoutinformationen des HTML-Dokuments in den „head“-Bereich der HTML-Dokumente verschoben werden und redundante Angaben bei gleichartigen Elementen durch die Auswahl mit CSS vermieden werden. Der dritte und zugleich flexibelste Ansatz ermöglicht die Nutzung von einen oder mehreren externen CSS-Dateien für die Layoutinformationen mehrerer Dokumente.

Die Extensible Markup Language

Die Extensible Markup Language (XML) ist im Gegensatz zu HTML keine konkrete Ausprägung einer Auszeichnungssprache, sondern eine Metasprache, auf deren Basis Auszeichnungssprachen definiert werden können (Wöhr (2004), S. 52). XML ist somit auch keine Anwendung von SGML, sondern wie SGML eine Metasprache. Dabei basiert XML zwar auf SGML, ist aber keine echte Untermenge, da verschiedene Vereinfachungen gegenüber SGML vorgenommen werden (Bormann und Bormann (2002) S. 259). Eine Auszeichnungssprache, die mittels XML definiert wurde, ist die Extensible Hypertext Markup Language (XHTML). Aktuell in der Version 1.0 ist XHTML eine XML-basierte Definition der Elemente von HTML 4.01 (Wöhr (2004), S. 52; Ehlers (2003), S. 65). Die Spezifikation von Auszeichnungssprachen kann bei XML mittels der von SGML bekannten DTD sowie durch XML Schema erfolgen (Bormann und Bormann (2002) S. 373). XML Schema bietet gegenüber DTD die Vorteile, dass die Beschreibung der Elemente selbst durch XML erfolgt und eigene Datentypen für Attribute sowie Elementinhalte definiert werden können (Bormann und Bormann (2002) S. 373).

3.1.2 Modellierung von Web-Anwendungen

Komplexe Web-Anwendungen erfordern eine systematische Vorgehensweise und eine genaue Spezifikation der Anforderungen, wie sie mithilfe von Modellen erfolgen kann (Schwinger und Koch (2006b), S. 40). Dabei können Web-Anwendungen entweder als Erweiterung von bisherigen Informationssystemen um die Ebene der nichtlinearen Navigation und komplexen Informationsstrukturen angesehen werden oder als Erweiterung von Webseiten um Anwendungsfunktionalität (Baresi et al. (2000), S. 1). Bei der Modellierung von Web-Anwendungen ist es erforderlich, die Modelle, Methoden und Techniken der Modellierung von Informationssystemen mit denen von Webseiten (Hypermedia) zu integrieren (Baresi et al. (2000), S. 1).

Innerhalb der letzten Jahre wurden eine Vielzahl verschiedener Methoden mit unterschiedlichen Notationen für die Entwicklung von Web-Anwendungen vorgeschlagen (Schwinger und Koch (2006b), S. 41). Grundlage der meisten Modellierungsmethoden für Web-Anwendungen ist die Unified Modeling Language (Schwinger und Koch (2006b), S. 41). Trotz der verschiedenen Notationen der einzelnen Ansätze lassen sich Grundsätze identifizieren, die in vielen dieser Ansätze befolgt werden (Koch und Kraus (2003), S. 1). So erfolgt bei den meisten Methoden eine Unterteilung der Modelle in die drei Ebenen Content, Navigation und Präsentation (Florescu et al. (1998), S. 60; Fraternali (1999), S. 228; Schwinger und Koch (2006b), S. 41 Ehlers (2003), S. 137; Retschitzegger und Schwinger (2000), S. 2).

In Modellen der Contentebene wird die Informationsstruktur und die semantischen Beziehungen zwischen einzelnen Informationseinheiten festgelegt, wodurch Redundanzen vermieden werden können (Fraternali (1999), S. 230; Schwinger und Koch (2006b), S. 41). Für die Modellierung des Content genügt bei einfachen, dokumentenzentrierten Web-Anwendungen die Betrachtung der Struktur durch die Datenmodellierung in Form von beispielsweise UML Klassendiagrammen. Komplexere Web-Anwendungen erfordern zusätzlich eine Betrachtung des Verhaltens, wie sie in der UML durch z. B. Zustands- und Interaktionsübersichtsdiagramme ermöglicht wird (Schwinger und Koch (2006b), S. 45). Die Content-Modellierung stützt sich auf Konzepte und Methoden der Datenmodellierung und objektorientierten Modellierung (Schwinger und Koch (2006b), S. 45).

Die nichtlineare Navigationsmöglichkeit von Hypertext ist bei der Modellierung zu beachten (Schwinger und Koch (2006b), S. 46). Modelle der Navigationsebene definieren, wie und wo Informationen innerhalb der Anwendung durch den Benutzer gefunden werden können und wie die Benutzer zwischen diesen navigieren können (Fraternali (1999), S. 230; Schwinger und Koch (2006b), S. 42). Hierbei können das Hypertext-Stukturmodell und das Zugriffsmodell unterschieden werden (Schwinger und Koch (2006b), S. 47).

Das *Hypertext-Strukturmodell* definiert die Struktur der unterschiedlichen Sichten auf den Content (Schwinger und Koch (2006b), S. 47). Innerhalb des *Zugriffsmodells* wird festgelegt wie die einzelnen Sichten innerhalb der Anwendung erreicht werden können (Schwinger und Koch (2006b), S. 49). Es können verschiedene Zugriffsstrukturen wie *Index* (eine Liste einer Klasse von Content-Elementen) und *Guided-Tour* (eine sequentielle Anordnung von Knoten) unterschieden werden (Schwinger und Koch (2006b), S. 49). In Lyardet et al. (1999), Rossi und Schwabe (2002) und Germán und Cowan (2000) sind Navigations-Designmuster, die für eine gute Handhabbarkeit der Anwendung genutzt werden können, beschrieben (Schwinger und Koch (2006b), S. 49). Durch Modelle der Navigationsebene werden die Elemente der Ebene des Content mit den Elementen der Ebene der Präsentation verbunden (Schwinger und Koch (2006b), S. 50). In Abhängigkeit von der Modellierungsmethode besteht eine unterschiedlich starke Verbindung zwischen den Modellen der Content- und Navigationsebene (Schwinger und Koch (2006b), S. 50).

Innerhalb von Modellen der Präsentationsebene wird festgelegt, wie der Content und die Navigation dem Benutzer dargestellt werden (Fraternali (1999), S. 230). Hierbei werden keine ästhetischen Aspekte, sondern vornehmlich die Struktur der Präsentation betrachtet (Schwinger und Koch (2006b), S. 42). Durch die Modellierung der Struktur wird die Aggregation der Seiten aus Elementen des Hypertext-Strukturmodells beschrieben. Bei dieser Modellierung können die drei Elementtypen Präsentationsseite, Präsentationseinheit und Präsentationselement unterschieden werden (Schwinger und Koch (2006b), S. 51). Eine Präsentationsseite ist eine bestimmte Sicht, die beispielsweise einer Benutzergruppe zugeordnet werden kann und verschiedene Präsentationseinheiten enthält (Schwinger und Koch (2006b), S. 51). Präsentationseinheiten Gruppieren logisch zusammengehörige Präsentationselemente zu einer Einheit. Präsentationselemente repräsentieren einzelne Content-Elemente und können verschiedenen Typs wie Text, Bild, Audio- und Videodatei sein (Schwinger und Koch (2006b), S. 51). Die strukturelle Modellierung kann durch UML Klassendiagramme mittels Komposition von

Klassen und Vergabe von UML Stereotypen für die Präsentationsseiten, -einheiten und -elemente erfolgen (Schwinger und Koch (2006b), S. 51). Die Modellierung des Verhaltens der Präsentationsebene kann durch verschiedene Verhaltensdiagramme erfolgen, wobei Interaktionen mit den Ebenen Content (Methoden auf Content-Elementen) und Navigation zu beachten sind (Schwinger und Koch (2006b), S. 52).

Die Trennung dieser drei Ebenen, die z. B. durch die explizite Angabe der Abhängigkeiten und Zuordnungen der Modellelemente der Ebenen sichergestellt werden kann, ist eine Voraussetzung um eine hohe Wiederverwendbarkeit und Flexibilität zu erreichen sowie die Komplexität der Web-Anwendung aufteilen zu können (Retschitzegger und Schwinger (2000), S. 2; Schwinger und Koch (2006b), S. 41). Durch die Trennung der Ebenen Content, Navigation und Präsentation muss die Struktur des Contents nur einmal definiert werden und kann an unterschiedlichen Stellen der Navigation auf verschiedene Art präsentiert werden (Retschitzegger und Schwinger (2000), S. 2; Schwinger und Koch (2006b), S. 42). In Abhängigkeit der zu realisierenden Web-Anwendung kann die Gewichtung der Modelle der drei Ebenen variieren (Schwinger und Koch (2006b), S. 42).

3.2 Modellgetriebene Softwareentwicklung

Das Ziel der modellgetriebenen Softwareentwicklung ist die Verringerung der so genannten semantischen Lücke (France und Rumpe (2007), S. 1; Petrasch und Meimberg (2006), S. 27), die, wie in Abbildung 3.1 dargestellt, zwischen dem Problembereich der Anwendungsdomäne und dem Lösungsbereich der Implementierungsdomäne besteht (France und Rumpe (2007), S. 1; Petrasch und Meimberg (2006), S. 27; Gruhn et al. (2006), S. 16). Dies soll durch den Einsatz von Techniken erreicht werden, welche die automatische Transformation von Artefakten der Anwendungsdomäne zu Artefakten der Implementierung ermöglichen (France und Rumpe (2007), S. 1). Mit dem Begriff *Artefakt* werden alle (Zwischen-) Ergebnisse einer Entwicklungstätigkeit, in Form von beispielsweise Modellen, Quellcode, Konfigurationsdateien oder Textdokumenten, bezeichnet (Petrasch und Meimberg (2006), S. 12).

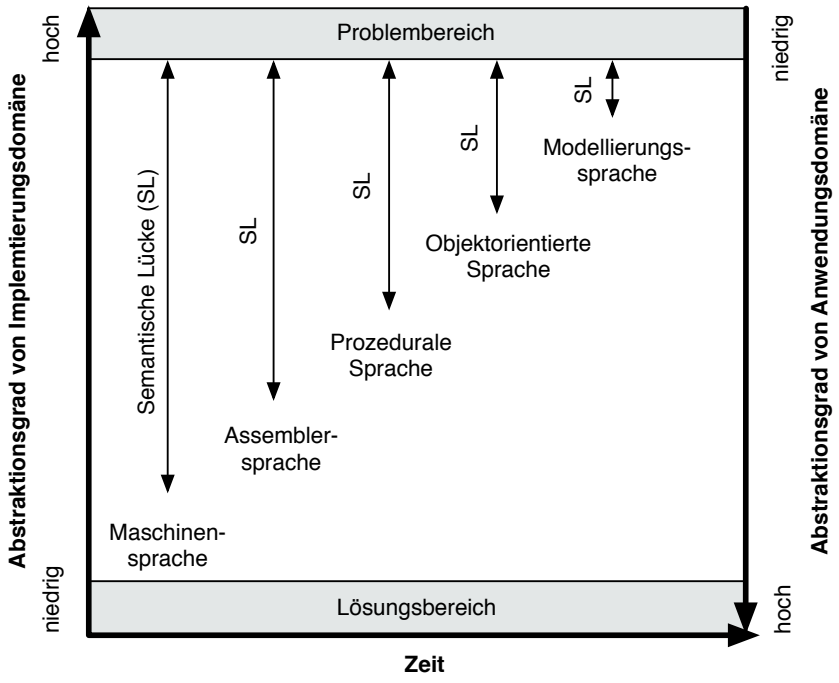

Abbildung 3.1: Abstraktionsgrad von Implementierungs- und Anwendungsdomäne (Petrasch und Meimberg (2006), S. 46; Gruhn et al. (2006), S. 15; Frankel (2003), S. 61)

Bei der modellgetriebenen Softwareentwicklung werden textuelle oder grafische Modelle (Stahl et al. (2007), S. 4) genutzt, um das zu realisierende Softwaresystem auf verschiedenen Abstraktionsebenen sowie unter verschiedenen Perspektiven zu beschreiben und eine automatische Transformation zu ermöglichen (France und Rumpe (2007), S. 1; Petrasch und Meimberg (2006), S. 28). Voraussetzung hierfür ist eine (semi-) formal definierte, auf die Anwendungsdomäne ausgerichtete und somit domänenspezifische Sprache (Stahl et al. (2007), S. 4ff). Unter einer formal definierten Sprache wird eine Sprache verstanden, die auf einem Regelwerk basiert, das "mathematisch exakt, widerspruchsfrei und vollständig definiert werden kann" (Petrasch und Meimberg (2006) S. 42). Während dieses Regelwerk bei einer formalen Sprache sowohl die Syntax (die Elemente der Sprache) als auch die Semantik (die Bedeutung dieser Elemente) definiert, wird bei einer semi-formalen Sprache nur die Syntax definiert (Frappier und Habrias (2006), S. 4). Durch diese formale Spezifikation wird die Mehrdeutigkeit der natürlich-sprachlichen Beschreibungen des Anwendungsbereichs verringert und die semantische Lücke zwischen der Anwendungs- und Implementierungsdomäne, wie in Abbildung 3.1 dargestellt, verkleinert (Petrasch und Meimberg (2006), S. 28).

Neben den Modellen sind die Transformatoren, die Modelle als Quellartefakte nutzen, um bestimmte Zielartefakte zu erzeugen, ein weiterer Bestandteil der modellgetriebenen Softwareentwicklung (Stahl et al. (2007), S. 5). Bei den Transformatoren können die beiden Ansätze der Generatoren und Interpreter unterschieden werden. Ein Generator erzeugt aus einem Quellartefakt ein Zielartefakt. Ein Interpreter dagegen führt, basierend auf einem Quellartefakt, bestimmte Aktionen zur Laufzeit aus. Mittels beider Ansätze entsteht aus Modellen lauffähige Software (Stahl et al. (2007), S. 12).

Im Gegensatz zu der in Abschnitt 2.1.2 beschriebenen modellbasierten Softwareentwicklung werden bei der modellgetriebenen Softwareentwicklung die Modelle nicht mehr nur für die Spezifikation und Dokumentation genutzt, sondern bilden das Bindeglied zwischen der Anwendungsdomäne und der Implementierungsdomäne und werden somit zu den zentralen Elementen der

Softwareentwicklung (Kempa und Mann (2005), S. 298;France und Rumpe (2007), S. 1; Stahl et al. (2007), S. 3; Gruhn et al. (2006), S. 20). Hierdurch erfolgt ein Übergang von der deskriptiven zu einer transienten (sowohl deskriptiv, als auch präskriptiv, siehe Abschnitt 2.1.1) Verwendung von Modellen innerhalb des Softwareentwicklungsprozesses (Gruhn et al. (2006), S. 20).

Durch die Einbeziehung der Modelle in die Erstellung der Implementation wird die in Abbildung 3.2 auf der linken Seite dargestellte, in der Praxis häufig vorgenommene Verkürzung des iterativen Prozess bei der Softwareentwicklung verhindert (Kleppe et al. (2003), S. 7). Der iterative Prozess wird durch die modellgetriebenen Softwareentwicklung, wie in Abbildung 3.2 auf der rechten Seite dargestellt, auf alle Phasen, in denen Systemmodelle (siehe Abschnitt 2.1.2) erstellt werden, ausgedehnt (Kleppe et al. (2003), S. 7). Hierdurch wird auch das in Abschnitt 2.1.2 erwähnte Problem, dass Modelle über den Entwicklungsverlauf hinweg inkonsistent werden (weil Änderungen nicht nachgetragen werden), verhindert. Die Dokumentationsfunktion wird somit durch die Aktualität und Formalität der Modelle besser unterstützt (Stahl et al. (2007), S. 14).

Die Erstellung einer domänenspezifischen Sprache kann nach folgenden vier verschiedenen Ansätzen erfolgen (v. Deursen et al. (2000), S. 28f):

- Entwicklung einer eigenen neuen Sprache (externe DSL (Stahl et al. (2007), S. 98))
- Einbettung der DSL in eine bereits existierende (Host-) Sprache (interne DSL (Stahl et al. (2007), S. 98))
- Nutzung eines Präprozessors, um die eigenen Sprachelemente in eine bestehende Sprache übersetzen zu lassen
- Aufnahme der eigenen Sprachelemente in den Compiler einer bestehenden Sprache

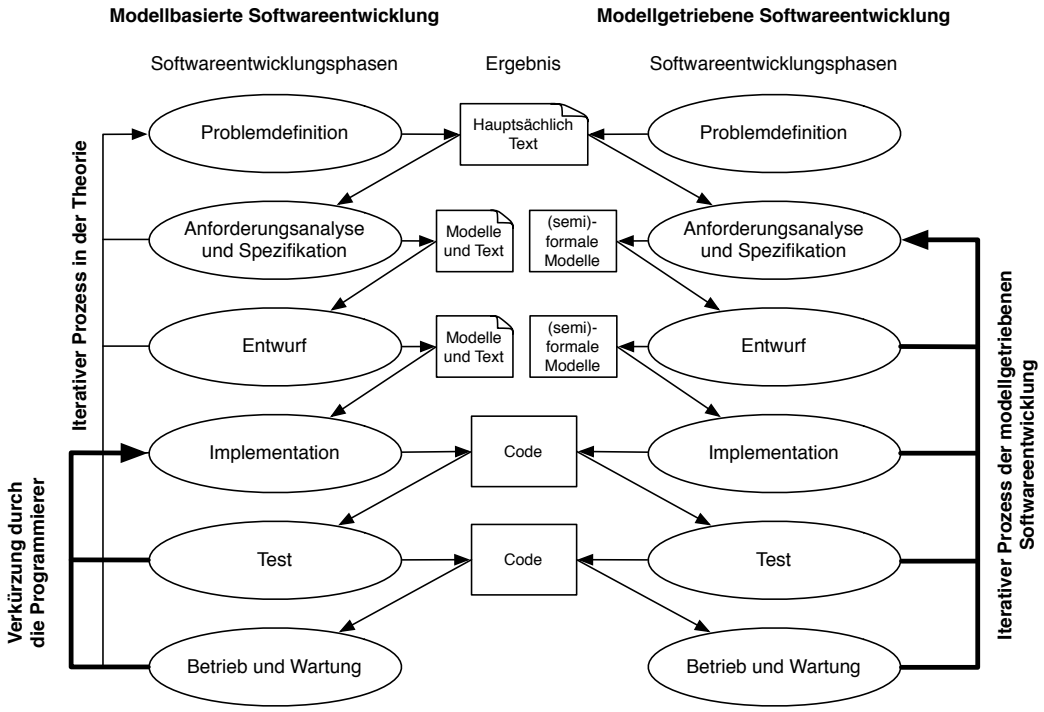

Abbildung 3.2: Iterativer Softwareentwicklunsprozess bei modellgetriebener Softwareentwicklung (Kleppe et al. (2003), S. 3ff)

Nach dieser allgemeinen Vorstellung der modellgetriebenen Softwareentwicklung wird im folgenden Abschnitt die Model Driven Architecture, das Standardisierungsbestreben der Object Management Group zur modellgetriebenen Softwareentwicklung, vorgestellt.

