Eine Vorgehensweise zur systematischen Modellierung webbasierter Anwendungen

Von der Fakultät Maschinenbau
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

von

Dipl.-Inform. Michael Wissen

aus Daun

Hauptberichter: Prof. Dr.-Ing. habil. Prof. e. h. Dr. h. c. mult. H.-J. Bullinger
Mitberichter: Prof. Dr.-Ing. Prof. E. h. Dr.-Ing. E. h. Dr. h. c. mult. E. Westkämper

Tag der Einreichung: 26. Januar 2004
Tag der mündlichen Prüfung: 2. Dezember 2004

Institut für Arbeitswissenschaft und
Technologiemanagement (IAT) der Universität
Stuttgart, 2004

IPA-IAO Forschung und Praxis

Berichte aus dem
Fraunhofer-Institut für Produktionstechnik und
Automatisierung (IPA), Stuttgart,
Fraunhofer-Institut für Arbeitswirtschaft und
Organisation (IAO), Stuttgart,
Institut für Industrielle Fertigung und
Fabrikbetrieb (IFF), Universität Stuttgart
und Institut für Arbeitswissenschaft und
Technologiemanagement (IAT), Universität Stuttgart

Herausgeber:
Univ.-Prof. Dr.-Ing. Prof. e.h. Dr.-Ing. e.h. Dr. h.c. mult. Engelbert Westkämper
und
Univ.-Prof. Dr.-Ing. habil. Prof. e.h. Dr. h.c. Hans-Jörg Bullinger

I·A·T Institut
Arbeitswissenschaft und
Technologiemanagement
Universität Stuttgart

Fraunhofer IAO Institut
Arbeitswirtschaft und
Organisation

Michael Wissen

Eine Vorgehensweise zur systematischen Modellierung webbasierter Anwendungen

Nr. 410

JOST-JETTER VERLAG
Fachverlag · 71296 Heimsheim

Dr.-Ing. Dipl.-Inform. Michael Wissen

Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO), Stuttgart

Univ.-Prof. Dr.-Ing. Prof. e.h. Dr.-Ing. e.h. Dr. h.c. mult. Engelbert Westkämper

ord. Professor an der Universität Stuttgart
Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA), Stuttgart

Univ.-Prof. Dr.-Ing. habil. Prof. e.h. Dr. h.c. Hans-Jörg Bullinger

ord. Professor an der Universität Stuttgart
Präsident der Fraunhofer-Gesellschaft, München

D 93

ISBN 3-936947-49-X Jost Jetter Verlag, Heimsheim

Druck: printsystem GmbH, Heimsheim

Geleitwort der Herausgeber

Über den Erfolg und das Bestehen von Unternehmen in einer marktwirtschaftlichen Ordnung entscheidet letztendlich der Absatzmarkt. Das bedeutet, möglichst frühzeitig absatzmarktorientierte Anforderungen sowie deren Veränderungen zu erkennen und darauf zu reagieren.

Neue Technologien und Werkstoffe ermöglichen neue Produkte und eröffnen neue Märkte. Die neuen Produktions- und Informationstechnologien verwandeln signifikant und nachhaltig unsere industrielle Arbeitswelt. Politische und gesellschaftliche Veränderungen signalisieren und begleiten dabei einen Wertewandel, der auch in unseren Industriebetrieben deutlichen Niederschlag findet.

Die Aufgaben des Produktionsmanagements sind vielfältiger und anspruchsvoller geworden. Die Integration des europäischen Marktes, die Globalisierung vieler Industrien, die zunehmende Innovationsgeschwindigkeit, die Entwicklung zur Freizeitgesellschaft und die übergreifenden ökologischen und sozialen Probleme, zu deren Lösung die Wirtschaft ihren Beitrag leisten muss, erfordern von den Führungskräften erweiterte Perspektiven und Antworten, die über den Fokus traditionellen Produktionsmanagements deutlich hinausgehen.

Neue Formen der Arbeitsorganisation im indirekten und direkten Bereich sind heute schon feste Bestandteile innovativer Unternehmen. Die Entkopplung der Arbeitszeit von der Betriebszeit, integrierte Planungsansätze sowie der Aufbau dezentraler Strukturen sind nur einige der Konzepte, welche die aktuellen Entwicklungsrichtungen kennzeichnen. Erfreulich ist der Trend, immer mehr den Menschen in den Mittelpunkt der Arbeitsgestaltung zu stellen - die traditionell eher technokratisch akzentuierten Ansätze weichen einer stärkeren Human- und Organisationsorientierung. Qualifizierungsprogramme, Training und andere Formen der Mitarbeiterentwicklung gewinnen als Differenzierungsmerkmal und als Zukunftsinvestition in *Human Resources* an strategischer Bedeutung.

Von wissenschaftlicher Seite muss dieses Bemühen durch die Entwicklung von Methoden und Vorgehensweisen zur systematischen Analyse und Verbesserung des Systems Produktionsbetrieb einschließlich der erforderlichen Dienstleistungsfunktionen unterstützt werden. Die Ingenieure sind hier gefordert, in enger Zusammenarbeit mit anderen Disziplinen, z. B. der Informatik, der Wirtschaftswissenschaften und der Arbeitswissenschaft, Lösungen zu erarbeiten, die den veränderten Randbedingungen Rechnung tragen.