3.3 Model Driven Architecture

Die Object Management Group ist eine 1989 gegründete, unabhängige, offene und nicht-kommerzielle Standardisierungsorganisation im Bereich der Softwareentwicklung, die mehr als 800 Mitglieder zählt (Petrasch und Meimberg (2006), S. 36).

Die primären Ziele der MDA sind die Verbesserungen der Portabilität, der Interoperabilität und der Wiederverwendung von Artefakten der Softwareentwicklung (Miller und Mukerji (2003), S. 2-2). Dies soll durch die Aufteilung der Komplexität auf Modelle und Plattformen verschiedener Abstraktionsebenen erfolgen (Kempa und Mann (2005), S. 298). Dadurch wird eine Trennung der fachlichen und technischen Spezifikationen angestrebt (Gruhn et al. (2006), S. 119). Zu diesem Zweck werden vier Modellarten unterschieden: berechnungsunabhängige Modelle (Computation Independent Model), plattformunabhängige Modelle (Platform Independent Model), plattformspezifischen Modelle (Platform Specific Model) und Plattformmodelle (Platform Model) (Miller und Mukerji (2003), S. 2-5f).

3.3.1 Modelle der Model Driven Architecture

Das *berechnungsunabhängige Modell (CIM: Computation Independent Model)* ist das Ergebnis der Phase der Problemdefinition des Softwareentwicklungsprozesses und dient somit der fachlichen Beschreibung des Softwaresystems unabhängig von jeder Hard- und Software (Miller und Mukerji (2003), S. 2-5f; Petrasch und Meimberg (2006), S. 100; Kempa und Mann (2005), S. 299). Es wird für die Abstimmung zwischen den Stakeholdern der Anwendungsdomäne und den Softwarearchitekten genutzt (Kempa und Mann (2005), S. 299).

Die weiteren Modellarten bauen auf dem Begriff der *Plattform* auf. Dieser bezeichnet eine Menge von (Sub-) Softwaresystemen und Techniken, die eine kohärente Menge an Funktionen über definierte Schnittstellen bereitstellen. Die Schnittstellen können von Anwendungen genutzt werden, ohne Informationen über die Implementierung zu besitzen (Miller und Mukerji (2003), S. 2-3).

In einem *plattformunabhängigen Modell (PIM: Platform Independent Model)* werden die fachlichen Funktionalitäten des Softwaresystems unabhängig von

der technischen Plattform, die für die Realisierung genutzt wird, modelliert (Miller und Mukerji (2003), S. 2-5). Da mehrere Plattformen auf verschiedenen Abstraktionsebenen für die Modellierung eines Softwaresystems genutzt werden können (Miller und Mukerji (2003), S. 6-4), ist die Bezeichnung als plattformunabhängig und plattformabhängig immer relativ zu einer konkreten Plattform zu sehen (Kempa und Mann (2005), S. 299).

Das *Plattformmodell (PM: Platform Model)* beschreibt die Konzepte, die durch die Plattform umgesetzt werden und die von ihr angebotenen Funktionalitäten (Miller und Mukerji (2003), S. 2-6; Petrasch und Meimberg (2006), S. 103). Diese Plattformmodelle können generisch, technik- und herstellerspezifisch sein (Miller und Mukerji (2003), S. 2-4).

Das *plattformspezifische Modell (PSM: Platform Specific Model)* ist das Ergebnis der Modelltransformation aus dem plattformunabhängigen Model und einem Plattformmodel (Petrasch und Meimberg (2006), S. 103). Bei dieser Transformation werden die im PIM modellierten Daten mit Daten aus dem Plattformmodell angereichert (Kempa und Mann (2005), S. 299). Die Transformation kann manuell, semi-automatisch oder vollautomatisch erfolgen (Miller und Mukerji (2003), S. 3-7; Kempa und Mann (2005), S. 299). Durch die Nutzung verschiedener Plattformmodelle können zu einem PIM mehrere PSM existieren (Kleppe et al. (2003), S. 6). Eine Sonderform des PSM ist die *plattformspezifische Implementierung (PSI: Platform-specific Implementation)*. Diese stellt die letzte Ebene der Softwareentwicklung, zumeist in Form von Quellcode, dar (Petrasch und Meimberg (2006), S. 106).

Die *Transformationsaufzeichnungen (Record of Transformation)* sind ein weiteres Ergebnis der Modelltransformation und verzeichnen, welche Elemente des PIM, durch welche Mapping-Regeln, in welche Elemente des PSM transformiert wurden (Miller und Mukerji (2003), S. 3-7). Durch eine Transformationsaufzeichnung wird die Synchronisierung zwischen PIM und PSM bei Änderungen an einem der beiden unterstützt (Miller und Mukerji (2003), S. 3-8).

Um die Modelle, deren Austausch sowie eigene Metamodelle zu spezifizieren, vereint die MDA mehrere etablierte Standards der OMG (Fettke und Loos (2003), S. 556). Hierzu zählt die *Unified Markup Language*, die neben der Nutzung als eigene Modellierungssprache durch die Erweiterung des Sprachumfangs mittels UML-Profil auch als Hostsprache für eine interne DSL eingesetzt werden kann.

Durch ein UML-Profil kann das Metamodell der UML somit für eine bestimmte Plattform oder Fachdomäne um Stereotypen und Randbedingungen erweitert werden (Jeckle et al. (2005), S. 538f). Der Hauptvorteil von UML-Profilen liegt allerdings nicht in der Erweiterung, sondern in der Beschränkung der UML auf die spezifizierten Elemente (Moreno et al. (2006), S. 2).

Die *Object Constraint Language (OCL)* kann bei Transformationen sowie der semantischen Präzisierung der Modelle eingesetzt werden (Petrasch und Meimberg (2006), S. 23; Fettke und Loos (2003), S. 556).

Die *Meta Object Facility (MOF)* ist eine abstrakte Sprache, die genutzt werden kann, um Metamodelle zu spezifizieren (OMG (2002), S. xi). Die OMG hat auf Basis der MOF auch das Metamodell der UML spezifiziert. Dadurch ist die MOF das Meta-Metamodell von Modell-Instanzen der UML.

XML Metadata Interchange (XMI) definiert basierend auf der Extensible Markup Language die Serialisierung von Sprachen, die MOF-konform spezifiziert sind (Fettke und Loos (2003), S. 556). XMI stellt somit einen standardisierten Austauschmechanismus von Modellen zwischen verschiedenen Modellierungswerkzeugen und Transformatoren dar (Petrasch und Meimberg (2006), S. 84).

3.3.2 Transformationen in der Model Driven Architecture

Die Modelltransformationen zwischen den plattformunabhängigen und plattformabhängigen Modellen basieren auf zwei Arten von Transformationsregeln, welche in der MDA als Mappings bezeichnet werden (Miller und Mukerji (2003), S. 3-2). Diese können in Model-Type Mappings und Model-Instance Mappings (in Abbildung 3.3 durch die dick gestrichelten Kästen von einander getrennt) unterschieden werden (Miller und Mukerji (2003), S. 3-2ff).

Wie in Abbildung 3.3 dargestellt, können die *Model-Type Mappings* in die Metamodell-Transformationen und die Modell-zu-Modell-Transformationen unterschieden werden. Bei den Metamodell-Transformationen wird das Metamodell des PIM auf das Metamodell des PSM abgebildet. Die Spezifikation der Metamodelle von den Modellierungssprachen kann mittels der Meta Object Facility erfolgen. Bei den Modell-zu-Modell Transformationen können bestimmte Typen des PIM auf bestimmte Typen des PSM abgebildet werden (Miller und Mukerji (2003), S. 3-3; Patig (2007), S. 19).

Die zweite Art von Modelltransformationen sind die *Model-Instance Mappings*, bei denen innerhalb der Modelle Modellelemente markiert werden, um vom Transformator erkannt und mittels der Transformationsdefinition zu einem Konzept des PSM transformiert zu werden (Miller und Mukerji (2003), S. 3-3f). Markierungen können beispielsweise mittels UML-Profilen und verschiedenen Transformationsregeln zur Nutzung bereitgestellt werden (Miller und Mukerji (2003), S. 3-5).

Laut der OMG kann davon ausgegangen werden, dass die meisten Transformationen aus einer Kombination aus Model-Type und Model-Instance Mapping bestehen. Die Schwäche der reinen Model-Type Mappings ist, dass keine zusätzlichen Markierungen durch den Entwickler oder Architekten vorgenommen werden können. Hierdurch ist die Transformation einzig von den Daten im plattformunabhängigen Modell bestimmt. Auch bei Model-Instance Map-

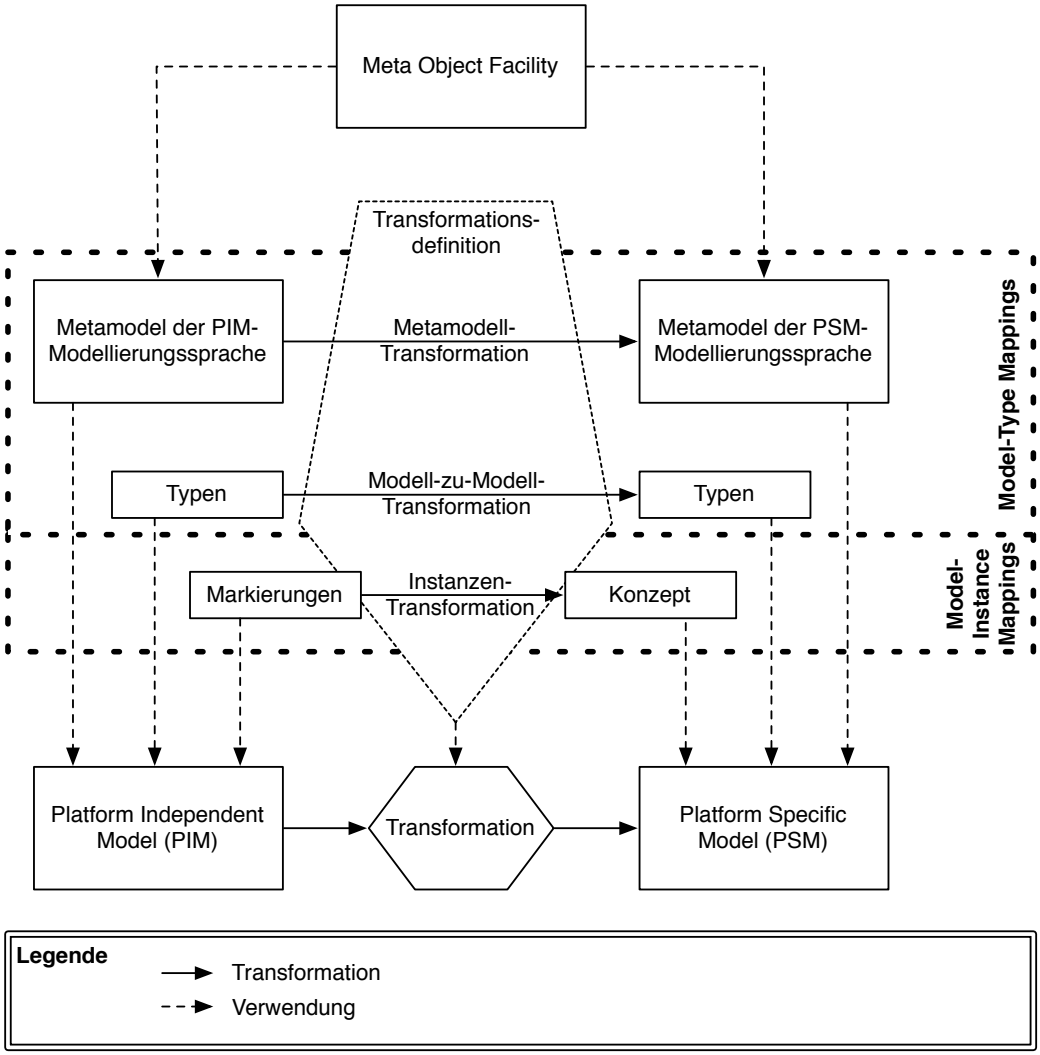

Abbildung 3.3: Modelltransformationen nach der Model Driven Architecture (Patig (2007), S. 19; Kleppe et al. (2003), S. 105; Miller und Mukerji (2003), S. 3-2ff)

pings müssen die Modelltypen implizit beachtet werden, da bestimmte Elemente nur für bestimmte Markierungen geeignet sind. (Miller und Mukerji (2003), S. 3-4).

Bezüglich der Methoden der Modell-Transformation nennt die OMG die folgenden vier Beispiele, wie Transformationen von einem plattformun-

abhängigen in ein plattformabhängiges Modell durchgeführt werden können (Miller und Mukerji (2003), S. 4-2f):

Manuelle Transformation: Die Entscheidungen für die plattformspezifische Implementierung werden manuell getroffen. Dadurch reduziert sich der Nutzen der MDA auf die explizite Unterscheidung zwischen plattformunabhängigen und plattformspezifischen Modellen.

Transformation mittels Profilen: Ein mithilfe eines plattformunabhängigen UML-Profils erstelltes Modell wird unter Nutzung eines plattformspezifischen UML-Profils durch Markierungen erweitert und in ein plattformspezifisches Modell transformiert.

Transformation mittels Pattern und Markierungen: Das plattformunabhängige Modell nutzt Markierungen, um Elemente von Pattern zu kennzeichnen, die bei der Transformation genutzt werden können.

Automatische Transformation: Falls das plattformunabhängige Modell alle relevanten Daten für die Transformation besitzt, wird kein plattformspezifisches Modell benötigt um die Zielartefakte zu generieren.

3.3.3 Unified Modeling Language Profile

Durch ein UML-Profil kann, wie schon in Abschnitt 3.3 erwähnt, eine domänenspezifische Modellierungssprache auf der Basis der UML erstellt werden. Dabei können aber nur bestimmte neue Elemente (Stereotypen) mit Attributen angelegt und Regeln (Constraints) aufgestellt werden. Die UML bleibt somit erhalten und das Profil darf auch nicht in Widerspruch zu den Konzepten der UML stehen (OMG (2007), S. 177; Jeckle et al. (2005), S. 528). Aus diesem Grund wird bei UML-Profilen auch von einer leichtgewichtigen Erweiterung der UML gesprochen (OMG (2007), S. 14).

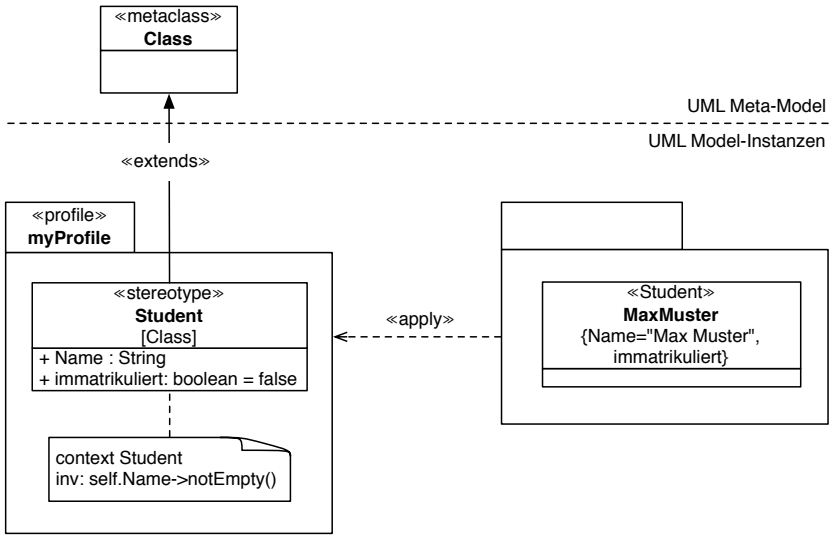

Abbildung 3.4: Unified Modeling Language Profil

Ein Profil ist aus Modellierungssicht ein UML-Paket, welches die innerhalb von Profilen zur Verfügung stehenden Erweiterungsmechanismen Stereotypen, Tagged Values und Constraints gruppiert (Jeckle et al. (2005), S. 530). Für die Definition von Profilen werden UML-Klassendiagramme genutzt. Diese dienen dazu, die Struktur von Systemen anhand deren statischen Eigenschaften und Beziehungen zueinander abzubilden (Jeckle et al. (2005), S. 95).

In Abbildung 3.4 ist beispielhaft die Definition und Anwendung des Profils *myProfile* dargestellt. Auf der linken Seite wird das Profil mit einem Stereotyp *Student* definiert und auf der rechten Seite durch die Auszeichnung der Klasse *MaxMuster* mit dem definierten Stereotyp angewendet. Durch die «extends»-Assoziation wird festgelegt, welches UML-Element von dem Stereotyp erweitert wird. Dadurch wird auch bestimmt, auf welche Modellelemente der Stereotyp angewendet werden darf. Angewandte Stereotypen werden, wie im Beispiel bei der Klasse *MaxMuster* dargestellt, oberhalb des Klassennamens innerhalb französischer Guillemets benannt (Jeckle et al. (2005), S. 533).

Stereotypen können Attribute besitzen (im Beispiel sind es die Attribute *Name* und *immatrikuliert*), welche bei Anwendung des Stereotyps mit Werten belegt werden können und als Eigenschaftswerte (Tagged Values) erscheinen (im Beispiel *Max Muster* und *immatrikuliert* bei der Klasse *MaxMuster*). Durch Constraints können innerhalb von Profilen Beschränkungen für die Modellelemente definiert werden. Dies kann, wie im Beispiel, als Notiz mit einem Element assoziiert sein oder direkt in der Klasse angegeben werden (Petrasch und Meimberg (2006), S. 70f).

3.4 Das Web-Content-Management-System OpenCms

OpenCms ist ein Open-Source Web-Content-Management-System welches in der Version 7.0.5 vorliegt und von der Alkacon Software GmbH federführend entwickelt wird (Alkacon Software GmbH (2007b)). OpenCms basiert auf der Programmiersprache Java und nutzt die Extensible-Markup-Language zur Strukturierung des Content sowie für die Konfiguration des Systems (Alkacon Software GmbH (2007b); Alkacon Software GmbH (2007a)). OpenCms wird als Java-Servlet in einem Java-Servlet-Container (wie beispielsweise dem von der Apache Software Foundation entwickelten Apache Tomcat) ausgeführt (Butcher (2006), S. 13). Die Benutzeroberfläche von OpenCms ist über einen Browser erreichbar (Alkacon Software GmbH (2009)). Im diesem Kapitel wird auf die technische Realisierung der in Abschnitt 2.3.1 beschriebenen Grundmodule eines Web-Content-Management-System anhand OpenCms eingegangen.

3.4.1 Content-Repository

OpenCms nutzt die Java Database Connectivity (JDBC), um die Verbindung zu einer relationalen Datenbank herzustellen, in welcher der Content sowie weitere OpenCms-spezifische Daten gespeichert werden (Butcher (2006), S. 13f). Es werden verschiedene Datenbanken wie MySQL, PostgreSQL oder Oracle unterstützt (Butcher (2006), S. 13). Der Content wird innerhalb von XML-Dateien strukturiert in der Datenbank abgelegt (Butcher (2006), S. 14).

OpenCms ermöglicht die Definition von Content entsprechend dem in Abschnitt 2.2.1 definierten Content im engeren Sinn. Dabei können OpenCms-spezifische als auch selbst definierte Inhaltstypen genutzt werden. Die Definition der Struktur des Content erfolgt innerhalb von XML Schema De-

finitionen (XSD) (siehe Abschnitt 3.1.1). Der Benutzer arbeitet innerhalb des WCMS mit dem Content wie in einem Dateisystem, dem so genannten Virtual File System (VFS) (Butcher (2006), S. 44). Während des Publikationsprozesses werden aus dem in der Datenbank abgelegten Content die HTML-Dokumente generiert und zusammen mit den Mediendaten in das Dateisystem geschrieben.

3.4.2 Berechtigungsverwaltung

Die Benutzerverwaltung von OpenCms erfolgt über die Administrationsoberfläche. Dabei können neben Benutzern auch Gruppen und Webbenutzer verwaltet werden (Butcher (2006), S. 97). Benutzer können auf das Web-Content-Management-System zugreifen und entsprechend den ihnen zugewiesenen Rechten Änderungen am System und am Content vornehmen. Gruppen ermöglichen neben der Aggregation von Benutzern und Webbenutzern auch die Möglichkeit die Gruppe als Rolle für die Workflowsteuerung zu nutzen (Butcher (2006), S. 106). Unter *Webbenutzern* werden Benutzer verstanden, die Zugriff auf einen bestimmten geschützten Bereich der Web-Content-Management-Anwendung, nicht aber auf das Web-Content-Management-System haben (Butcher (2006), S. 98).

3.4.3 Workflowsteuerung

Auch Workflows werden über die Administrationsoberfläche verwaltet (Butcher (2006), S. 182). Einzelne Aufgaben können dabei entweder einzelnen Benutzern oder Rollen von Benutzern zugewiesen werden (Butcher (2006), S. 184). Von Seiten der Workflowsteuerung besteht dabei eine Beschränkung auf einen einstufigen Workflowprozess. Mehrstufige Workflowprozesse sind nur durch die Weiterleitung einer Aufgabe an den nächsten Benutzer oder die nächste Rolle realisierbar und können nicht vom Initiator des Workflows

festgelegt werden (Butcher (2006), S. 188). Eine weitere Einschränkung besteht darin, dass einem Workflow keine Elemente des VFS zugewiesen werden können, wodurch ein schneller Zugriff möglich wäre (Butcher (2006), S. 190).

3.4.4 Import- und Exportverwaltung

Der Import und Export von Content kann in OpenCms über die Administrationsoberfläche unter dem Punkt Datenbankverwaltung geschehen (Butcher (2006), S. 114). Hierbei kann zwischen einem Export der gesamten Datenbank und bestimmten Ordnern des VFS gewählt werden (Butcher (2006), S. 115). Es können auch nur die Daten exportiert werden, die nach einem bestimmten Datum geändert wurden (Butcher (2006), S. 116). Der Content wird als XML-Dateien, wie er in der Datenbank abgelegt ist, exportiert (Butcher (2006), S. 119). Des weiteren besteht die Möglichkeit, die HTML-Dokumente und Mediendaten der CMA zu exportieren (Butcher (2006), S. 119). Ein Import oder Export des Contents in ein anderes als das proprietäre Format von OpenCms wird nicht unterstützt (Butcher (2006), S. 114ff).

3.4.5 Templateverwaltung

Die Templates in OpenCms sind Java-Server-Pages (JSP). Innerhalb dieser kann mittels JSP-Tags oder JSP-Scriptlets auf die Inhaltswerte des Content zugegriffen werden (Butcher (2006), S. 196). JSP-Tags sind in Tag-Libraries definierte Code-Elemente. OpenCms bietet eine Tag-Library als eine einfache und schnelle Möglichkeit, um über bestimmte Content-Elemente zu iterieren und Inhaltswerte HTML-konform auszugeben (Butcher (2006), S. 197). Darüber hinaus bietet OpenCms ein Application-Programming-Interface (API), um innerhalb von JSP-Scriptlets oder eigenen Java-Klassen auf OpenCms-Objekte zugreifen zu können (Butcher (2004), S. 100). Die Java-Server-Pages enthalten neben den JSP-Tags und JSP-Scriptlets die

HTML-Tags für die statischen Elemente des HTML-Dokuments (bei anderen Ausgabeformaten entsprechend andere Tags) (Butcher (2006), S. 196). Der HTML-Code kann mit Javascript-Code sowie Cascading-Style-Sheet (CSS) Daten angereichert sein.

3.4.6 Hyperlinkverwaltung

Die Hyperlinkverwaltung von OpenCms bietet seit der Version 7.0 eine so genannte Content-Relationship-Engine zur Verwaltung der Beziehungen zwischen Content-Elementen. Hierbei wird zwischen den starken und schwachen Verlinkungen von Content unterschieden. Die starken Verlinkungen werden genutzt, wenn das verlinkte Content-Element für das verweisende Content-Element essentiell benötigt wird. Schwache Verlinkungen dienen dagegen dem Verweis auf Content-Elemente, welche nicht von essentieller Wichtigkeit für das verweisende Content-Element sind (Liliedahl (2008), S. 68). Für die Erstellung der Navigation kann die API von OpenCms genutzt werden (Butcher (2004), S.100).

Kapitel 4

Modellgetriebenes Customizing von Web-Content-Management-Systemen

4.1 Modelle für das Customizing eines Web-Content-Management-Systems

Im Folgenden wird ein plattformunabhängiges Metamodell zur Modellierung einer Web-Content-Management-Anwendung vorgestellt. Dabei werden die in Abschnitt 3.1.2 beschriebenen Besonderheiten von Web-Anwendungen sowie die in Abschnitt 2.2.1 betrachtete Zusammensetzung von Content beachtet. Im Gegensatz zu bestehenden Modellierungsmethoden, welche einen so genannten „Grüne-Wiese-Ansatz“ verfolgen (Ehlers (2003), S. 192), werden die Funktionalitäten der in Abschnitt 2.3 vorgestellten Grundmodule eines Web-Content-Management-System durch deren Customizing-Möglichkeiten (siehe Abschnitt 2.3.2) berücksichtigt. So kann die Erstellung einzelner Dokumente und die Eingabe von Content direkt im WCMS erfolgen und muss nicht modelliert werden (Ehlers (2003), S. 192).

Das Metamodell wird entsprechend den meisten Modellierungsmethoden für Web-Anwendungen (siehe Abschnitt 3.1.2) in die Ebenen Content, Navigation und Präsentation unterteilt. Aus der Einbeziehung der für das Meta-

modell relevanten Grundmodule (siehe Abschnitt 2.3.2) der Web-Content-Management-Systeme ergibt sich die Anforderung, diese Ebenen um die Ebene der Berechtigungsverwaltung zu ergänzen. Die weiteren relevanten Grundmodule Content-Repository, Hyperlinkverwaltung und Templateverwaltung können durch die Ebenen Content, Navigation und Präsentation angepasst werden.

Für das Metamodell wird die Unified Modeling Language genutzt. Diese wird durch die Erstellung eines UML-Profils als Hostsprache für eine interne domänenspezifische Sprache, für die Domäne der Web-Content-Management-Systeme, genutzt.

In der Abbildung 4.1 ist schematisch ein Softwareentwicklungsprozess für die Erstellung einer Web-Content-Management-Anwendung und damit einhergehend das Customizing eines Web-Content-Management-Systems nach den Vorgaben der Model Driven Architecture dargestellt.

Im ersten Schritt werden innerhalb der Phase der Problemdefinition durch die Stakeholder die funktionalen, qualitativen, system- und prozessbezogenen Anforderungen an das WCMS beschrieben (Dumke (2003), S. 25). Unter *Stakeholder* sind Personen oder Organisationen zu verstehen, die direkt oder indirekt Einfluss auf die Anforderungen an das WCMS haben (Stoyan (2007), S. 11). Das Ergebnis der ersten Phase ist das berechnungsunabhängige Modell (CIM).

Dieses wird innerhalb der Anforderungsanalyse und Spezifikation durch den Fachdesigner auf Korrektheit, Vollständigkeit, Sachgerechtheit, Konsistenz und Machbarkeit überprüft und Änderungen sowie Detaillierungen vorgenommen (Dumke (2003), S. 36). Die Anforderungen werden innerhalb dieser Phase mittels UML-Modellen formalisiert. Grundlage hierfür ist das technik- und herstellerunabhängige Metamodell des Platform Independent Model, welches auch als Architektur-Metamodell bezeichnet werden kann (Gruhn et al. (2006), S. 120f).

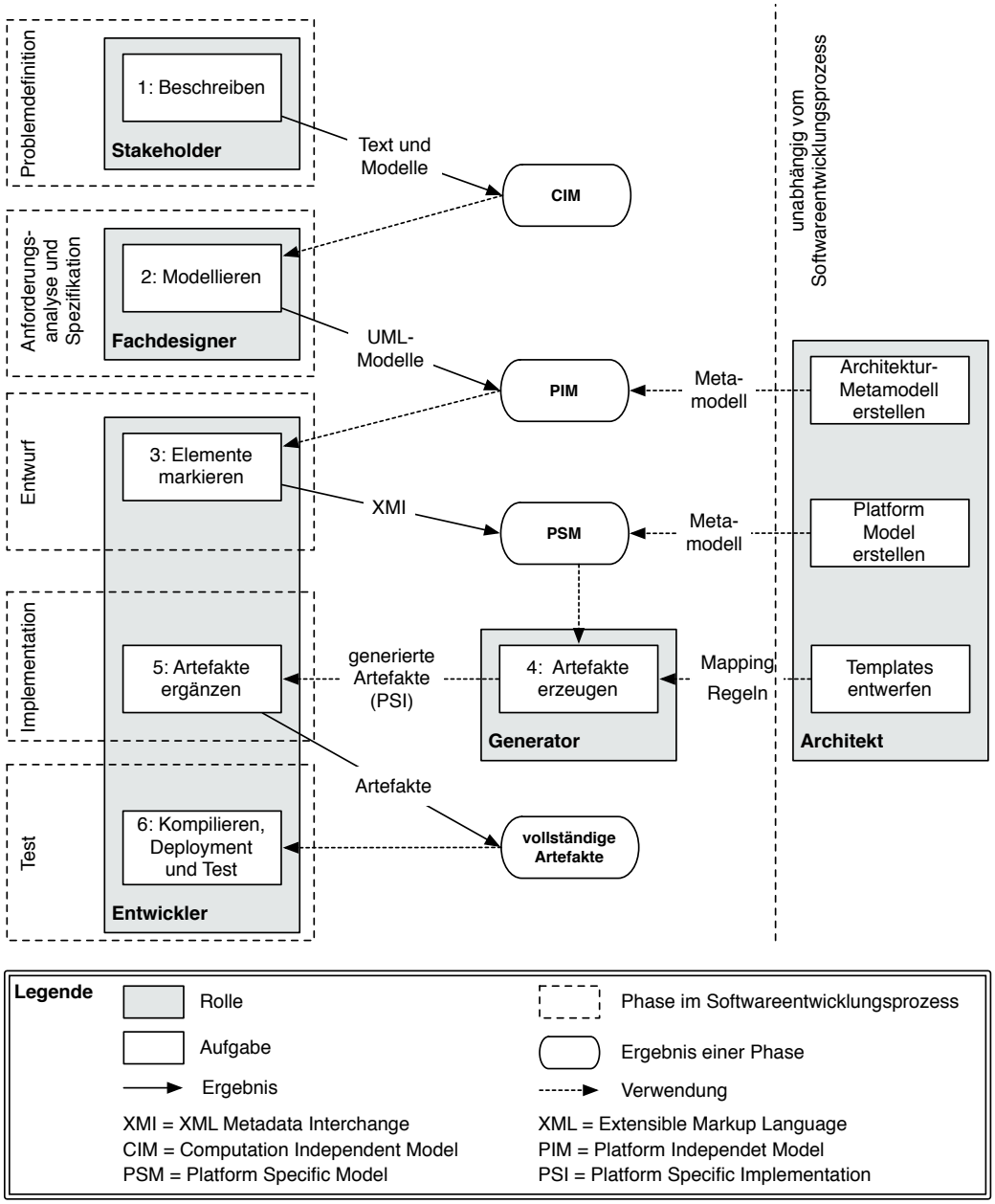

Abbildung 4.1: Modellgetriebener Softwareentwicklungsprozess (Starke (2008), S. 303; Urbainczyk (2005), S. 42; Gruhn et al. (2006), S. 120)

Innerhalb des Architektur-Metamodells werden die Zusammenhänge der architekturellen Elemente wie Konzepte, Muster und Schnittstellen einer oder mehrerer horizontalen Domänen festgehalten (Gruhn et al. (2006), S.

121). Eine horizontale Domäne beschreibt dabei einen meist technologischen Querschnittsaspekt eines Softwaresystems (z. B. Sicherheit oder Benutzeroberfläche) (Gruhn et al. (2006), S. 129). Das für das Customizing zu verwendende Architektur-Metamodell für WCMS setzt sich zusammen aus den Domänen Content, Präsentation, Navigation und Berechtigungsverwaltung. Architektur-Metamodelle können durch UML-Profile realisiert werden (Gruhn et al. (2006), S. 130). Das Ergebnis der zweiten Phase sind die plattformunabhängigen Modelle (PIM).

Innerhalb der Phase des Entwurfs werden im Anschluss diese plattformunabhängigen Modelle durch Markierungen um systembezogene Angaben ergänzt (Dumke (2003), S. 54) und somit in plattformspezifische Modelle überführt. Das hierfür genutzte Metamodell (in Form eines weiteren UML-Profils) basiert auf den Konzepten der Plattform (z. B. OpenCms), welche innerhalb des Platformmodel modelliert sind (Gruhn et al. (2006), S. 140). Die Modelle werden daraufhin nach dem XML-Metadata-Interchange Schema in ein XML-Dokument serialisiert, um vom Generator unter Beachtung der plattformspezifischen Transformationsvorschriften (Mapping-Regeln) zur Erzeugung von Artefakten genutzt zu werden. Diese Artefakte werden in der Implementations-Phase durch nicht-generierten Code ergänzt, um danach innerhalb der Test-Phase den erforderlichen Tests unterzogen zu werden.

Die hier beschriebene Methode zur Transformation der plattformunabhängigen in die plattformspezifischen Modelle entspricht der in Abschnitt 3.3.2 vorgestellten Transformation mittels Profilen.

Die Erstellung der Metamodelle der plattformunabhängigen und plattformabhängigen Modelle sowie der Transformationsregeln kann, wie in Abbildung 4.1 durch die senkrechte gestrichelte Linie dargestellt, unabhängig von dem Softwareentwicklungsprozess und durch unterschiedliche Personen erfolgen (Petrasch und Meimberg (2006), S. 152).

Im Folgenden wird zunächst das Architektur-Metamodell, welches für die Erstellung des Platform Independent Model benötigt wird, vorgestellt. Dieses

teilt sich in die bereits genannten Ebenen Content, Navigation, Präsentation und Berechtigungsverwaltung. Die in den Abbildungen vorhandenen Assoziationen zwischen Elementen (Aggregations-Assoziationen, gerichtete Assoziationen und ungerichtete Assoziationen) sind nicht Teil des UML-Profils, sondern dienen nur der besseren Verständlichkeit des Metamodells. Innerhalb der Beschreibung des Metamodells werden die definierten Stereotypen in Guillemets (z. B. ≪wcmsObject≫) und Attribute kursiv (z. B. *description*) geschrieben.

4.1.1 Metamodell der plattformunabhängigen Modelle

Bevor die Ebene des Content näher betrachtet wird, werden zunächst die gemeinsamen Elemente der vier Ebenen vorgestellt. Diese sind in Abbildung 4.2 dargestellt. Die gemeinsame Metaklasse aller Stereotypen ist die *Class*. Sie wird durch den abstrakten Stereotyp ≪wcmsObject≫ erweitert, welcher als Attribut eine Beschreibung (*description*) des Elements haben kann. Dieser Stereotyp dient als oberstes Element der Stereotyp-Hierarchie als Generalisierung aller von der Metaklasse *Class* abgeleiteten Stereotypen des Profils. Änderungen an diesem Stereotyp vererben sich auf alle anderen Stereotypen. Dies kann bei einer Anpassung oder Erweiterung des Profils genutzt werden.

Der abstrakte Stereotyp ≪wcmsElement≫ dient der Gruppierung von den ebenfalls abstrakten Stereotypen ≪orderedElement≫, ≪elementWithCardinality≫ und ≪hierarchicalElement≫. Der Stereotyp ≪elementWithCardinality≫ wird durch den Stereotyp ≪createableElement≫ spezialisiert. Diese Stereotypen bilden jeweils eine bestimmte Eigenschaft ab, die durch die Spezialisierung an andere Stereotypen vererbt wird. Die Eigenschaften sind im Folgenden kurz beschrieben:

≪**orderedElement**≫**:** Ermöglicht durch die Attribute *isFirst* und *nextElement* die Definition einer Reihenfolge von Elementen.