Die von den Herausgebern langjährig geleiteten Institute, das

- Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA),

- Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO),

- Institut für Industrielle Fertigung und Fabrikbetrieb (IFF), Universität Stuttgart,

- Institut für Arbeitswissenschaft und Technologiemanagement (IAT), Universität Stuttgart

arbeiten in grundlegender und angewandter Forschung intensiv an den oben aufgezeigten Entwicklungen mit. Die Ausstattung der Labors und die Qualifikation der Mitarbeiter haben bereits in der Vergangenheit zu Forschungsergebnissen geführt, die für die Praxis von großem Wert waren. Zur Umsetzung gewonnener Erkenntnisse wird die Schriftenreihe „IPA-IAO - Forschung und Praxis“ herausgegeben. Der vorliegende Band setzt diese Reihe fort. Eine Übersicht über bisher erschienene Titel wird am Schluss dieses Buches gegeben.

Dem Verfasser sei für die geleistete Arbeit gedankt, dem Jost Jetter Verlag für die Aufnahme dieser Schriftenreihe in seine Angebotspalette und der Druckerei für saubere und zügige Ausführung. Möge das Buch von der Fachwelt gut aufgenommen werden.

Engelbert Westkämper Hans-Jörg Bullinger

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO) in Stuttgart. In die Arbeit sind Ergebnisse mehrerer Projekte eingeflossen, die vom Bundesministerium für Bildung und Forschung sowie von Auftraggebern aus der Wirtschaft gefördert wurden.

Meinem Hauptberichter, Herrn Prof. Dr.-Ing. habil. Prof. e. h. Dr. h. c. mult. H.-J. Bullinger, Präsident der Fraunhofer-Gesellschaft, gilt mein besonderer Dank für die Förderung meiner wissenschaftlichen Tätigkeit und die Unterstützung dieser Arbeit.

Herrn Prof. Dr.-Ing. Prof. E. h. Dr.-Ing. E. h. Dr. h. c. mult. E. Westkämper, Leiter des Fraunhofer-Instituts für Produktionstechnik und Automatisierung (IPA), danke ich sehr für die Übernahme des Mitberichts.

Herrn Prof. Dr.-Ing. J. Ziegler möchte ich für seine Betreuung und Förderung des Themas sowie für zahlreiche Anregungen und konstruktive Diskussionen danken.

Meinen Kollegen am IAO bin ich für kritische Fragen im Zusammenhang der Arbeit, für ihre Hilfsbereitschaft und ein kreatives, stets angenehmes und anregendes Arbeitsumfeld sehr dankbar.

Ein besonderer persönlicher und herzlicher Dank gilt Leonie für das entgegengebrachte Verständnis, sowie die Unterstützung und Motivation während der Dissertationsphase.

Stuttgart, im Januar 2005 Michael Wissen

Inhaltsverzeichnis

Verzeichnis der verwendeten Abkürzungen

BEPL	Business Process Execution Language
CASE	Computer Aided Software Engineering
DAML-S	DARPA Agent Markup Language Services
DBPL	Database Programming Language
EORM	Enhanced Object Relationship Method
ER-Modell	Entity-Relationship Model
HDM	Hypertext Design Model
HFPM	Hypermedia Flexible Process Modeling
OOHDM	Object-Oriented Hypermedia Design Method
RMM	Relationship Management Methodology
UML	Unified Modeling Language
WebML	Web Modeling Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Die Entwicklung webbasierter Informationssysteme wird technologisch seit einigen Jahren durch eine Vielzahl unterschiedlicher Systeme wie z. B. Web-Editoren oder insbesondere Content-Management-Systeme (Bullinger et al., 2000) unterstützt. Der Fokus dieser Technologien liegt im Wesentlichen auf der Gestaltung und flexiblen Erzeugung der grafischen Präsentation von Informationen, dem Einstellen und Verwalten von Inhalten innerhalb eines Redaktionsprozesses sowie der Anbindung an Datenbanken und existierende Applikationen.

Inzwischen sind zahlreiche webbasierte Informationssysteme entstanden, die Inhalte von Organisationen, Unternehmen, einzelnen Personen etc. im World Wide Web zur Verfügung stellen. Die Intention solcher Systeme liegt vorrangig in der Bereitstellung einer für den Besucher der Website attraktiven und bedienbaren Informationsplattform, die eine professionelle Außendarstellung erkennen lässt. Diese Zielsetzung wird jedoch oftmals durch die zumeist hohe Komplexität der darzustellenden Informationen und einer Vielzahl von Navigationsmöglichkeiten, die eine Orientierung des Besuchers erschweren (Costagliola et al., 2002), verfehlt. Zusätzlich sind im Unterschied zu konventionellen Applikationen webbasierte Anwendungen oft einer ständigen Überarbeitung sowohl hinsichtlich ihrer Struktur als auch ihrer Inhalte unterworfen. Eine Erneuerung des Informationsangebotes, z. B. aufgrund der Einbindung externer Dienste oder der Änderung der Navigationsstruktur, erfordert oftmals eine weitgehende Neugestaltung der Anwendung.