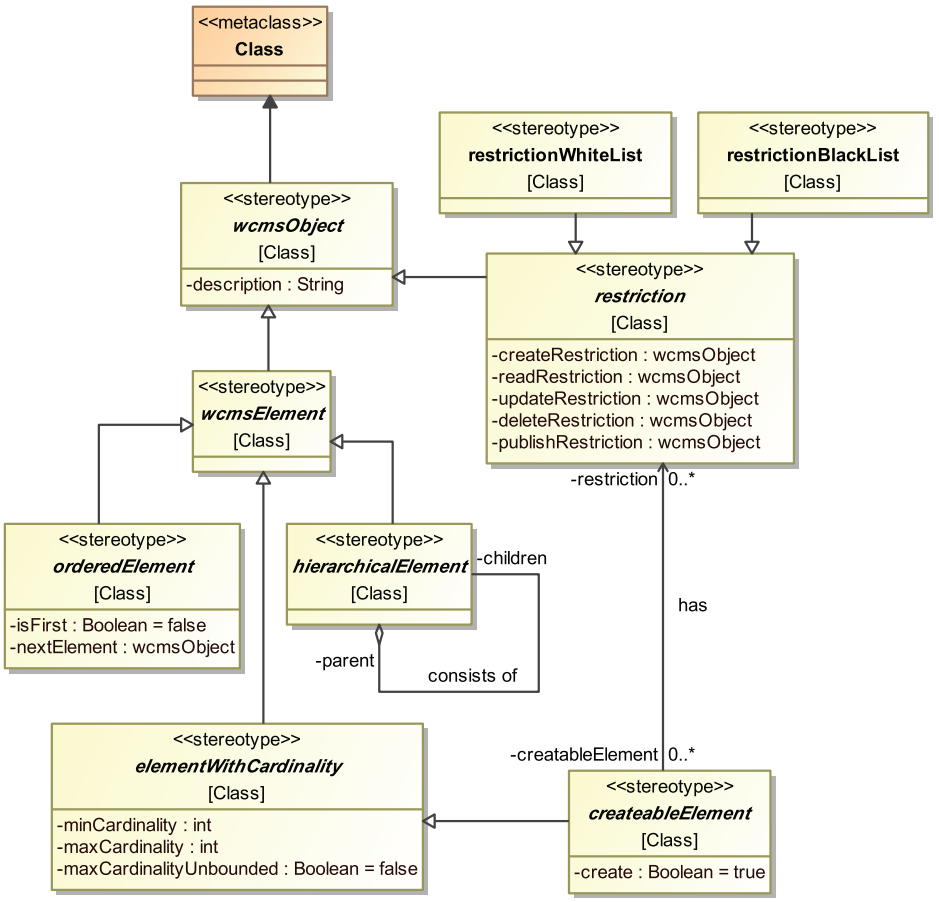

Abbildung 4.2: Gemeinsame Elemente der vier Ebenen des plattformunabhängigen Metamodell

«**elementWithCardinality**»**:** Kann genutzt werden, um die Kardinalität von Elementen durch die Attribute *minCardinality* und *maxCardinality* vom Typ *int* zu definieren. Durch das Attribut *maxCardinalityUnbounded* vom Typ *boolean* kann anegegeben werden, dass eine unbeschränkte Anzahl dieser Elemente existieren kann. Die Kardinalitätsbeschränkungen können auch an den Assoziationsenden angegeben werden, falls eine Klasse in mehreren Assoziationen mit unterschiedlichen Kardinalitäten genutzt werden soll.

«**createableElement**»**:** Dient als Generalisierung von Elementen (Stereotypen), welche innerhalb des Web-Content-Management-System von

einem autorisierten Benutzer angelegt, gelesen, geändert, gelöscht und publiziert werden können (Content, Seiten und Autoren- Navigationspunkte). Durch das Attribut *created* vom Typ *boolean* kann angegeben werden, ob das Element angelegt werden soll oder nur angelegt werden kann.

«**hierarchicalElement**»**:** Dient als Generalisierung der Elemente (Stereotypen), die eine Aggregations-Beziehung zu Elementen vom gleichen Typ haben und dadurch eine Hierarchie bilden können (Navigationspunkte und Seitenbereiche).

Der abstrakte Stereotyp «restriction» dient als Generalisierung der Stereotypen «restrictionWhiteList» und «restrictionBlackList». Diese erben von diesem Attribute für die Erstellung von Einschränkungen (Schreiben, Lesen, Löschen, Ändern und Publizieren), welche aufgrund ihres Parametertyps *wcmsObject* alle Stereotypen aufnehmen können. Durch die gerichtete Assoziation von dem Stereotyp «createableElement» zu «restriction» beschreibt die mögliche Verwendung einer Beschränkung an einem von «createableElement» spezialisierten Element. Dabei können zu einem «createableElement» mehrere Einschränkungen existieren und einer Einschränkung mehrere «createableElement» zugewiesen werden (Kardinalitätsbeschränkungen von *0..**).

Durch «restrictionWhiteList» werden Elemente angegeben, welche die möglichen Operationen ausführen dürfen, während die mittels einer «restrictionBlackList» zugewiesenen Elemente dies nicht dürfen.

Das plattformunabhängige Metamodell der Contentebene

Auf der in Abbildung 4.3 dargestellten Ebene des Content des Metamodells wird die Struktur der verschiedenen Content-Typen gemäß der Definition von Content im engeren Sinn (in Abschnitt 2.2.1) modelliert. Das Metamo-

dell geht durch die Einbeziehung der Inhaltstypen des Content über die in Abschnitt 3.1.2 beschriebene Modellierung der Struktur des Content (bei der Modellierung von Web-Anwendungen) hinaus.

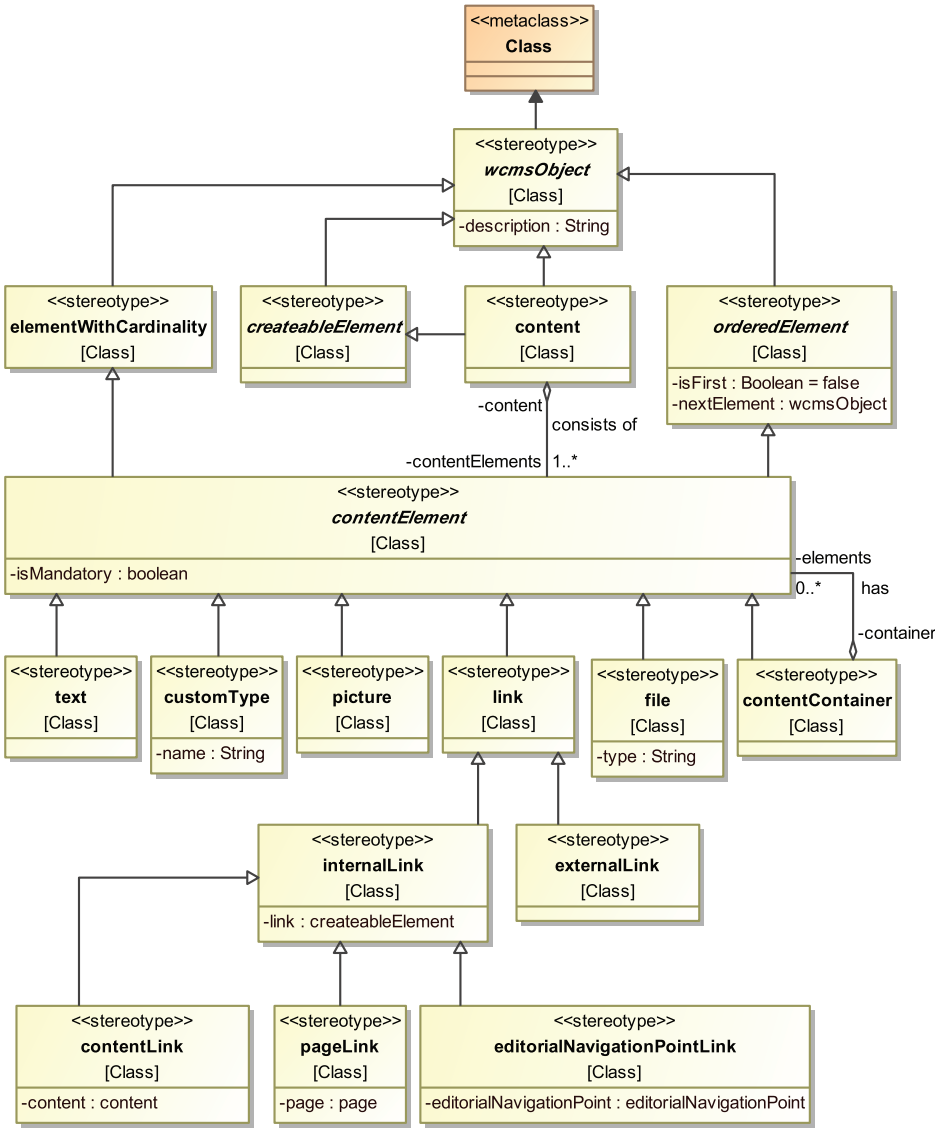

Abbildung 4.3: Plattformunabhängiges Metamodell der Contentebene

Content setzt sich, wie in Abschnitt 2.2.1 beschrieben, aus Inhaltstupeln (bestehend aus Inhaltsattribut, Inhaltstyp und Inhaltswert) sowie möglichen weiteren Content-Elementen zusammen. Der Stereotyp «content» dient der Modellierung der verschiedenen Typen von Content. Durch ihn kann der Na-

me des Content (durch den Klassennamen der Klasse, auf den der Stereotyp angewendet wird) und die von «wcmsObject» an alle Stereotypen vererbte Beschreibung angegeben werden. Wie schon im letzten Abschnitt erwähnt ist «content» eine Spezialisierung von «createableElement». Die Zusammensetzung von Content aus Inhaltstupeln und weiteren Content-Elementen ist durch die Aggregations-Assoziation zwischen «content» und dem abstrakten Stereotypen «contentElement» modelliert.

Der abstrakte Stereotyp «contentElement» dient als Generalisierung der möglichen, den Inhaltstyp bestimmenden Stereotypen «text», «customType», «picture», «file», «contentContainer» sowie den verschiedenen Arten von Links, die unter dem Stereotyp «link» generalisiert sind. Von «contentElement» erben alle von ihm abgeleiteten Stereotypen das Attribut *isMandatory*, vom Typ *boolean*. Dieses gibt an, ob das Element bei der Erstellung von neuem Content zwingend erforderlich ist. Ausserdem vererbt «contentElement» durch die Spezialisierungen von «orderedElement» und «elementWithCardinality» die Eigenschaft, die Reihenfolge von Elementen festzulegen und Kardinalitätsbeschränkungen anzugeben. Durch die von «contentElement» spezialisierten Stereotypen können die Inhaltstupel modelliert werden. Die Klassennamen der mittels dieser Stereotypen ausgezeichneten Klassen bilden dabei das Inhaltsattribut. Der Inhaltstyp wird durch die einzelnen Stereotypen bestimmt (verschiedene Links, Datei, Bild, Text und mögliche eigene Typen). Durch die Spezialisierungen des Stereotypen «link» können Verlinkungen zu Content («contentLink»), Seiten («pageLink»), Navigationspunkten («editorialNavigationPointLink»), internen Quellen («internalLink») und externen Quellen («externalLink») modelliert werden. Der ebenfalls von «contentElement» abgeleitete Stereotyp «contentContainer» dient der Aggregation von «contentElement»-Elementen. Die Stereotypen «contentLink», «pageLink» und «editorialNavigationLink» sind unter dem abstrakten Stereotyp «internalLink» als interne Verlinkungen generalisiert.

Das plattformunabhängige Metamodell der Navigationsebene

Durch die in Abbildung 4.4 dargestellte Ebene der Navigation des Metamodells werden die Navigationsstrukturen der Web-Content-Management-Anwendung modelliert. Dadurch wird das in Abschnitt 2.3.1 beschriebene Grundmodul der Hyperlinkverwaltung angepasst. Ausserdem wird auf dieser Ebene die Autoren-Navigationsstruktur für die Verwaltung des Content und der Seiten im Web-Content-Management-System modelliert. Eine WCMA kann verschiedene Navigationsstrukturen besitzen, welche die innerhalb der Autoren-Navigationsstruktur angelegten Seiten unterschiedlich strukturieren. Die bei der Modellierung von Web-Anwendungen vorgenommene Trennung der Modelle der Navigationsebene in Hypertext-Strukturmodell und Zugriffsmodell (siehe Abschnitt 3.1.2) wird nicht übernommen. Die im Zugriffsmodell definierten Zugriffsstrukturen können innerhalb der Präsentationsebene unter Nutzung des Content-Selektor modelliert werden.

Durch den abstrakten Stereotyp «navigationPoint» werden durch die Spezialisierungen von den Stereotypen «orderedElement» und «hierarchicalElement» die Eigenschaften vererbt, eine Hierarchie zu modellieren, bei welcher alle Kindelemente eines Elements eine bestimmte Reihenfolge besitzen können. Der Stereotyp «navigationPoint» hat die beiden Spezialisierungen «editorialNavigationPoint» und «presentationNavigationPoint». Durch Klassen mit diesen beiden Stereotypen werden die Autoren-Navigationspunkte des Web-Content-Management-System beziehungsweise die Navigationspunkte der Web-Content-Management-Anwendung modelliert.

Bei Klassen, die mit dem Stereotyp «editorialNavigationPoint» ausgezeichnet sind, kann durch eine gerichtete Assoziation zu dem Stereotyp «createableElement» modelliert werden, wo in der Autoren-Navigationsstruktur «createableElement»-Spezialisierungen (Content, Seiten und Autoren-Navigationspunkte) angelegt werden können. Dadurch, dass «createable-

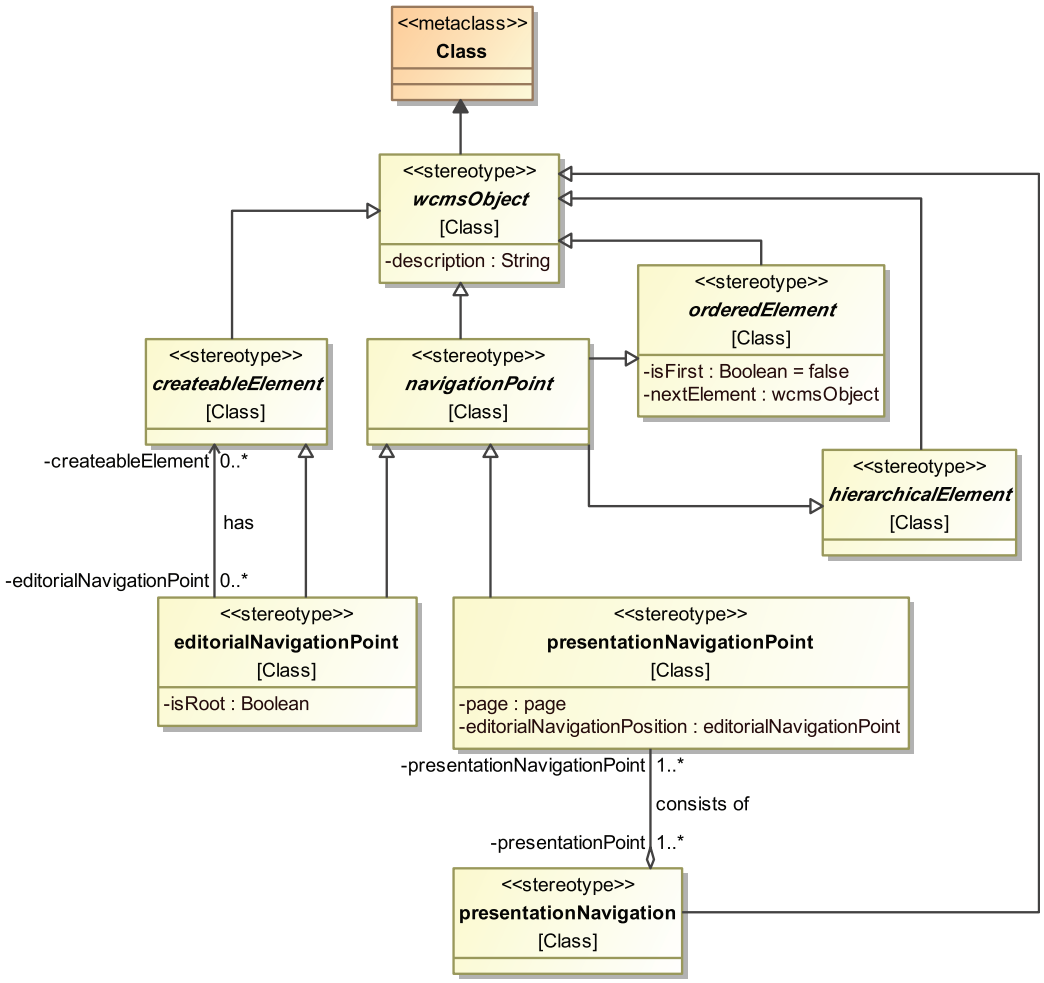

Abbildung 4.4: Plattformunabhängiges Metamodell der Navigationsebene

Element» eine Spezialisierung von «elementWithCardinality» ist, können Kardinalitätsbeschränkungen angegeben werden (z. B. darf nur eine Seite *Homepage* existieren). Durch das Attribut *isRoot* vom Typ *Boolean* wird angegeben, ob dieser Autoren-Navigationspunkt der Ursprungspunkt ist. Durch die Nutzung der für «createableElement» angegebenen gerichteten Assoziation zu «restriction» kann festgelegt werden, welche Benutzergruppen und Benutzerrollen bestimmte Zugriffsrechte auf den Autoren-Navigationspunkt haben.

Beim Stereotyp «presentationNavigationPoint» kann durch das Attribut *page* vom Typ *page* der Seitentyp angegeben werden, der durch den Navigationspunkt erreichbar ist. Das Attribut *editorialNavigationPosition* vom Typ *editorialNavigationPoint* dient der Angabe der Position der Seite innerhalb der Autoren-Navigationsstruktur. Der «presentationNavigation»-Stereotyp hat eine Aggregations-Assoziation zu «presentationNavigationPoint» und dient als Ursprungselement der einzelnen Navigationsstrukturen. Durch Klassen der beiden Stereotypen «presentationNavigation» und «presentationNavigationPoint» können verschiedene Navigationsstrukturen für die WCMA modelliert werden, wobei jeweils auf Seitentypen innerhalb der Autoren-Navigationsstruktur verwiesen wird. Die Autoren-Navigationsstruktur des WCMS kann somit innerhalb der WCMA durch unterschiedliche Navigationsmöglichkeiten repräsentiert werden.

Das plattformunabhängige Metamodell der Präsentationsebene

Die in Abbildung 4.5 dargestellte Ebene der Präsentation des Metamodells ermöglicht die Modellierung verschiedener Seitentypen. Dies bildet die Grundlage für die Erstellung der in Abschnitt 2.3.1 beschriebenen Templates des Grundmoduls Templateverwaltung. Modelliert wird dabei nur die Struktur der späteren Templates, die layoutspezifischen Informationen werden nicht modelliert. Die Modelle der Präsentationsebene sind dadurch im Gegensatz zu den Templates noch nicht medienspezifisch. Wie in Abschnitt 2.3.1 erwähnt, wird innerhalb der Templates auf den Content und die einzelnen Inhaltstupel zugegriffen und die Inhaltswerte ausgegeben. Für die Modellierung der Präsentationsebene von Web-Anwendungen wurden in Abschnitt 3.1.2 die Elementtypen Präsentationsseite, -einheit und -element vorgestellt. Diese Aufteilung wird durch die Stereotypen «page», «pageUnit» und «pageElement» abgebildet.

Der Stereotyp «page», der eine Spezialisierung von «wcmsObject» und «createableElement» ist, kann genutzt werden, um Seitentypen zu definie-

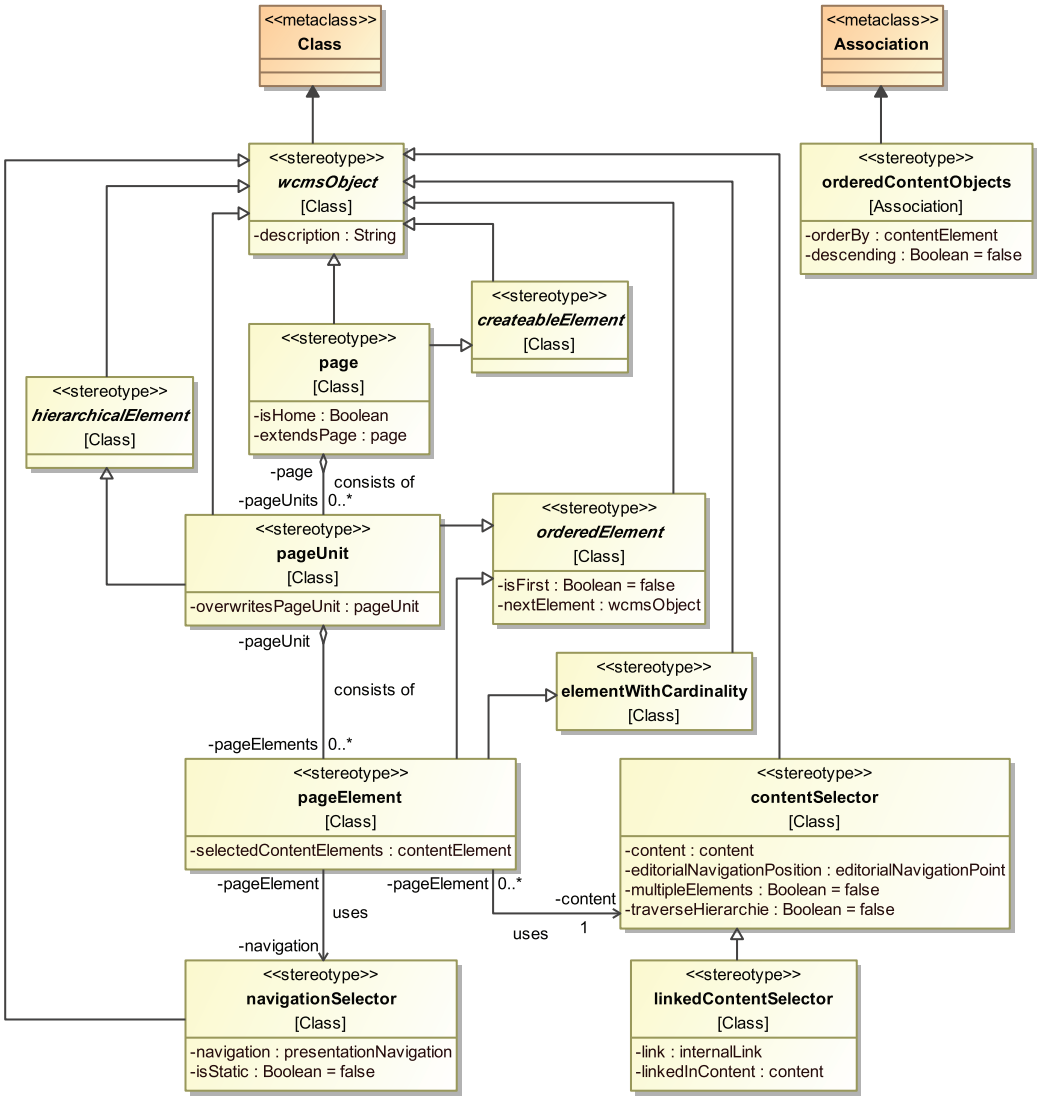

Abbildung 4.5: Das plattformunabhängiges Metamodell der Präsentationsebene

ren. Das Attribut *isHome* vom Typ *Boolean* dient dazu, den Seitentypen der Startseite auszuzeichnen. Durch das Attribut *extendsPage* vom Typ *page* kann angegeben werden, ob diese Seite auf einer bestehenden Seite aufbaut und nur bestimmte Bereiche dieser Seite überschreibt. Klassen mit dem Stereotyp «page» bestehen aus Klassen vom Stereotyp «pageUnit».

Logisch zusammengehörende Elemente eines Seitentyps werden durch Klassen des Stereotyp «pageUnit» zu einem Bereich zusammengefasst. Dieser spezialisiert «orderedElement» und «hierarchicalElement» und kann somit für die Modellierung von Hierarchien mit geordneten Elementen genutzt werden. Durch das Attribut *overwritesPageUnit* kann angegeben werden, welcher Bereich einer anderen Seite, die im Stereotyp «page» als *extendsPage* angegeben wurde, von dieser «pageUnit» überschrieben werden soll. Klassen des Stereotyp «pageUnit» setzen sich aus Klassen des Stereotyp «pageElement» zusammen.

Die Klassen, die mit dem Stereotyp «pageElement» ausgezeichnet sind, erben von «orderedElement» die Eigenschaft, die Reihenfolge der Elemente festzulegen. Durch die gerichteten Assoziationen zu den Stereotypen «contentSelector» und «navigationSelector» kann ein Seitenelement auf Content oder eine Navigationsstruktur (vom Stereotyp «presentationNavigation») zugreifen. Da ein «contentSelector» von mehreren «pageElement»-Klassen genutzt werden kann, erfolgt die Auswahl der Inhaltstupel des Content durch das Attribut *selectedContentElements* innerhalb von «pageElement». Durch die Spezialisierung von «elementWithCardinality» kann durch Kardinalitätsbeschränkungen festgelegt werden, wie viele «pageElement»-Elemente innerhalb eines Seitenbereichs («pageUnit») genutzt werden sollen. Die Kardinalitätsbeschränkung ist wie das Attribut *selectedContentElements* nur bei der Nutzung eines «contentSelector» relevant.

Klassen, die mit «contentSelector» oder dessen Spezialisierung «linkedContentSelector» ausgezeichnet sind, werden zur Auswahl von Content genutzt, dessen Inhaltswerte auf der Seite angezeigt werden sollen. Durch die Attribute *content* und *editorialNavigationPosition* kann bei Klassen mit dem Stereotyp «contentSelector» der Content ausgewählt und dessen Position innerhalb der Autoren-Navigationsstruktur angegeben werden. Die Attribute *multipleElements* und *traverseHierarchie* vom Typ *Boolean* geben an, ob mehrere Content-Elemente genutzt werden sollen und ob die Autoren-Navigationsstruktur ab dem angegebenen Autoren-Navigationspunkt tra-

versiert werden soll. Falls der zu nutzende Content innerhalb eines anderen Content verlinkt ist, kann durch Klassen des Stereotyp «linkedContentSelector» dieser Link (durch das Attribut *link*) und der Content (durch das Attribut *linkedInContent*), in dem der Link existiert, ausgewählt werden.

Der Stereotyp «orderedContentObjects» erweitert die Metaklasse *Association* und dient der Angabe eines Elements zur Sortierung (Attribut *orderBy* des Typs *contentElement*) und einer Sortierungsrichtung (Attribut *descending* vom Typ *Boolean*). Der Stereotyp kann auf gerichtete Assoziationen von Klassen des Stereotyp «pageElement» zu «contentSelector» angewendet werden, falls mehrere Elemente eines Content-Typ (Attribut *multipleElements* = true) selektiert werden.

Durch die Klassen des Stereotyp «navigationSelector» werden Navigationsstrukturen der WCMA vom Stereotyp «presentationNavigation» ausgewählt, um innerhalb der Seite angezeigt zu werden. Das Attribut *isStatic* kann genutzt werden, um anzugeben, ob die Navigationsstruktur unverändert bleibt oder der aktuellen Navigationsposition des Benutzers angepasst werden soll.

Das plattformunabhängige Metamodell der Berechtigungsverwaltungs-Ebene

Die Ebene der Berechtigungsverwaltung, deren Metamodell in Abbildung 4.6 dargestellt ist, dient der Modellierung des in Abschnitt 2.3.1 beschriebenen Grundmoduls der Berechtigungsverwaltung. Entsprechend der dortigen Beschreibung können die einzelnen Benutzer hierbei Benutzergruppen und Benutzerrollen zugeordnet werden, welche dann für die Verwaltung der Rechte genutzt werden. Benutzergruppen können dabei Hierarchien bilden.

Der Stereotyp «party» dient als Generalisierung der Stereotypen «userGroup», «userRole» und «user». Die Benutzergruppen und Benutzerrollen, die durch Klassen mit den Stereotypen «userGroup» und «userRole» mo-

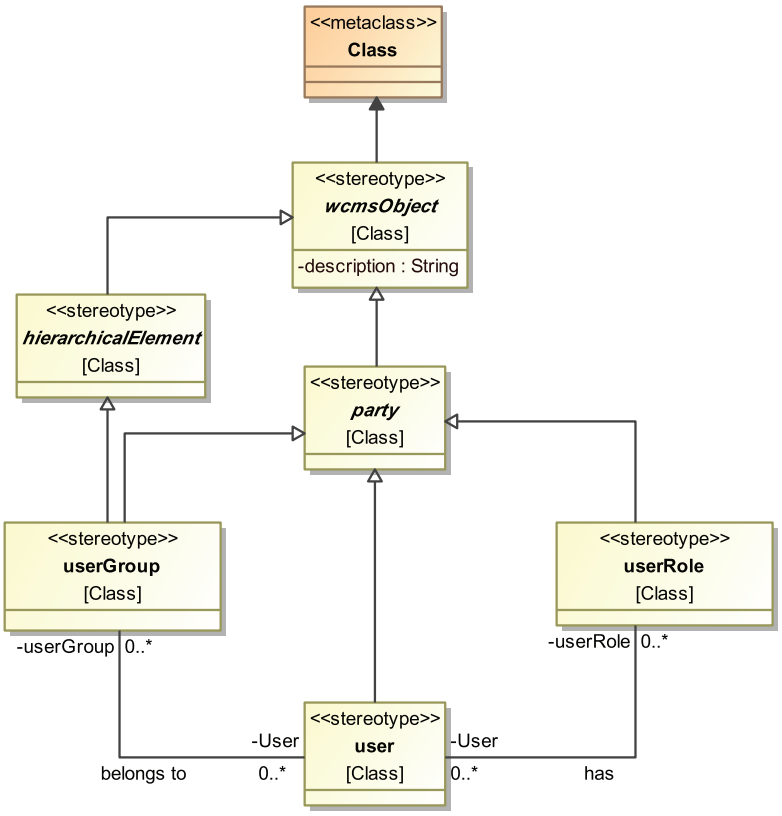

Abbildung 4.6: Das plattformunabhängige Metamodell der Berechtigungsverwaltungsebene

delliert werden, können innerhalb von Beschränkungen (Spezialisierungen von «restriction») genutzt werden, um ihnen bestimmte Rechte zuzuweisen.

Mit dem Stereotyp «userGroup» ausgezeichnete Klassen können durch die Spezialisierung von dem Stereotyp «hierarchicalElement» zur Bildung von Hierarchien genutzt werden. Durch Klassen des Stereotyp «userRole» können die Rollen modelliert werden. Den durch Klassen des Stereotyp «user» modellierten Benutzern können durch Assoziationen Benutzergruppen und Benutzerrollen zugeordnet werden.

4.1.2 plattformabhängige Modelle

Ein mittels des in Abschnitt 4.1.1 vorgestellten plattformunabhängigen Metamodells erstelltes Modell wird, wie im Softwareentwicklungsprozess zu Anfang des Abschnitts 4.1 beschrieben, in der folgenden Phase der Softwareentwicklung in ein plattformspezifisches Modell transformiert. Diese Transformation kann, wie in Abschnitt 3.3.2 beschrieben, auf der Basis von Model-Type und Model-Instance Mappings durch einen Transformator unterstützt werden. Die für die Model-Instance Mappings erforderlichen Markierungen sind von dem Fachdesigner an dem plattformunabhängigen Modell vorzunehmen. Im folgenden Abschnitt dienen diese beiden Arten von Mappings zur Unterscheidung der bereits auf Basis des PIM möglichen automatischen Transformationen (durch Model-Type Mappings zwischen PIM und PSM) von den noch durch manuelle Markierungen zu ermöglichenden Model-Instance Mappings. Erst durch diese zusätzlichen manuellen Markierungen entsteht ein plattformspezifisches Modell, welches alle notwendigen Informationen für die automatische Transformation in die Zielartefakte besitzt.

Als Plattform dient das in Abschnitt 3.4 vorgestellte WCMS OpenCms. Um die automatische Transformation zu den für das Customizing des WCMS erforderlichen Artefakten zu ermöglichen, sind die bereits durch das Platform Independent Model abgebildeten Informationen mit den zu generierenden Artefakten zu vergleichen. Fehlende Informationen sind durch das Platform Specific Model abzubilden. Innerhalb dieses Abschnitts erfolgt eine kurze Betrachtung einer möglichen Vorgehensweise die zur Erstellung der plattformspezifischen Modelle und der Templates für die Generierung der Zielartefakte führt.

Das plattformabhängige Modell der Contentebene

Die Zielartefakte der Contentebene sind die in Abschnitt 3.4.1 beschriebenen XML-Schema-Definitionen (XSD). Diese enthalten neben der Strukturdefinition des Content auch Angaben zu den innerhalb des Web-Content-Management-System zu nutzenden Widgets und Beschränkungen der Inhaltswerte einzelner Content-Elemente. Dabei wird unter einem Widget ein Modul zur Erstellung und Änderung eines bestimmten Content-Typen verstanden. In Tabelle 4.1 sind die möglichen Mappings der Inhaltstypen aufgelistet. In der plattformunabhängigen Spalte sind die Stereotypen (innerhalb französischer Guillemets) und Eigenschaftswerte (kursiv) aufgelistet, in der OpenCms-Spalte die daraus zu generierenden Attribute innerhalb der XSD. Die Model-Instance Mappings haben noch keine eindeutige Zuordnung. Diese muss durch den Fachdesigner mittels ergänzender Markierungen vorgenommen werden. Hier ist bei Klassen des Stereotyp «text» zwischen den beiden Alternativen OpenCmsHtml und OpenCmsString zu entscheiden und die OpenCms-Typen OpenCmsDateTime, OpenCmsDateBoolean sowie OpenCmsColor sind bei entsprechenden Klassen des Stereotyp «customType» als Mapping zu definieren.

Neben diesen Stereotypen und Eigenschaftswerten des plattformunabhängigen Modells kann auch die im PIM modellierte Reihenfolge der Content-Elemente genutzt werden, um die Einträge in den XSD und dadurch die Eingabemasken entsprechend aufzubauen.

Durch den Stereotyp «contentContainer» zusammengefasste logische Einheiten von Content-Elementen können in eigene XSD ausgelagert und durch Verweis referenziert werden. Auf diese Weise können durch die Wiederverwendung von Content-Gruppen redundante Angaben vermindert werden.

Für die Beschränkungen der Inhaltswerte kann die Object Constraint Language der OMG genutzt werden (Miller und Mukerji (2003), S. 3-2).

Mapping	Plattformunabhängig	OpenCms
Model-Type	≪file≫	type="OpenCmsVfsFile"
	≪link≫	type="OpenCmsVarLink"
	≪internalLink≫	type="OpenCmsVarLink"
	≪externalLink≫	type="OpenCmsVarLink"
	≪contentLink≫	type="OpenCmsVarLink"
	≪pageLink≫	type="OpenCmsVfsFile"
	≪editorialNavigationLink≫	type="OpenCmsVfsFile"
	≪picture≫	type="OpenCmsVfsFile"
	minCardinality="int"	minOccurs="int"
	maxCardinality="int"	maxOccurs="int"
	maxCardinalityUnbounded	maxOccurs="unbounded"
Model-Instance	≪text≫	type="OpenCmsHtml" type="OpenCmsString"
	≪customType≫	type="OpenCmsDateTime"
	≪customType≫	type="OpenCmsBoolean"
	≪customType≫	type="OpenCmsColor"

Tabelle 4.1: Mapping-Möglichkeiten vom plattformunabhängigen Metamodell zu OpenCms

Das plattformabhängige Modell der Navigationsebene

Die Navigationsstruktur der WCMA wird, wie in Abschnitt 2.3.1 beschrieben, innerhalb der Templates eingebunden. Zielartefakte der Ebene der Navigation können Java-Server-Pages und Java-Code sein. In beiden Fällen kann innerhalb des generierten Quellcodes die OpenCms API genutzt werden (siehe Abschnitt 3.4.5).

Um eine Mischung von generiertem Code und manuell vom Entwickler ergänztem Code innerhalb der Java-Server-Pages zu verhindern, kann der

generierte Code innerhalb von Page-Fragments (Dunkel und Holitschke (2003), S. 306) gespeichert und vom Entwickler eingebunden werden.

Das plattformabhängige Modell der Präsentationsebene

Durch die Modellelemente der Ebene der Präsentation wird, wie in Abschnitt 3.1.2 geschrieben, festgelegt, wie der Content und die Navigation dem Benutzer der WCMA dargestellt werden. Damit sind die Zielartefakte dieser Ebene die Templates des Moduls der Templateverwaltung (siehe Abschnitt 2.3.1). Wie bei der Navigationsebene kann auch hier in dem generierten Quellcode die OpenCms API genutzt werden.

Durch die Möglichkeit, dass ein Seitentyp («page») die Struktur einer bestehenden Seite nutzten kann (durch den Eigenschaftswert *extendsPage*), um nur bestimmte Bereiche («pageUnit») mit eigenem Inhalt zu überschreiben (Eigenschaftswert *overwritesPageUnit*), wird ein modularer Aufbau der generierten JSP-Templates ermöglicht.

In Abhängigkeit von beispielsweise den Kardinalitätsbeschränkungen und den Typen der Content-Elemente, auf die innerhalb der Präsentationsebene zugegriffen wird, können verschiedene Code-Artefakte vom Generator ausgewählt werden.

Da die innerhalb des plattformabhängigen Modells modellierten Seitentypen, wie in Abschnitt 4.1.1 geschrieben, noch formatunabhängig sind, können den Seitentypen im plattformabhängigen Modell verschiedene Ausgabeformate zugewiesen werden.

Das plattformabhängige Modell der Berechtigungsverwaltungsebene

Die Berechtigungsverwaltung bei OpenCms erfolgt, wie in Abschnitt 3.4.2 beschrieben, direkt über die Benutzeroberfläche. Die Benutzergruppen, Benutzerrollen und Benutzer sind entweder manuell zu übertragen oder es ist mit Hilfe der OpenCms API Java-Code zu generieren, der die Benutzergruppen, Benutzerrollen und Benutzer anlegt.

4.2 Beispiel für die Nutzung des Architektur-Metamodells

Basierend auf dem in Abschnitt 4.1.1 vorgestellten Architektur-Metamodell für eine Web-Content-Management-Anwendung werden in diesem Abschnitt Modelle vorgestellt, welche dieses Metamodell nutzen um Teile der WCMA des Very Large Business Applications Lab (VLBA-Lab) (http://www.vlba-lab.de) als UML-Klassendiagramme zu modellieren.

Abbildung 4.7: Seitenstruktur der Web-Content-Management-Anwendung des VLBA-Lab

In Abbildung 4.7 ist die Struktur der WCMA anhand der Startseite dargestellt. Die durch schwarze Kästen kenntlich gemachten und in der jeweils unteren rechten Ecke benannten Struktur-Elemente finden sich auf den weiteren Seitentypen mit teilweise anderen Inhalten wieder. Die Veränderungen betreffen dabei hauptsächlich den Bereich *MainContent*. Die Navigation im

Bereich *DomainNavigation* wird entsprechend der aktuellen Position innerhalb der Navigationsstruktur angepasst oder überschrieben.