Auch wenn dieser Problemstellung in den vergangenen Jahren mit Hilfe verschiedener Content-Management-Lösungen begegnet wurde, hat sich eine systematische und methodenorientierte Vorgehensweise bei der Entwicklung webbasierter Anwendungen nicht etabliert. Die methodische und technologische Unterstützung konzentriert sich ausschließlich auf Teilaspekte des Lebenszyklus von Web-Applikationen, während die Verwendung von Entwurfsmodellen nicht Gegenstand derartiger Systeme ist. Im Ergebnis genügen Websites oftmals nur unzureichend den gesteckten Zielen der Betreiber und qualitativen Ansprüchen der Besucher. Die Anwendung von Verfahren und Werkzeugen des Software-Engineering (Pagel & Six, 1994), insbesondere zur Unterteilung des Entwicklungsprozesses in aufeinander folgende Bearbeitungsschritte, hätte jedoch nicht nur positive Auswirkungen auf die Komplexität der Anwendungserstellung, sondern gleichermaßen auf die Qualität und die Wartbarkeit einer Web-Applikation.

Das Fehlen eines Engineering-Ansatzes bei der Entwicklung webbasierter Anwendungen ist insofern beachtlich, als dass starke Bezüge zum Software-Engineering existieren. Auch wenn Web-Anwendungen gegenüber konventionellen Applikatio-

nen eine Reihe von Eigenschaften aufweisen, die neue Anforderungen an die Entwurfs- und Realisierungsmethodik stellen, ist eine Übertragung der methodischen und werkzeuggestützten Vorgehensweisen bei der Softwareentwicklung auf die Web-Anwendungsentwicklung viel versprechend. Nicht zuletzt aus diesem Grund sind vornehmlich aus dem akademischen Umfeld heraus inzwischen etwas mehr als zwanzig Modelle und Methoden zur Hypermedia- bzw. Web-Anwendungsentwicklung entstanden, die in mehreren Studien diskutiert werden (Christodoulou et al., 1998; Ceri et al., 1999; Fraternali, 1999; Koch, 1999; Costagliola et al., 2002; Woukeu, 2003). Dennoch zeigt sich, dass eine Übertragung der entwickelten Konzepte und Methoden auf die Entwicklungspraxis in Unternehmen bislang nicht stattgefunden hat und, falls überhaupt eine systematische Vorgehensweise zugrunde liegt, eigene Methoden favorisiert werden (Barry & Lang, 2001). Die geringe Akzeptanz vorhandener Methodologien trotz nach wie vor hoher Unterstützungsbedarfe seitens der Unternehmen zeigt, dass existierende Modellierungstechniken und Entwicklungsmethoden den gestellten Anforderungen nicht gerecht werden und für die betriebliche Praxis im Allgemeinen ungeeignet sind.

Um den Entwicklungsaufwand webbasierter Anwendungen zu reduzieren und gleichzeitig eine hohe Qualität der Website zu erreichen, bedarf es einer Vorgehensweise, die einer durchgängigen Entwicklungsmethodik unter Berücksichtigung der betrieblichen Praxis folgt und zusätzlich auf eine systemtechnische Unterstützung ausgerichtet ist.

2 Ziele und Aufbau der Arbeit

2.1 Gegenstand und Bedeutung

Seit der Entstehung des World Wide Web (WWW) zu Beginn der neunziger Jahre hat sich der Stellenwert webbasierter Anwendungen kontinuierlich erhöht. Durch die stetig ansteigende Zahl von Benutzern und dem dadurch verstärkten wirtschaftlichen Interesse dient das WWW inzwischen nicht nur als reines Informationssystem, sondern wird zusätzlich als Plattform kommerzieller Anwendungen genutzt. Innerhalb weniger Jahre sind neue verteilte Web-Anwendungen mit verstärkt funktionalen Merkmalen entstanden, die sich zunehmend durch verteilte Prozesse, heterogene Systeme, komplexe Benutzungsschnittstellen und Navigationsstrukturen sowie kurzen Lebenszyklen auszeichnen (Gaedke, 1999). Gleichzeitig werden Web-Projekte immer umfangreicher und müssen zudem mit anderen Anwendungen interagieren (Ceri et al., 2000).

Auch wenn die Erstellung webbasierter Anwendungen in ihrer Komplexität mit der klassischen Softwareentwicklung gleichbedeutend ist (Gellersen, 1997), erfolgte bislang die Entwicklung von Web-Applikationen zumeist ad hoc. Inzwischen vollzieht sich ein Wandel zu einer verstärkt professionellen Herangehensweise, jedoch unterliegt der Entwicklungsprozess noch immer keiner systematischen und methodenunterstützten Vorgehensweise. Begünstigt wird dieser Ansatz durch zahlreiche Softwaretools wie HTML-Editoren, Database Publishing Wizards, Web Site Managers etc., die lediglich den Implementierungsaspekt berücksichtigen (Retschitzegger & Schwinger, 2000). In der Folge sind webbasierte Anwendungen nicht nur in ihrer Entwicklung u. a. aufgrund fehlender Konzepte zur Wiederverwendbarkeit aufwändig, sondern auch kostenintensiv in Bezug auf Wartbarkeit und Pflege.