Das plattformunabhängige Modell der Contentebene

In Abbildung 4.8 ist die Struktur des Bereichs *MainContent* vom Seitentyp Homepage dargestellt.

Abbildung 4.8: Struktur der Homepage des VLBA-Lab

Die hier dargestellten Bereiche sind spezifisch für die Homepage und können durch den Content-Typ Homepage verwaltet werden. Dieser ist in Abbildung 4.9 als Klassendiagramm der Ebene Content dargestellt.

Der Content-Typ wird durch die Klasse *Homepage* des Stereotyp «content» gebildet und setzt sich aus den für die Verwaltung der Homepage benötigten Content-Elementen zusammen. Die Reihenfolge der Content-Elemente ist durch die Eigenschaften *isFirst* und *nextElement* angegeben.

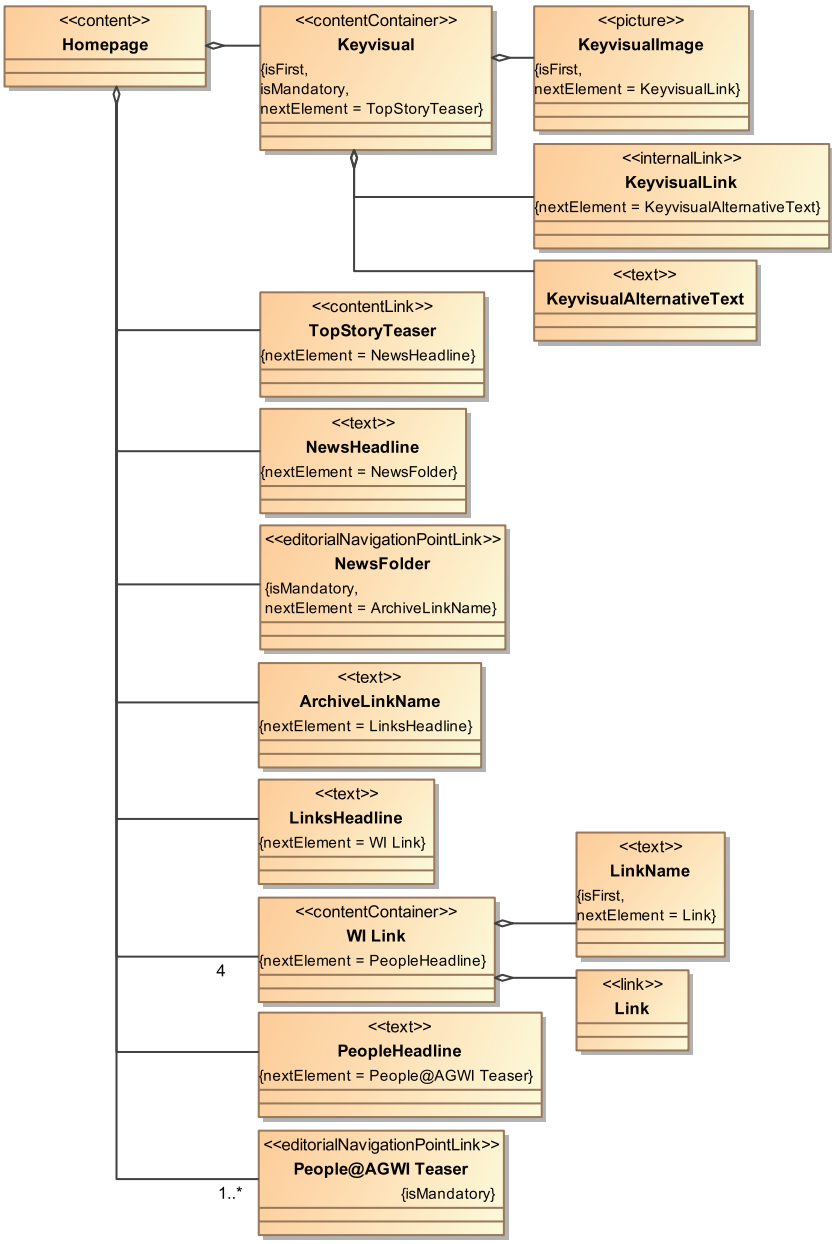

Abbildung 4.9: Modell der Contentebene der Homepage des VLBA-Lab

Der Bereich *ContentKeyvisual* aus Abbildung 4.8 ist durch den erforderlichen (Eigenschaft *isMandatory*) «contentContainer» *Keyvisual* abgebildet. Dieser setzt sich aus einem Bild (Klasse *KeyvisualImage* mit Stereotyp «picture»), einem internen Link für dieses Bild (Klasse *KeyvisualImage* des Stereotyp

«internalLink») und einem alternativen Text für das Bild (Klasse *KeyvisualAlternativeText* mit Stereotyp «text») zusammen.

Der «contentLink» *TopStoryTeaser* dient der Auswahl der aktuellen Top-Story, welche im gleichnamigen Bereich der Abbildung 4.8 dargestellt wird.

Durch die mit dem Stereotyp «text» ausgezeichneten Klassen *NewsHeadline*, *LinksHeadline*, *ArchiveLinkHeadline* und *PeopleHeadline* kann der Text für die Überschriften der drei Bereiche *News*, *Links* und *Menschen* der Abbildung 4.8 verwaltet werden.

Die Klassen *NewsFolder* und *People@AGWI Teaser* ermöglichen die Auswahl von Autoren-Navigationspunkten. Diese Auswahl ist bei beiden Klassen zwingend erforderlich (Eigenschaft *isMandatory*). Bei der Klasse *People@AGWI Teaser* können beliebig viele Autoren-Navigationspunkte angegeben werden (Kardinalitätsbeschränkung *1..**).

Durch den «contentContainer» *WI Link* ist modelliert, dass vier Links (Klasse *Link* mit Stereotyp «link») mit Namen (Klasse *LinkName* mit Stereotyp «text») angegeben werden müssen.

Das plattformunabhängige Modell der Navigationsebene

Wie in Abschnitt 4.1.1 beschrieben, werden auf der Navigationsebene sowohl die Autoren-Navigationsstruktur des WCMS, als auch die innerhalb der WCMA genutzten Navigationsstrukturen, welche auf die Autoren-Navigationsstruktur verweisen, modelliert.

Abbildung 4.10 stellt einen Ausschnitt des Klassendiagramms der Autoren-Navigationsstruktur dar.

Das Wurzelelement (Eigenschaft *isRoot*) der Autoren-Navigationsstruktur bildet die Klasse *root* vom Stereotyp «editorialNavigationPoint». Diese be-

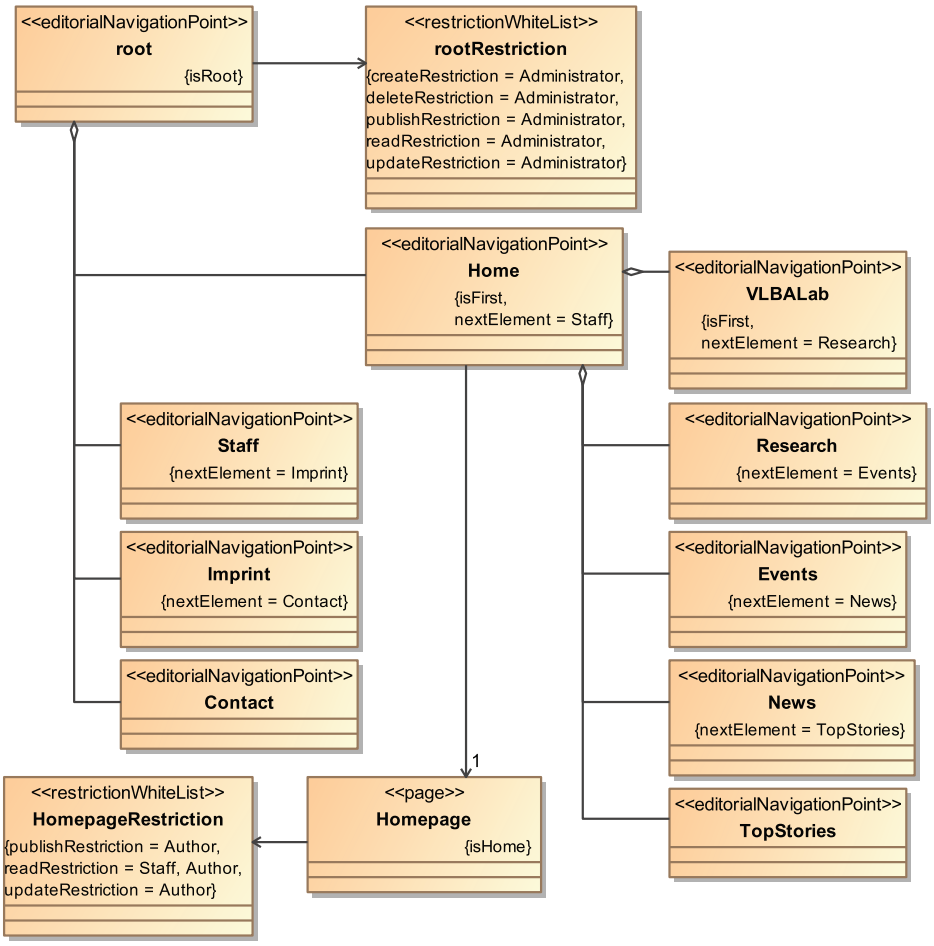

Abbildung 4.10: Modell der Navigationsebene (Autoren-Navigationsstruktur) des VLBA-Lab

steht aus den Autoren-Navigationspunkten (Klassen mit Stereotyp «editorialNavigationPoint») *Home*, *Staff*, *Imprint* und *Contact*. Durch die gerichtete Assoziation der Klasse *root* mit der Klasse *rootRestriction* des Stereotyp «restrictionWhiteList» ist eine Benutzerbeschränkung modelliert. Laut dieser haben nur Benutzer der Rolle *Administrator* das Recht, Elemente unter dem Punkt *root* anzulegen, zu löschen, zu publizieren, zu lesen und zu ändern (Eigenschaften *createRestriction*, *deleteRestriction*, *publishRestriction*, *readRestriction*, *updateRestriction* mit Wert *Administrator*). Die Reihenfolge der Navigationspunkte (ausser dem Wurzelelement) ist durch die Eigenschaften *isFirst* und *nextElement* angegeben.

Der Navigationspunkt *Home* setzt sich aus den weiteren Autoren-Navigationspunkten *VLBA-Lab*, *Research*, *Events*, *News* und *TopStories* zusammen. Durch die Assoziation mit der Klasse *Homepage* des Stereotyp «page» und der Kardinalitätsbeschränkung *1* ist modelliert, dass unter diesem Autoren-Navigationspunkt eine Seite des Seitentyp *Homepage* angelegt wird. Die *Homepage* hat durch die gerichtete Assoziation mit der Klasse *rootRestriction* des Stereotyp «restrictionWhiteList» wie der Autoren-Navigationspunkt *root* eine Benutzerbeschränkung. In diesem Fall haben nur Benutzer der Rolle *Autor* das Recht, die Seite *Homepage* zu publizieren, zu lesen und zu ändern (Eigenschaften *publishRestriction*, *readRestriction* und *updateRestriction* mit Wert *Staff*), sowie Benutzer der Gruppe *Staff* das Recht, die Seite zu lesen (Eigenschaft *readRestriction* mit Wert *Staff*).

Der Autoren-Navigationspunkt *Staff* und die Unterpunkte von *Home* enthalten weitere Autoren-Navigationspunkte, die in diesem Klassendiagramm nicht abgebildet sind.

Wie in Abbildung 4.7 dargestellt, besitzt die Seite des VLBA-Lab zwei Navigationsstrukturen innerhalb der Bereiche *DomainNavigation* und *AdministrativeNavigation*. Diese Navigationsstrukturen sind in Abbildung 4.11 durch die Klassen *AdministrativeNavigation* und *DomainNavigation*, mit dem Stereotyp «presentationNavigation», modelliert.

Die Navigationsstruktur *DomainNavigation* setzt sich aus den Navigationspunkten (Klassen mit dem Stereotyp «presentationNavigationPoint») *Startseite*, *Mitarbeiter*, *Kontakt* und *Impressum* zusammen. Durch die Eigenschaften *isFirst* und *nextElement* wird die Reihenfolge der Navigationspunkte festgelegt. Die Eigenschaft *editorialNavigationPosition* verweist auf einen Autoren-Navigationspunkt. Mit der Eigenschaft *page* sind bei den einzelnen Navigationspunkten bestimmte Seitentypen als Ziel des Verweises festgelegt (z. B. beim Navigationspunkt *Startseite* der Seitentyp *Homepage*).

Die Navigationspunkte der Navigationsstruktur *DomainNavigation* sind *Das VLBA Lab*, *Forschung*, *Veranstaltung*, *News* und *Leitartikel*. Diese sind wie

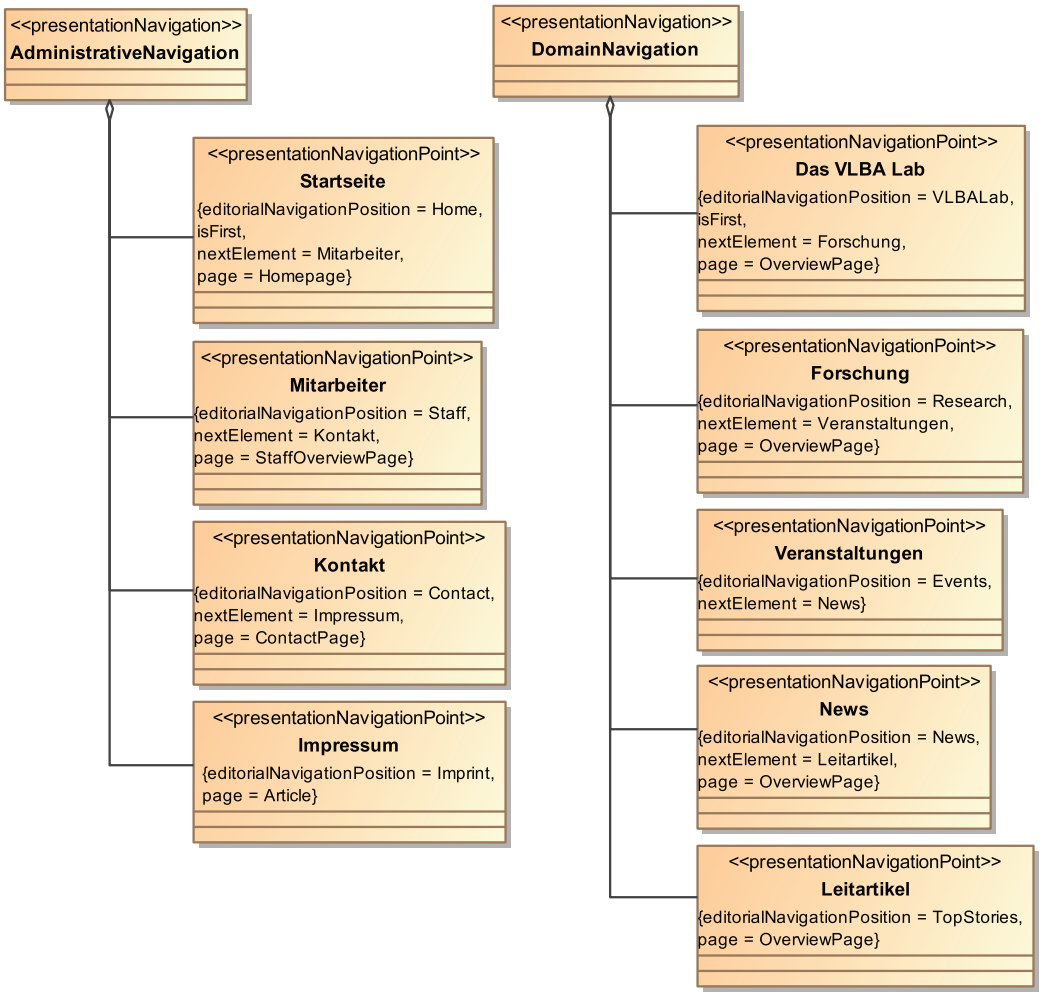

Abbildung 4.11: Modell der Navigationsebene (WCMA-Navigationsstruktur) des VLBA-Lab

die Navigationspunkte der *DomainNavigation* geordnet und verweisen auf verschiedene Autoren-Navigationspunkte, sowie den Seitentyp *OverviewPage*.

Das plattformunabhängige Modell der Präsentationsebene

Wie im Abschnitt 4.1.1 beschrieben, werden auf der Ebene der Präsentation die benötigten Seitentypen der WCMA modelliert. In diesem Beispiel wird

dafür der Seitentyp *Homepage* betrachtet. Die *Homepage* setzt sich aus den in Abbildung 4.7 dargestellten allgemeinen Seitenbereichen der WCMA und den *Homepage*-spezifischen Bereichen aus Abbildung 4.8 zusammen. Die *Homepage*-spezifischen Bereiche befinden sich im Bereich *MainContent* der allgemeinen Seitenbereiche aus Abbildung 4.7.

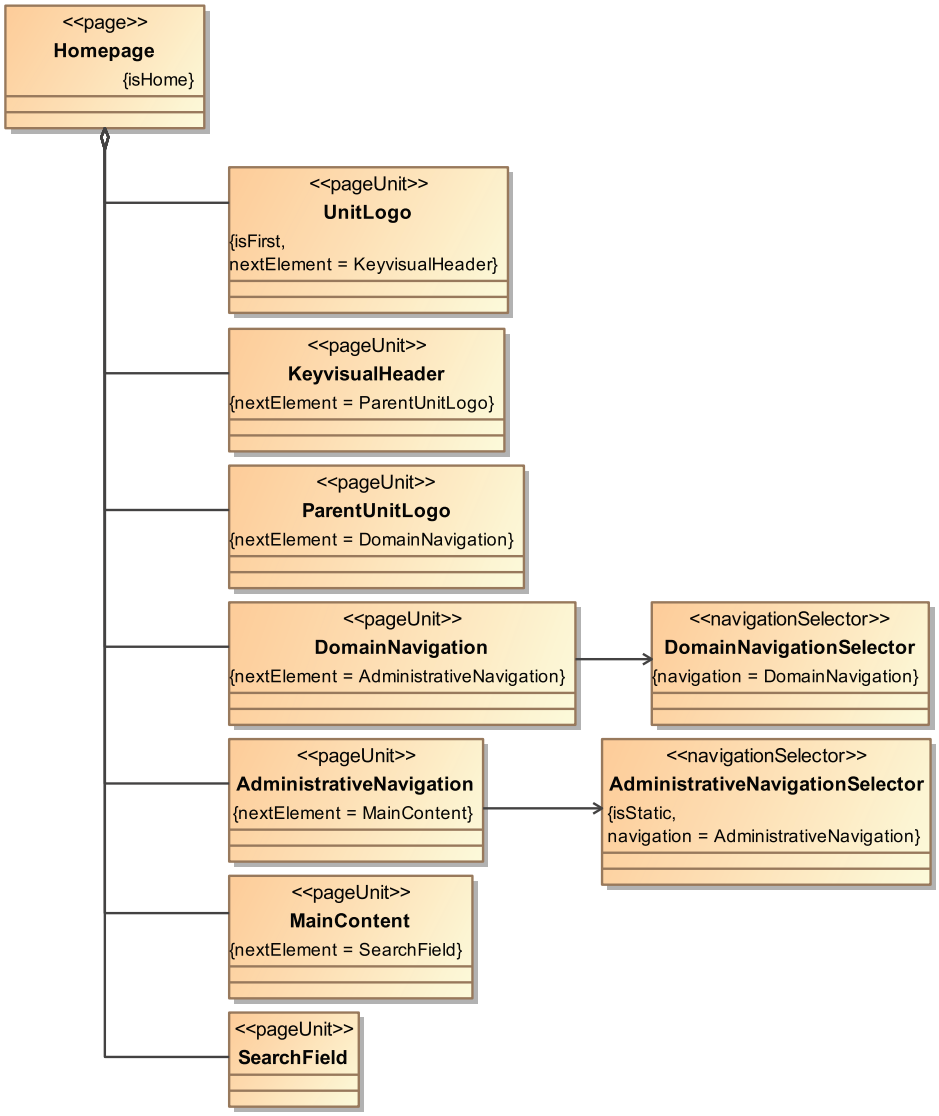

Abbildung 4.12: Modell der Präsentationsebene der Homepage (Übersicht) des VLBA-Lab

In Abbildung 4.12 sind zunächst die allgemeinen Seitenbereiche der WCMA modelliert. Die Klasse *Homepage* des Stereotyp «page» besteht aus den Bereichen (Klassen des Stereotyp *pageUnit*) *UnitLogo*, *KeyvisualHeader*, *ParentUnitLogo*, *DomainNavigation*, *AdministrativeNavigation*, *MainContent* und *SearchField*. Diese Bereiche bilden die gleichnamigen Bereiche aus Abbildung 4.7 ab. Die Reihenfolge der Bereiche ist durch die Eigenschaften *isFirst* und *nextElement* angegeben.

Die Bereiche *DomainNavigation* und *AdministrativeNavigation* nutzen die Klassen des Stereotyp «navigationSelector» *DomainNavigationSelector* und *AdministrativeNavigationSelector*. Durch die Eigenschaft *navigation* verweisen beide auf die Navigationsstrukturen (*DomainNavigation* und *AdministrativeNavigation*) aus der Ebene der Navigation, welche in diesem Bereich darzustellen sind.

In dem in Abbildung 4.13 dargestellten Klassendiagramm werden Bestandteile des Bereichs *MainContent* modelliert. Der *MainContent* besteht aus den bereits aus Abbildung 4.8 bekannten Bereichen *Keyvisual*, *TopStory*, *NewsBlock*, *People@AGWI* und *Links*. Die Bereiche sind im Klassendiagramm durch Klassen mit dem Stereotyp «pageUnit» modelliert. Durch die Eigenschaften *isFirst* und *nextElement* ist die Reihenfolge der Bereiche festgelegt.

Der Bereich *Keyvisual* besteht aus dem Seitenelement (Klasse mit Stereotyp «pageElement») *KeyvisualElements*. Innerhalb des Seitenelements werden mittels der Eigenschaftswerte *KeyvisualLink*, *KeyvisualImage* und *KeyvisualAlternativeText* der Eigenschaft *selectedContentElements* die benötigten Content-Elemente ausgewählt. Der Content-Typ und seine Position werden innerhalb der Klasse *HomepageSelector* (Stereotyp «contentSelector») selektiert. Dies geschieht innerhalb von *HomepageSelector* durch die Eigenschaften *content* und *editorialNavigationPosition*. Durch die Eigenschaft *content* wird der in Abschnitt 4.2 modellierte Content-Typ *Homepage* ausgewählt. Die Eigenschaft *editorialNavigationPosition* gibt die Position des Content-Elements

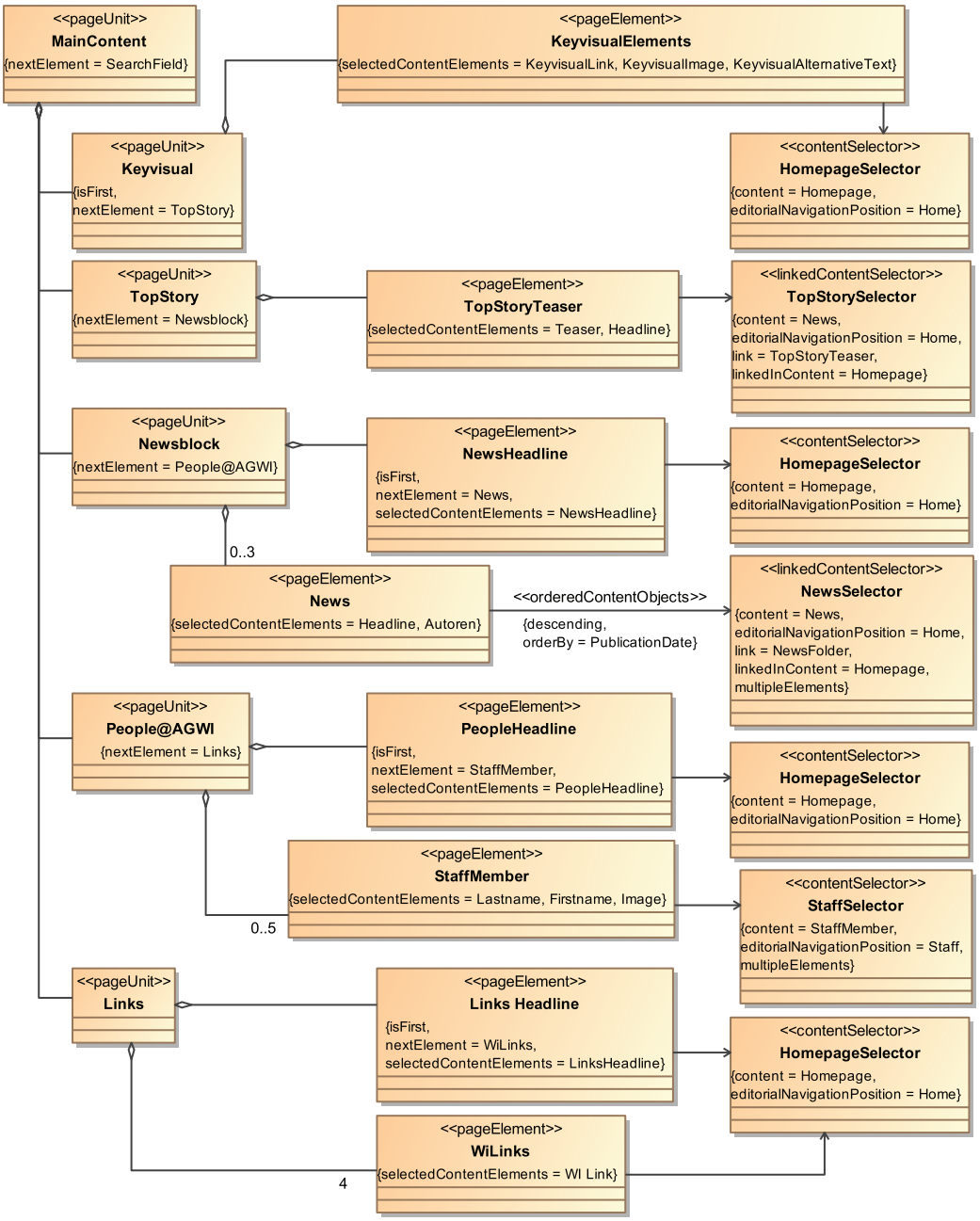

Abbildung 4.13: Modell der Präsentationsebene der Homepage (Bereich MainContent) des VLBA-Lab

innerhalb der in Abschnitt 4.2 modellierten Autoren-Navigationsstruktur mit *Home* an.

Der Bereich *TopStory* besteht aus dem Seitenelement *TopStoryTeaser*. Durch die Assoziation mit *TopStorySelector* (Klasse mit Stereotyp «linkedcontentSelector») wird der zu nutzende Content festgelegt. Durch den Stereotyp «linkedContentSelector» ist abgebildet, dass der Content innerhalb eines anderen Content verlinkt ist. Die Eigenschaften *content* und *editorialNavigationPosition* legen mit den Werten *News* und *Home* den Content-Typ und Autoren-Navigationspunkt des Content fest. Durch *editorialNavigationPosition* wird dabei die Autoren-Navigationsposition des Content angegeben, in dem die Verlinkung existiert. Die Position des verlinkten Content ist zu diesem Zeitpunkt unbekannt. Durch die Eigenschaft *linkedInContent* mit dem Wert *Homepage* wird der Content-Typ des Content bestimmt, in dem der gesuchte Content verlinkt ist. Die Eigenschaft *link* mit dem Wert *TopStoryTeaser* bestimmt das Content-Element des durch *linkedInContent* angegebenen Content-Typ, der den Link auf den zu nutzenden Content enthält.

Der Bereich *Newsblock* setzt sich aus dem Seitenelement *NewsHeadline* und zwischen keinem und drei (Kardinalitätsbeschränkung *0..3* an der Aggregations-Assoziation) Seitenelementen *News* zusammen. Die Reihenfolge der Seitenelemente ist durch die Eigenschaften *isFirst* und *nextElement* festgelegt.

Das Seitenelement *Newsheadline* benutzt wie das Seitenelement *KeyvisualElements* die Klasse *HomepageSelector*, um den Content-Typ *Homepage* der Autoren-Navigationsposition *Home* zu nutzen. Durch die Eigenschaft *selectedContentElements* wird bei *Newsheadline* das Content-Element *NewsHeadline* ausgewählt.

Das Seitenelement *News* nutzt die Klasse *NewsSelector* (Stereotyp «linkedContentSelector»), um auf die Content-Elemente *Headline* und *Autoren* zugreifen zu können (Eigenschaftswerte der Eigenschaft *selectedContentElements*). Die Klasse *NewsSelector* legt durch die Eigenschaften *content* und *editorialNavigationPosition* den Content-Typ *News* und die Autoren-Navigationsposition *Homepage* fest. Die Eigenschaften *linkedInContent* und

link bestimmen den Content-Typ und den Link zu dem zu nutzenden Content. Durch die Eigenschaft *multipleElements* ist festgelegt, dass an der angegebenen Autoren-Navigationsposition mehrere Elemente des Content-Typ existieren und selektiert werden. Die gerichtete Assoziation von der Klasse *News* zu der Klasse *NewsSelector* mit dem Stereotyp «orderedContentAssociation» gibt durch die Eigenschaften *descending* und *orderBy* mit dem Wert *PublicationDate* an, nach welchem Content-Element der durch *NewsSelector* ausgewählte Content absteigend sortiert werden soll.

Der Bereich *People@AGWI* setzt sich zusammen aus dem Seitenelement *PeopleHeadline* und zwischen keinem und fünf Seitenelementen *StaffMember*. Die Reihenfolge der Seitenelemente ist durch die Eigenschaften *isFirst* und *nextElement* festgelegt.

Das Seitenelement *PeopleHeadline* benutzt die Klasse *HomepageSelector*, um den Content-Typ *Homepage* der Autoren-Navigationsposition *Home* zu nutzen. Durch die Eigenschaft *selectedContentElements* wird das Content-Element *PeopleHeadline* ausgewählt.

Das Seitenelement *StaffMember* nutzt die Klasse *StaffSelector* (Stereotyp «contentSelector»), um die in der Eigenschaft *selectedContentElements* ausgewählten Content-Elemente *Lastname*, *Firstname* und *Image* nutzen zu können. In der Klasse *StaffSelector* sind durch die Eigenschaften *content* und *editorialNavigationPosition* der Content-Typ *StaffMember* die Autoren-Navigationsposition *Staff* festgelegt. Die Eigenschaft *multipleElements* gibt an, dass an der angegebenen Autoren-Navigationsposition mehrere Elemente des Content-Typ existieren und selektiert werden.

Der letzte Bereich *Links* setzt sich aus dem Seitenelement *LinksHeadline* und vier Seitenelementen *WiLinks* zusammen. Die Reihenfolge der Seitenelemente ist durch die Eigenschaften *isFirst* und *nextElement* festgelegt.

Das Seitenelement *LinksHeadline* benutzt die Klasse *HomepageSelector*, um den Content-Typ *Homepage* der Autoren-Navigationsposition *Home* zu

nutzen. Durch die Eigenschaft *selectedContentElements* wird das Content-Element *LinksHeadline* ausgewählt.

Das Seitenelement *LinksHeadline* benutzt ebenfalls die Klasse *Homepage-Selector*, um den Content-Typ *Homepage* der Autoren-Navigationsposition *Home* zu nutzen. Durch die Eigenschaft *selectedContentElements* wird hier das Content-Element *WI Link* ausgewählt.

Das plattformunabhängige Modell der Berechtigungsverwaltungsebene

Auf der Berechtigungsverwaltungsebene werden die Benutzergruppen und Benutzerrollen definiert, die für die Aufgaben im WCMS relevant sind (siehe Abschnitt 4.1.1).

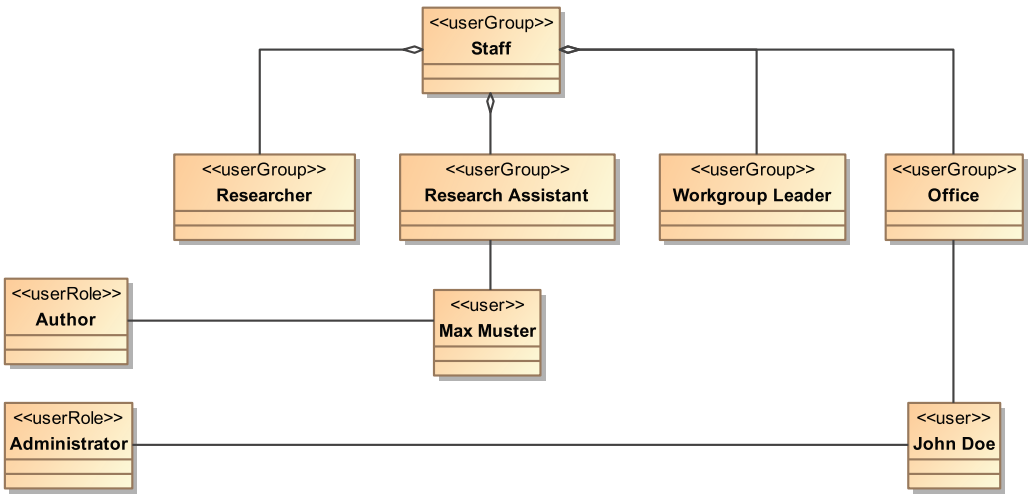

Abbildung 4.14: Modell der Berechtigungsverwaltungsebene der Homepage des VLBA-Lab

In Abbildung 4.14 ist durch die Klasse *Staff* des Stereotyp «userGroup» die Obergruppe aller Mitarbeiter modelliert. Diese besteht aus den Benutzergruppen *Researcher*, *Research Assistant*, *Workgroup Leader* und *Office*. Mit den Klassen *Author* und *Administrator* (Stereotyp «userRole») existieren

zwei Benutzerrollen. Die Benutzer *Max Muster* und *John Doe* (Klassen des Stereotyp «user») sind durch Assoziationen den Benutzergruppen *Research Assistant* bzw. *Office* und den Benutzerrollen *Author* bzw. *Administrator* zugeordnet.

Die vorgestellten Modelle zeigen exemplarisch die Nutzung des in Abschnitt 4.1.1 vorgestellten Metamodells für die Modellierung einer Web-Content-Management-Anwendung.

Kapitel 5

Zusammenfassung und Ausblick

Im ersten Kapitel wurde die Motivation, das Ziel und der Aufbau der Arbeit vorgestellt.

In Kapitel 2 wurde der allgemeine Modellbegriff betrachtet und auf die Nutzung von Modellen in der Softwareentwicklung und der Wirtschaftsinformatik eingegangen. Dabei wurden unterschiedliche Sichtweisen auf den Modell-Begriff sowie verschiedene Modell-Typen identifiziert. Ergebnis der Betrachtung war die Feststellung, dass eine frühzeitige Nutzung von formalen Modellen innerhalb des Softwareentwicklungsprozesses das Potential für eine Steigerung der Effizienz besitzt. In den darauf folgenden Abschnitten wurden die Begrifflichkeiten im Umfeld der Content-Management-Systeme geklärt und mit den Grundmodulen eines Web-Content-Management-System die für diese Arbeit relevante Ausprägung der CMS vorgestellt.

Nach den theoretischen Grundlagen wurden im Kapitel 3 die technischen Grundlagen der Arbeit vorgestellt. Aufgrund der Betrachtung von Web-Content-Management-Systemen wurden Standards für die Entwicklung von Web-Anwendungen vorgestellt und die Nutzung von Modellen innerhalb bestehender Methoden für die Modellierung von Web-Anwendungen aufgezeigt. Mit der modellgetriebenen Softwareentwicklung und der Model Driven Architecture wurden im Anschluss daran die Grundlagen für die Entwicklung einer modellgetriebenen Vorgehensweise beim Customizing beschrieben.

Darauf folgend wurde auf die Umsetzung der beschriebenen Grundmodule eines WCMS am Beispiel von OpenCms eingegangen.

In Kapitel 4 wurden die erarbeiteten Grundlagen bei der Erstellung des Architektur-Metamodells für die Domäne der Web-Content-Management-Systeme zusammengeführt. Es wurde eine Vorgehensweise für das modellgetriebene Customizing innerhalb des Softwareentwicklungsprozess aufgezeigt und auf den Bezug zur Model Driven Architecture eingegangen. Anhand des Beispiels OpenCms wurden die Schritte zur Erstellung eines plattformspezifischen Modells vorgestellt. Im zweiten Abschnitt des Kapitels wurde anhand einer real existierenden Web-Content-Management-Anwendung die Nutzung des entwickelten Architektur-Metamodell bei der Modellierung aufgezeigt.

Das in dieser Arbeit vorgestellte Metamodell der plattformunabhängigen Modelle einer Web-Content-Management-Anwendung ermöglicht den ersten Schritt zu einem modellgetriebenen Customizing eines Web-Content-Management-System und damit die Anpassung an die organisationsspezifischen Anforderungen. Im nächsten Schritt sind Metamodelle für die plattformspezifischen Modelle zu erstellen. Durch die Definition von Mapping-Regeln kann danach eine (teil-) automatische Transformation der plattformunabhängigen in die jeweiligen plattformspezifischen Modelle erfolgen. Diese Modelle können im letzten Schritt in die Zielartefakte für das Customizing des WCMS transformiert werden.

Durch das vorgestellte Metamodell können somit Systemmodelle für Web-Content-Management-Systeme erstellt werden, die unabhängig von einem bestimmten Produkt einsetzbar sind. Durch die Definition von plattformspezifischen Metamodellen und Transformatoren für verschiedene WCMS-Produkte ist es möglich, die einmal erstellten Modelle als Ausgangsbasis für das Customizing verschiedenster WCMS zu nutzen. Bei einer Migration zwischen WCMS könnte somit auf die bestehende, WCMS-unabhängige Spezifikation zurückgegriffen werden.