Insgesamt zeigt der Entwicklungsprozess webbasierter Anwendungen, der in Analogie zur Softwaretechnik als *Web-Engineering* (Gellerson et al., 1997) bezeichnet wird, starke Bezüge zum Software-Engineering, die vor allem dadurch zu begründen sind, dass bei der Erstellung von Web-Anwendungen ebenfalls Programmierung und Software-Entwicklung von Bedeutung sind. Ebenso wie beim Software-Engineering unterteilt sich der Web-Engineering-Prozess in die Phasen *Analyse*, *Entwurf*, *Implementierung* und *Betrieb und Wartung*, jedoch existieren Unterschiede sowohl zwischen Web-Applikationen und herkömmlichen Anwendungen als auch den zugehörigen Entwicklungsprozessen:

- Web-Anwendungen sind überwiegend content-intensiv und enthalten oftmals Mischformen informationsvermittelnder und applikationsorientierter Teile mit vermehrt hoher multimedialer Ausprägung.

- Web-Applikationen unterliegen einer ständigen Veränderung bzw. Erweiterung der dargestellten Inhalte und Funktionalitäten.
- Visuelle Eigenschaften nehmen bei der Entwicklung von Web-Anwendungen einen hohen Stellenwert ein. Neben der Programmierung sind auch kreative bzw. künstlerische Aspekte bzgl. des Designs von Bedeutung.
- Web-Anwendungen berücksichtigen verstärkt unterschiedliche Besucherrollen, die den Benutzern eine an ihre Interessen und Aufgaben ausgerichtete inhaltliche Darstellung ermöglichen. Benutzerprofile erlauben darüber hinaus die Einbeziehung individueller Interessen.
- Web-Applikation unterliegen oftmals kurzen Entwicklungszeiten, so dass vor allem der Analyse- und Konzeptionsphase als auch der Testphase nicht die erforderliche Bearbeitungszeit beigemessen wird.
- Gewöhnlich sind an Web-Projekten Mitarbeiter verschiedener Disziplinen beteiligt, die unterschiedliche Ausrichtungen, Erfahrungen und Kenntnisse besitzen.

Die verschiedenen Eigenschaften und Merkmale webbasierter Applikationen zeigen die Notwendigkeit einer systematischen und formalen Vorgehensweise, die einem durchgängigen Entwicklungsprozess zugrunde liegt. Trotz der Komplexität der Web-Anwendungsentwicklung basiert die Konzeption und Umsetzung üblicherweise auf dem Wissen und den Erfahrungen einzelner Entwickler. Dies ist vor allem auf das Fehlen allgemein verständlicher Entwurfsmodelle zurückzuführen, die nicht nur die Kommunikation zwischen den einzelnen Projektmitarbeitern verbessern können (Garzotto et al., 1993a), sondern dem Entwickler ermöglichen, Detailfragen der Umsetzung zunächst außer Acht zu lassen. Formale Design-Techniken und Entwurfsmodelle, die auf einer grafischen Notation basieren, können darüber hinaus den Entwickler bei der Erstellung von Web-Anwendungen unterstützen, die hohen qualitativen Ansprüchen hinsichtlich Lesbarkeit, Navigierbarkeit und Wartung gerecht werden müssen (Bichler & Nusser, 1996a).

2.2 Problemstellung

Von diesen Überlegungen ausgehend, stellen sich verschiedene Anforderungen hinsichtlich einer systematischen und durchgängigen Vorgehensweise zur Entwicklung webbasierter Anwendungen.

Wesentlicher Aspekt einer Entwicklungsmethodik ist die Aufteilung des Web-Engineering-Prozesses in Teilschritte, die sich auf die verschiedenen Entwurfsbereiche beziehen. Auch wenn dieser Ansatz zumindest bei allen bedeutsamen Methodologien, die aus dem Forschungsumfeld entstanden sind, berücksichtigt wird, zeigen sich Schwierigkeiten in den Übergängen zwischen den einzelnen Bearbei-

tungsschritten. Die Übertragung der einem Prozessschritt zugehörigen Entwurfsmodelle zur weiteren Bearbeitung im nachfolgenden Schritt ist bislang nur unzureichend gelöst.

In diesem Zusammenhang stellt sich neben der Überführung der Modelle auch die Frage nach Mechanismen zur Qualitätskontrolle und -sicherstellung am Ende einer Bearbeitungsphase. Vor allem die Komplexität der Benutzungsschnittstellen erfordert eine frühzeitige Evaluation der Effizienz von Struktur, Navigation und Darstellung, die beispielsweise mittels Möglichkeiten zum rechtzeitigen Prototyping durchgeführt werden kann (Nielsen, 1993).

Neben den Anforderungen, die in erster Linie die Vorgehensweise der Web-Anwendungsentwicklung betreffen, existieren weitere, vor allem auf die Konzeption der Entwurfsmodelle bezogene Kriterien. Hierzu zählt zunächst die Anwendbarkeit der verschiedenen Modelle. Einerseits sollen sie den Entwickler im Sinne einer methodischen Vorgehensweise unterstützen und durch Abstraktion Detailfragen der Implementierung in den ersten Entwurfsphasen unberücksichtigt lassen. Andererseits müssen die grafischen Modelle auch für Zielgruppen ohne technische Vorbildung verständlich sein, um eine Kommunikation zwischen den beteiligten Mitarbeitern aus unterschiedlichen Fachdisziplinen zu ermöglichen.