Literaturverzeichnis

Adam, D. (1996): Planung und Entscheidung: Modelle- Ziele- Methoden. Gabler Verlag, Wiesbaden.

Alkacon Software GmbH (2007a): *Advanced features of OpenCms 7.*

Alkacon Software GmbH (2007b): *A non-technical, business introduction to OpenCms 7.*

Alkacon Software GmbH (2009): *OpenCms 7 client setup.* http://www.opencms.org/en/development/installation/client.html. Stand: 27.03.2009.

Alpar, P., Grob, H. L., Weimann, P. und Winter, R. (2005): Anwendungsorientierte Wirtschaftsinformatik. Vieweg Verlag, Wiesbaden.

Baresi, L., Garzotto, F. und Paolini, P. (2000): *From Web Sites to Web Applications: New Issues for Conceptual Modeling.* In ER '00: Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling, S. 89–100. Springer Verlag, London, UK.

Becker, J. (1995): *Strukturanalogien in Informationsmodellen: Ihre Definition, ihr Nutzen und ihr Einfluß auf die Bildung von Grundsätzen ordnungsmäßiger Modellierung (GoM).* In W. König (Hrsg.), Wirtschaftsinformatik"95: Wettbewerbsfähigkeit, Innovation, Wirtschaftlichkeit, S. 133–150. Physica Verlag, Heidelberg.

Becker, J. und Knackstedt, R. (2002): *Referenzmodellierung 2002: Methoden-Modelle-Erfahrungen.* wi.uni-muenster.de.

Becker, J. und Pfeiffer, D. (2007): Konzeptionelle Modellierung – ein wissenschaftstheoretischer Forschungsleitfaden, S. 1–17. GITO.

Berners-Lee, T. und Fischetti, M. (1999): Der Web-Report. Econ Verlag, Berlin.

Bodendorf, F. (2005): Daten- und Wissensmanagement. Springer Verlag, Berlin.

Bormann, U. und Bormann, C. (2002): Konzepte Content-Repräsentation & Markup-Sprachen. SPC TEIA Lehrbuch Verlag, Berlin.

Brehm, L., Heinzl, A. und Markus, M. L. (2001): *Tailoring ERP Systems: A Spectrum of Choices and their Implications*. In Implications, Proceedings of the 34 th Hawaii International Conference on System Sciences. Maui, S. 3–6. IEEE Computer Society Press.

Broy, M. und Steinbrüggen, R. (2004): Modellbildung in der Informatik. Springer Verlag, Berlin.

Bush, V. (1996): *As we may think*. interactions, Band 3, S. 35–46.

Butcher, M. (2004): Building websites with OpenCms. Packt Publishing, Birmingham.

Butcher, M. (2006): Managing And Customizing OpenCms Websites: Java/JSP XML Content Management. Packt Publishing, Birmingham.

Büchner, H., Zschau, O., Traub, D. und Zahradka, R. (2001): Web Content Management: Websites professionell betreiben. Galileo Press, Bonn.

Christ, O. (2003): Content-management in der Praxis: Erfolgreicher Aufbau und Betrieb unternehmensweiter Portale. Springer Verlag, Berlin.

Dittrich, Y. und Vaucouleur, S. (2008): *Practices around customization of standard systems*. In L.-T. Cheng, J. Sillito, M.-A. D. Storey, B. Tessem, G. Venolia, C. R. B. de Souza, Y. Dittrich, M. John, O. Hazzan, F. Maurer, H. Sharp, J. Singer und S. E. Sim (Hrsg.), CHASE, S. 37–40. ACM.

Dumke, R. (2003): Software Engineering: Eine Einführung für Informatiker und Ingenieure: Systeme, Erfahrungen, Methoden, Tools. Vieweg Verlag, Wiesbaden.

Dumke, R., Lother, M., Wille, C. und Zbrog, F. (2003): Web Engineering. Pearson Education Verlag, München.

Dunkel, J. und Holitschke, A. (2003): Softwarearchitektur für die Praxis. Springer Verlag, Berlin.

Ehlers, L. H. (2003): Content Management Anwendungen: Spezifikation von Internet-Anwendungen auf Basis von Content Management Systemen. Logos Verlag, Berlin.

Ferstl, O. K. und Sinz, E. J. (2006): Grundlagen der Wirtschaftsinformatik. Oldenbourg Wissenschaftsverlag, München.

Fettke, P. und Loos, P. (2003): *Model Driven Architecture (MDA).* Wirtschaftsinformatik, Band 45, Nr. 5, S. 555–559.

Fieser, J. und Dowden, B. (2008): *The Internet Encyclopedia of Philosophy.* http://www.iep.utm.edu/. Stand: 27.03.2009.

Florescu, D., Levy, A. und Mendelzon, A. (1998): *Database techniques for the World-Wide Web: A Survey.* SIGMOD Record, Band 27, Nr. 3, S. 59–74.

France, R. und Rumpe, B. (2007): *Model-driven Development of Complex Software: A Research Roadmap.* In FOSE '07: 2007 Future of Software Engineering, S. 37–54. IEEE Computer Society, Washington.

Frankel, D. S. (2003): Model-Driven Architecture: Applying MDA to Enterprise Computing. Wiley Publishing, Indianapolis.

Frappier, M. und Habrias, H. (2006): *A Comparison of the Specification Methods.* In Software Specification Methods: an Overview Using a Case Study, Kapitel 19. Hermes Science Publishing.

Fraternali, P. (1999): *Tools and approaches for developing data-intensive Web applications: A survey.* ACM Computing Surveys, Band 31, Nr. 3, S. 227–263.

Germán, D. M. und Cowan, D. D. (2000): *Towards a Unified Catalog of Hypermedia Design Patterns.* In Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6. IEEE Computer Society, Washington.

Gersdorf, R. (2002): *Potenziale des Content-Managements.* Wirtschaftsinformatik, Band 44, Nr. 1, S. 75–78.

Goik, M., Hedler, M. und J, W. (2007): *Einsatz von Markup-Languages.* In R. Schmitz (Hrsg.), Kompendium Medieninformatik: Medienpraxis. Springer Verlag, Berlin.

Großmann, M. und Koschek, H. (2005): Unternehmensportale: Grundlagen, Architekturen, Technologien. Xpert.Press, Berlin.

Gruhn, V., Pieper, D. und Röttgers, C. (2006): MDA: Effektives Softwareengineering mit UML2 und Eclipse. Xpert.Press, Berlin.

Gulbins, J., Seyfried, M. und Strack-Zimmermann, H. (2002): Dokumenten-Management: vom Imaging zum Business-Dokument. Springer Verlag.

Heinrich, L. J. (2002): Informationsmanagement. Oldenbourg Wissenschaftsverlag GmbH, München.

Heinrich, L. J., Heinzl, A. und Rothmayr, F. (2004): Wirtschaftsinformatik Lexikon. Oldenbourg Wissenschaftsverlag GmbH, München.

Hess, T. (2001): *Content Syndication.* Wirtschaftsinformatik, Band 43, Nr. 1, S. 83–85.

Hess, T. und Rawolle, J. (2000): *Redaktionssysteme für klassische und digitale Medien.* HMD - Praxis Wirtschaftsinform., Band 211, S. 53–65.

Hesse, W. und Mayr, H. C. (2008): *Modellierung in der Softwaretechnik.* Informatik Spektrum, Band 31, Nr. 5, S. 375.

Jeckle, M., Rupp, C., Hahn, J., Zengler, B. und Queins, S. (2005): UML 2 glasklar. Hanser Fachbuchverlag.

Kampffmeyer, U. (2003): Dokumenten-Technologien: Wohin geht die Reise?: die Bedeutung von DRT Document Related Technologies für Wirtschaft und Gesellschaft. Project Consult, Hamburg.

Karajannis, A. und Bissel, T. (2000): *Webenslauf: Content-Management-Systeme im Vergleich.* iX - Magazin für professionelle Informationstechnik, Band 12, Nr. 11, S. 66–75.

Kempa, M. und Mann, Z. Á. (2005): *Model Driven Architecture.* Informatik Spektrum, Band 28, Nr. 4, S. 298–302.

Kleppe, A. G., Warmer, J. und Bast, W. (2003): MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Company, Boston.

Koch, N. und Kraus, A. (2003): *Towards a Common Metamodel for the Development of Web Applications.* In Proc. of the 3 rd Int. Conf. on Web Engineering (ICWE, S. 497–506. Springer Verlag.

Kretzschmar, O. (2007): *Content-Related-Technologien.* In R. Schmitz (Hrsg.), Kompendium Medieninformatik: Medienpraxis. Springer Verlag, Berlin.

Lankhorst, M. (2005): Enterprise architecture at work: Modelling, Communication and Analysis. Springer Verlag, Berlin.

Lassmann, W. (2006): Wirtschaftsinformatik: Nachschlagwerk für Studium und Praxis. Springer Verlag, Berlin.

Liliedahl, D. (2008): Opencms 7 Development. Packt Publishing, Birmingham.

Lohr, J. und Deppe, A. (2001): Der CMS-Guide. Content Management-Systeme: Erfolgsfaktoren, Geschäftsmodelle, Produktübersicht. Vieweg Verlag, Wiesbaden.

Loos, P. und Scheer, A.-W. (1995): *Vom Informationsmodell zum Anwendungssystem - Nutzenpotentiale für den effizienten Einsatz von Informationssystemen.* In W. König (Hrsg.), Wirtschaftsinformatik"95: Wettbewerbsfähigkeit, Innovation, Wirtschaftlichkeit, S. 185–204. Physica Verlag, Heidelberg.

Ludewig, J. und Lichter, H. (2006): Software Engineering. Dpunkt Verlag, Heidelberg.

Lyardet, F., Rossi, G. und Schwabe, D. (1999): *Discovering and Using Design Patterns in the WWW.* Multimedia Tools Applications, Band 8, Nr. 3, S. 293–308.

Mertens, P., Back, A. und Becker, J. (Hrsg.) (2001): Lexikon der Wirtschaftsinformatik. Springer Verlag, Berlin.

Merz, M. (2002): E-Commerce und E-Business: Marktmodelle, Anwendungen und Technologien. Dpunkt Verlag, Heidelberg.

Milleg, D. und Wagner, B. (2001): *Effizientes Content Management: Ein Hebel für das Medienhaus der Zukunft.* Fachzeitschrift für Information Management & Consulting (IM), Band 16, Nr. 3.

Miller, J. und Mukerji, J. (2003): *MDA Guide Version 1.0.1.* Technischer Bericht, Object Management Group (OMG).

Moreno, N., Fraternalli, P. und Vallecillo, A. (2006): *A UML 2.0 profile for WebML modeling.* In ICWE '06: Workshop proceedings of the sixth international conference on Web engineering, S. 4. ACM, New York.

Möllering, T. und Scheube, S. (2001): *Prozessorientiertes Web Content Management.* Fachzeitschrift für Information Management & Consulting (IM), Band 16, Nr. 3.

Nussbaumer, M. und Gaedke, M. (2006): *Technologies for Web Applications.* In G. Kappel, B. Pröll, S. Reich und W. Retschitzegger (Hrsg.), Web Engineering: The Discipline of Systematic Development of Web Applications. John Wiley & Sons.

OMG (2002): *Meta Object Facility (MOF) Specification*. Technischer Bericht, Object Management Group (OMG).

OMG (2007): *OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2*. Technischer Bericht, Object Management Group (OMG).

OMG (2009): *Business Process Modeling Notation (BPMN) Specification*. Technischer Bericht, Object Management Group (OMG).

Ortner, E. (1997): Methodenneutraler Fachentwurf. B. G. Teubner Verlagsgesellschaft, Leipzig.

Patig, S. (2007): *Modellierung statt Programmierung? Model Driven Architecture für betriebliche Anwendungssysteme*. In C. Rautenstrauch (Hrsg.), Die Zukunft der Anwendungssoftware - die Anwendungssoftware der Zukunft. Shaker Verlag, Aachen.

Petrasch, R. und Meimberg, O. (2006): Model Driven Architecture: Eine Praxisorientierte Einführung in die MDA. Dpunkt Verlag, Heidelberg.

Raggett, D., Hors, A. L. und Jacobs, I. (1999): *HTML 4.01 Specification*. W3C Recommendation, W3C - World Wide Web Consortium.

Rahm, E. und Vossen, G. (2003): Web & Datenbanken. Dpunkt Verlag, Heidelberg.

Rautenstrauch, C. und Schulze, T. (2002): Informatik für Wirtschaftswissenschaftler und Wirtschaftsinformatiker. Springer Verlag, Berlin.

Retschitzegger, W. und Schwinger, W. (2000): *Towards Modeling of DataWeb Applications - A Requirement's Perspective*.

Rohloff, M. (1995): *Integrierte Informationssysteme durch Modellierung von Geschäftsprozessen*. In W. König (Hrsg.), Wirtschaftsinformatik"95: Wettbewerbsfähigkeit, Innovation, Wirtschaftlichkeit, S. 139–152. Physica Verlag, Heidelberg.

Rosemann, M. (1996): Komplexitätsmanagement in Prozessmodellen: Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Gabler Verlag, Wiesbaden.

Rossi, G. und Schwabe, D. (2002): *Object-Oriented Design Structures in Web Application Models.* Annals of Software Engineering, Band 13, Nr. 1-4, S. 97–110.

Rothfuss, G. und Ried, C. (2003): Content Management mit XML: Grundlagen und Anwendungen. Springer Verlag, Berlin.

Rückel, D. C., Steininger, K., Riedl, R. und Roithmayr, F. (2007): *Fallstudie: Einführung eines Enterprise-Content-Management-Systems.* HMD - Praxis Wirtschaftsinformatik, Band 258.

Sarshar, K., Weber, M. und Loos, P. (2006): *Einsatz der Informationsmodellierung bei der Einführung betrieblicher Standardsoftware: Eine empirische Untersuchung bei Energieversorgerunternehmen.* Wirtschaftsinformatik, Band 48, Nr. 2, S. 120–127.

Scheer, A.-W. (2002): ARIS - Vom Geschäftsprozess zum Anwendungssystem. Springer Verlag, Berlin.

Schmidt, G. (1999): Informationsmanagement - Modelle, Methoden, Techniken. Springer Verlag, Berlin, Heidelberg.

Schneider, H. J. (Hrsg.) (1998): Lexikon Informatik und Datenverarbeitung: Version 4.0. R. Oldenbourg Verlag, München, 4. Auflage.

Schoop, E. und Gersdorf, R. (2001): *Content Management für Single Source Multiple Media and Multiple Usage Publishing.* Wirtschaftsstudium (wisu), Band 31, Nr. 7, S. 991–998.

Schwinger, W. und Koch, N. (2006a): *An Introduction to Web Engineering.* In G. Kappel, B. Pröll, S. Reich und W. Retschitzegger (Hrsg.), Web Engineering: The Discipline of Systematic Development of Web Applications. John Wiley & Sons.

Schwinger, W. und Koch, N. (2006b): *Modeling Web Applications*. In G. Kappel, B. Pröll, S. Reich und W. Retschitzegger (Hrsg.), Web Engineering: The Discipline of Systematic Development of Web Applications, S. 39–64. John Wiley & Sons.

Schütte, R. (1998): Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle. Gabler Verlag.

Stachowiak, H. (1973): Allgemeine Modelltheorie. Springer Verlag, Wien.

Stahl, T., Völter, M., Efftinge, S. und Haase, A. (2007): Modellgetriebene Softwareentwicklung. Dpunkt Verlag, Heidelberg.

Stahlknecht, P. und Hasenkamp, U. (2005): Arbeitsbuch Wirtschaftsinformatik. Springer Verlag, Berlin.

Starke, G. (2008): Effektive Software-Architekturen: Ein praktischer Leitfaden. Carl Hanser Verlag, München.

Stein, T. (2000): *Intranet-Organisation: Durch Content-Management die Potenziale des unternehmensinternen Netzwerkzusammenschlusses nutzen*. Wirtschaftsinformatik.

Steinmann, H. und Schreyögg, G. (2000): Management: Grundlagen der Unternehmensführung. Gabler Verlag, Wiesbaden.

Stoyan, R. (2007): Management von Webprojekten: Führung, Projektplan, Vertrag. Springer Verlag, Berlin.

Strahringer, S. (1998): *Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips*. In K. Pohl, A. Schürr und G. Vossen (Hrsg.), Modellierung, Band 9 von *CEUR Workshop Proceedings*. CEUR-WS.org.

Strahringer, S. (2009): *Modellbasierte Einführung von Standardsoftware*. In K. Kurbel, J. Becker, N. Gronau, E. J. Sinz und

L. Suhl (Hrsg.), Enzyklopädie der Wirtschaftsinformatik. Online-Lexikon. http://www.enzyklopaedie-der-wirtschaftsinformatik.de. Stand: 27.03.2009. Oldenbourg.

Szyperski, C. (1998): Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Limited, Essex.

Thomas, O. (2005): *Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation.* In A. W. Scheer (Hrsg.), Veröffentlichungen des Instituts für Wirtschaftsinformatik. Nummer 184. Universität Saarbrücken, Saarbrücken.

Thomas, O. und Scheer, A. W. (2005): Service Engineering: Entwicklung und Gestaltung innovativer Dienstleistungen, Kapitel Customizing von Dienstleistungsinformationssystemen. Springer Verlag, Berlin.

Turowski, K. (2003): Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Shaker Verlag, Aachen.

Urbainczyk, J. (2005): *Von der Idee zum Projekt - Möglichkeiten und Grenzen der Model Driven Architecture.* In R. Breu, T. Matzner, F. Nickl und O. Wiegert (Hrsg.), Software-Engineering: Objektorientierte Techniken, Methoden und Prozesse in der Praxis, S. 29–50. Oldenbourg Wissenschaftsverlag, München.

v. Deursen, A., Klint, P. und Visser, J. (2000): *Domain-specific languages: an annotated bibliography.* SIGPLAN Notices, Band 35, Nr. 6, S. 26–36.

vom Brocke, J. (2003): Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen. Logos Verlag, Berlin.

Walter, T. (2007): Kompendium der Web-Programmierung. Springer Verlag, Berlin.

Weber, W. und Kabst, R. (2006): Einführung in die Betriebswirtschaftslehre. Springer Verlag, Berlin.

Wilhelm, S. (2002): *Content Management Systeme*. In A. Berres und H.-J. Bullinger (Hrsg.), E-Business - Handbuch für Entscheider: Praxiserfahrungen, Strategien, Handlungsempfehlungen. Springer Verlag, Berlin.

Winand, U. und Schellhase, J. S. (2000): *Web-Content-Management*. WISU – Das Wirtschaftsstudium, Band 29, Nr. 10, S. 1334–1344.

Wöhr, H. (2004): Web-Technologien: Konzepte - Programmiermodelle - Architekturen. dpunkt Verlag, Heidelberg.

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Magdeburg, den 01.04.2009