Die hohe Änderungsdynamik webbasierter Anwendungen stellt einen weiteren Aspekt dar. Eine geeignete Methodik sollte hierzu Modellierungskonzepte bereitstellen, die nicht nur den Zugriff auf dynamische Inhalte erlauben, sondern auch Veränderungen in den übergeordneten Navigationsstrukturen einbeziehen. Von Bedeutung ist darüber hinaus neben der Möglichkeit zur Modifikation der grafischen Entwurfsmodelle auch die Wiederverwendbarkeit von Teilen des Entwurfs, um wiederkehrende Muster einer Anwendung bei der Modellierung berücksichtigen zu können.

Die Verwendung von Entwurfsmodellen soll einerseits den Entwicklungsprozess abstrahieren, andererseits jedoch die Erstellung qualitativ hochwertiger grafischer Interfaces vereinfachen. Hierzu ist es erforderlich, die inzwischen weit reichenden Design-Möglichkeiten der Benutzungsschnittstellen nicht einzuschränken und möglichst vollständig in die Modellentwicklung einzubeziehen.

Einen hohen Stellenwert nimmt neben der Bereitstellung derselben Inhalte für verschiedene Anwender bzw. der Berücksichtigung unterschiedlicher Kontexte wie Zeit, Ort etc. zudem die Einbindung externer Inhalte und Applikationen in die Web-Anwendung ein, da zunehmend funktionale Merkmale in webbasierten Anwendungen vorzufinden sind.

Eine wesentliche Anforderung ist schließlich das automatisierte Erstellen von Prototypen bzw. einer lauffähigen Anwendung anhand der entwickelten Entwurfsmodelle.

Die Überführung der Modelle sollte zwischen einzelnen Bearbeitungsphasen möglich sein, um qualitative Merkmale frühzeitig evaluieren zu können.

Die Anforderungen, die sich insgesamt aus diesen Überlegungen ergeben, lassen sich entsprechend Tabelle 2-1 zusammenfassen.

Anforderung	Bezug
Durchgängigkeit	Vorgehensweise
Phasenorientierung	Vorgehensweise
Verständlichkeit	Vorgehensweise / Konzeption der Entwurfsmodelle
Berücksichtigung der Änderungsdynamik	Konzeption der Entwurfsmodelle
Wiederverwendbarkeit	Konzeption der Entwurfsmodelle
Vollständigkeit der Modellkomponenten	Konzeption der Entwurfsmodelle
Berücksichtigung von Kontextfaktoren	Konzeption der Entwurfsmodelle
Anbindung externer Inhalte und Applikationen	Konzeption der Entwurfsmodelle
Automatisierbarkeit	Konzeption der Entwurfsmodelle

Tabelle 2-1: Anforderungen an die Methodik

2.3 Zielsetzung

Ziel der vorliegenden Arbeit ist die Konzeption einer Methodik für den komponentenorientierten Entwurf webbasierter Anwendungen, die auf Grundlage der Analyse und Anforderungsspezifikation webbasierter Anwendungen alle Modellierungsphasen in einem integrierten Ansatz unterstützt. In einem Methodenverbund sollen hierzu dem Benutzer Methoden und Metamodelle für die unterschiedlichen Entwicklungsphasen und Entwurfsbereiche bereitgestellt werden, die gleichzeitig als

Grundlage für die automatisierte Generierung der modellierten Web-Anwendung dienen.

Neben der Unterstützung des Entwicklungsprozesses und der Ausrichtung auf eine weitestgehend automatisierte Weiterverarbeitung der vom Benutzer erstellten Modelle, existieren weitere Anforderungen, die bei der Konzeption der Entwicklungsmethodik berücksichtigt werden sollen:

- Der Methodenverbund, der auf einen schrittweisen Aufbau der Web-Applikation ausgerichtet ist, sollte Möglichkeiten zur frühzeitigen Evaluation zur Verfügung stellen, mit denen die Qualität der Navigationsstruktur und der Präsentation überprüft werden kann.
- Eine formale Beschreibung der verschiedenen Entwurfsmodelle sollte die Überführbarkeit der vom Benutzer erstellten Modelle in den jeweils nachfolgenden Bearbeitungsschritt sicherstellen.
- Die verwendete grafische Beschreibung der Modelle sollte eine arbeitsfähige Notation darstellen, die für die unterschiedlichen Entwicklergruppen der verschiedenen Fachdisziplinen verständlich ist.
- Änderungen an Inhalten, Navigationspfaden bzw. der Darstellung sollten sowohl zur Modellierungszeit als auch zur Laufzeit unabhängig voneinander durchführbar sein.
- Die Modellierung sollte die Wiederverwendbarkeit von Teilen des Entwurfs ermöglichen.
- Der Methodenverbund sollte hinsichtlich der enthaltenen Modellierungskomponenten vollständig sein, um alle Aspekte der Web-Anwendungsentwicklung abzudecken. Dies schließt insbesondere die Berücksichtigung extern verfügbarer Dienste und Inhalte, die an die Website angebunden werden sollen, und die Betrachtung von Kontextfaktoren ein.

Ein wesentliches Ziel der vorliegenden Arbeit ist es, die den Entwicklungsphasen zugehörigen Entwurfsmodelle derart zu gestalten, dass sie durch geeignete softwaretechnische Werkzeuge erstellt werden können und dadurch den Benutzer bei der Modellierung unterstützen. Gleichzeitig ist die Methodologie auf die Integration mit existierenden Methoden des Content-Engineering, insbesondere des Content-Management ausgerichtet. Hierdurch sollen die in der Praxis üblichen Methoden und Werkzeuge zum webbasierten Content-Management durch Modellierungsaspekte erweitert werden, um die Anwendbarkeit der entwickelten Methodik im Rahmen eines Web-Engineering-Prozesses zu gewährleisten.

2.4 Aufbau der Arbeit

Die dieser Arbeit zugrunde liegende Vorgehensweise folgt dem in Abbildung 2-1 gewählten Ansatz. Den Ausgangspunkt bildet die Analyse existierender Methodologien sowie in der Praxis üblicher Verfahren zur Entwicklung von Web-Anwendungen (Kap. 3). Auf Grundlage der Untersuchungen werden Defizite vorhandener Methoden gegenüber den genannten Zielsetzungen herausgearbeitet und Anforderungen an den Gestaltungsprozess von Web-Applikationen erhoben, die eine Konkretisierung der Zielsetzungen beinhalten.

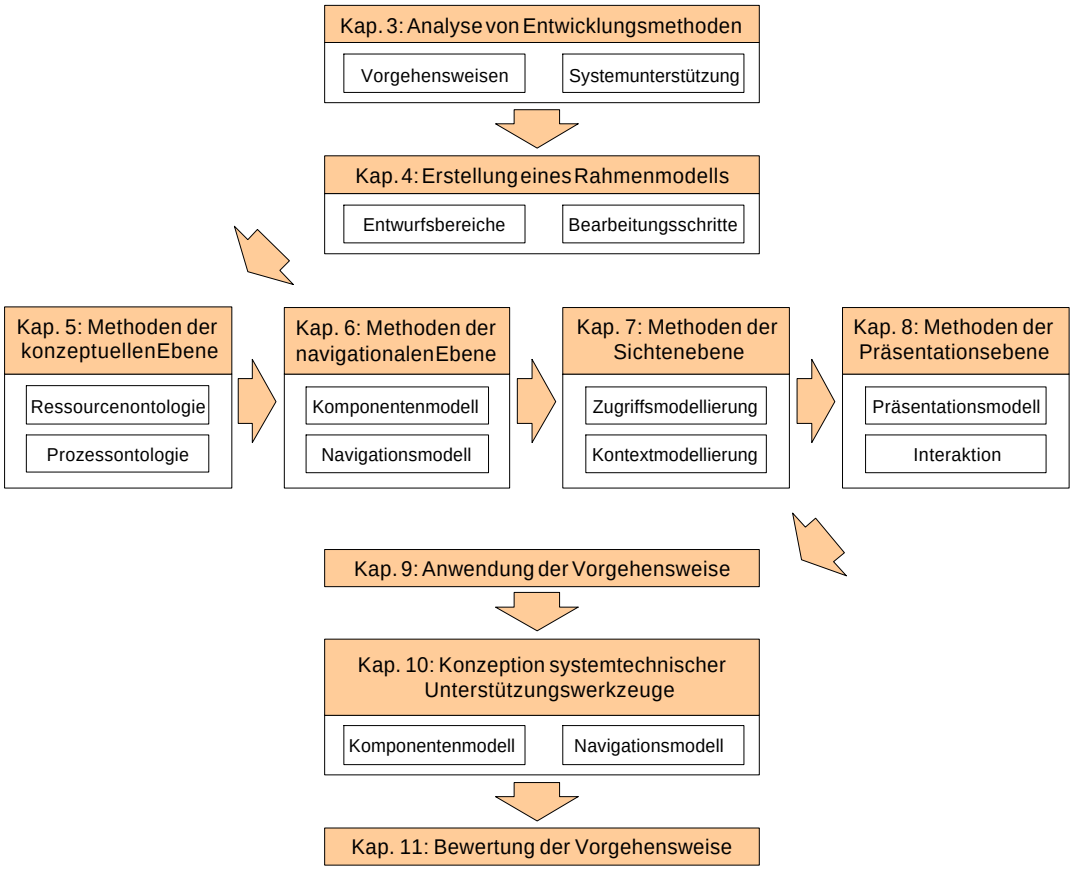


Abbildung 2-1: Aufbau der vorliegenden Arbeit

Für den Entwicklungsprozess webbasierter Anwendungen wird in Kapitel 4 zunächst ein Rahmenmodell erstellt, das den unterschiedlichen Entwurfsebenen einer Web-Applikation einzelne Bearbeitungsschritte zuweist. Zu den im Web-Kontext relevanten Entwurfsbereichen werden Entwicklungsmodelle konzipiert und innerhalb des Rahmenmodells zu einem Methodenverbund zusammengefasst. Die Entwurfsmodelle beziehen sich auf die Konzeptualisierung des Informationsbestandes (Konzeptuelles Modell), dem Aufbau einzelner Webseiten einschließlich ihrer Navi-

gationsstruktur (Komponenten- und Navigationsmodell), die Definition der Zugriffe (Sichtenmodell) sowie die Festlegung der Darstellung und Interaktion (Präsentationsmodell).

Kapitel 5 befasst sich mit Methoden zur Erstellung eines konzeptuellen Modells unter Verwendung ontologiebasierter Verfahren, mit deren Hilfe eine Strukturierung der im Kontext der Web-Applikation relevanten Informationsbestände erreicht werden kann. Das Modell stellt zum einen die Grundlage für den Zugriff auf die Inhalte seitens der Web-Anwendung zur Laufzeit dar und dient zum anderen dem systematischen Entwurf der Navigations- und Sichtenstruktur.

Der Aufbau einzelner Webseiten sowie die Konzeption einer seitenübergreifenden Navigationsstruktur ist Gegenstand der in Kapitel 6 thematisierten Komponenten- und Navigationsmodellierung. Hierzu werden die zum Entwurf von Websites erforderlichen Komponenten formal beschrieben und eine grafische Notation zur Modellierung der Navigationsstruktur eingeführt.

Die zur Entwicklung des Navigationsmodells verwendete Notation wird in Kapitel 7 durch Aspekte der Sichten- bzw. Zugriffsmodellierung erweitert. Durch die Aufnahme von Nebenbedingungen für den Informationszugriff können sowohl benutzerspezifische bzw. rollenbasierte Inhaltsdarstellungen als auch kontextbezogene Informationen in die Modellierung einbezogen werden.

Aufbauend auf den in den vorangegangenen Kapiteln definierten Metamodellen zur Navigations- und Sichtenmodellierung vervollständigt Kapitel 8 den Methodenverbund durch die Beschreibung von Präsentationsklassen. Hierdurch lassen sich Eigenschaften der grafischen Darstellung bzgl. der Website modellieren.

Anhand eines Beispiels wird in Kapitel 9 die Anwendung der in dieser Arbeit entwickelten Vorgehensweise einschließlich der verschiedenen Entwurfsmodelle erläutert. Das Anwendungsbeispiel, das die Neukonzeption der Website eines Versicherungsunternehmens beinhaltet, soll gleichzeitig die Funktionsweise der automatisierbaren Generierung der Website verdeutlichen.

Nachfolgend zeigt Kapitel 10 Möglichkeiten zur systemtechnischen Unterstützung der Modellentwicklung auf, deren Zielsetzung die Integration der Vorgehensweise und des zugehörigen Methodenverbundes in den systemunterstützten Entwicklungsprozess webbasierter Anwendungen ist.

Eine abschließende Diskussion der Vorgehensweise hinsichtlich ihrer Anwendbarkeit und der erreichten Ziele erfolgt in Kapitel 11. Die Bewertung der Ergebnisse orientiert sich an der Durchführbarkeit des gesamten Engineering-Prozesses seitens des Entwicklers sowie an der Nutzbarkeit und Verständlichkeit der verschiedenen Entwurfsmodelle.

3 Methoden für die Entwicklung von Web-Anwendungen

In den vergangenen zehn Jahren wurde eine Vielzahl von Methoden für den Entwurf von Hypermedia- bzw. Web-Applikationen vorgeschlagen. Web-Anwendungen sind eng verwandt mit Hypermedia-Anwendungen, so dass sich einige ursprünglich zur Modellierung von Hypermedia-Applikationen vorgesehene Methoden und Modelle auf den Webbereich übertragen ließen.

Mit dem Ziel, die systematische Entwicklung qualitativ hochwertiger Hypermedia-Anwendungen zu ermöglichen, wurde von Lowe und Hall ein Rahmenmodell für Entwicklungsprozesse von Hypermedia-Anwendungen bereitgestellt (Lowe & Hall, 1999). Es umfasst die Phasen *Domänenanalyse*, *Produktmodellierung*, *Projektmodellierung*, *Entwicklung* und *Dokumentation* und beschreibt die verschiedenen Aktivitäten jeder einzelnen Phase. Für den Entwicklungs- und Designprozess gilt die Zuordnung der Aktivitäten *Systemanalyse*, *Design*, *Produktion*, *Verifikation* und *Wartung.*

Wird das Rahmenmodell den für die Entwicklung von Hypermedia-Anwendungen existierenden Vorgehensweisen zugrunde gelegt, ist eine Fokussierung der vorhandenen Methoden auf die Design-Phase des Gesamtprozesses erkennbar. Die Ziele dieser Phase liegen im Wesentlichen in der Modellierung der Anwendungsdomäne, der Analyse von Eigenschaften und Relationen innerhalb der Domäne und der zielgruppenspezifischen Darstellung der Informationen gegenüber den Besuchern. Entsprechend konzentrieren sich die einzelnen Vorgehensweisen auf unterschiedliche Entwurfsbereiche innerhalb des Entwicklungsprozesses und stellen hierfür verschiedene Modelle zur Verfügung.

Im Folgenden werden zunächst existierende Entwicklungsmethoden für Web-Anwendungen untersucht, bevor auf die verschiedenen Werkzeuge zur Unterstützung des Entwicklungsprozesses eingegangen wird. Um eine Vergleichbarkeit der einzelnen Methodologien zu erreichen, soll in Analogie zur klassischen Softwareentwicklung der *Unified Process* (Jacobson et al., 1999) herangezogen werden, der die Phasen *Anforderungsdefinition*, *Analyse*, *Design*, *Implementierung* und *Test* beinhaltet. An diesen Prozessphasen orientiert sich die Bewertung sowohl der Entwicklungsmethoden als auch der Werkzeuge.

3.1 Entwicklungsmethoden für Web-Anwendungen

Die meisten Vorgehensweisen zur Modellierung von Hypermedia- bzw. Web-Anwendungen lassen sich entsprechend der verwendeten Modellierungstechnik in die Bereiche Entity-Relationship-basierte (E-R-basiert) (Chen, 1976), objektorien-

tierte oder ontologiebasierte Methoden einordnen (Woukeu et al., 2003). Im Folgenden sollen die wichtigsten Methoden und Modelle mit ihren Eigenschaften kurz vorgestellt werden.

3.1.1 Entity-Relationship-basierte Vorgehensweisen

Die ursprüngliche Intention einiger wesentlicher Vorgehensweisen zur Modellierung und Implementierung von Web-Anwendungen war die Entwicklung von Hypermedia-Anwendungen. Zu den ersten vorgeschlagenen Methoden gehören in diesem Zusammenhang das *Hypertext Design Model* (HDM) (Garzotto et al., 1993a) und dessen Nachfolger HDM2 (Garzotto et al., 1993b), die später auch zur Modellierung von Websites vorgesehen wurden. Ziel beider Methoden ist zunächst der Entwurf globaler Klassen und Informationseinheiten, denen kein konkreter Inhalt zugeordnet ist, sowie einer zugehörigen Navigationsstruktur. Die Implementierung der Details kann darauf aufbauend in einem weiteren Schritt erfolgen. HDM und HDM2 konzentrieren sich ausschließlich auf die Modellierung der Anwendungsdomäne, der Navigationsstruktur und der Präsentation und beinhalten keine Vorgehensweise zur Umsetzung bzw. Erstellung der Anwendung.

Neben HDM und HDM2 zählt ebenfalls die *Relationship Management Methodology* (RMM) (Isakowitz et al., 1995) zu den E-R-basierten Methoden. RMM zerlegt den Entwicklungsprozess in insgesamt sieben Schritte (E-R-Design, Entity-Design, Navigations-Design, Benutzungsschnittstellen-Design, Konvertierungsprotokoll-Design, Laufzeitverhalten-Design und Implementierung) und besitzt ein Datenmodell (RMDM), das auf dem von HDM aufbaut. Diesen Kernschritten sind die außerhalb von RMM liegenden Phasen Machbarkeitsanalyse, Anforderungsdefinition und Systemauswahl des Software-Engineering vorgelagert. Im Anschluss an die Implementierung folgt die Evaluations- und Testphase, hier wird allerdings auf andere Arbeiten verwiesen (Garzotto et al., 1993a; Botafogo et al., 1992). RMM betrachtet vorwiegend die Design-, Entwicklungs- und Konstruktionsphase und eignet sich für gut strukturierte Daten, die häufigen Änderungen und Updates unterzogen werden. Hierbei konzentriert sich die Unterstützung der Design-Phase ausschließlich auf die Verwendung eines speziellen Datenmodells (Lowe & Hall, 1999). Alle weiteren Phasen neben Design und Implementierung werden nicht berücksichtigt. Mit Hilfe des *Relationship Management Case Tool* (RMCase) (Diaz et al., 1995) können die verschiedenen Entwicklungsschritte der RM-Methode systemunterstützt durchgeführt werden.

3.1.2 Objektorientierte Vorgehensweisen

Zu den wesentlichen objektorientierten Vorgehensweisen gehört d*ie Object-Oriented Hypermedia Design Method* (OOHDM) (Rossi & Schwabe, 1994, 1995), die auf HDM aufbaut und über mehrere Jahre erweitert wurde (Schwabe & Rossi,

1998). OOHDM untergliedert den Entwicklungsprozess in vier Phasen: Das *konzeptuelle Design* befasst sich mit der Modellierung der Anwendungsdomäne und verwendet hierzu Konzepte der Object Modeling Technique (OMT) (Rumbaugh et al., 1991). Das Ergebnis ist ein Klassendiagramm, das die Anwendungsdomäne anhand von Klassen und Beziehungen zwischen diesen beschreibt. In der Phase des *Navigationsdesigns* werden die unterschiedlichen Sichten der potentiellen Besuchergruppen analysiert und dem konzeptuellen Modell zugeordnet. Die in dieser Entwurfsphase erstellte Navigationsstruktur enthält die Grundobjekte *Knoten*, *Links* und *Zugriffsstrukturen* zur Darstellung von Informationseinheiten, Beziehungen und Navigationsmechanismen. Das *Abstract Interface Design* legt in einem weiteren Schritt fest, wie die Navigationsstruktur dem Besucher der Website präsentiert wird und definiert die Interaktionsmöglichkeiten. Zu jedem Knoten wird ein Abstract Data View (ADV) (Cowan et al., 1993) entworfen, das die spätere Darstellung des Knotens ohne detaillierte Layoutinformationen definiert. In der Implementierungsphase werden die Navigationsmodelle sowie die Modelle der abstrakten Schnittstellen in konkrete Objekte der Zielplattform umgesetzt. Dennoch ist OOHDM zur Entwicklung von Web-Anwendungen nur bedingt geeignet, da die zugehörigen Dokumente keine einheitliche Definition der Modellierungskomponenten aufweisen und darüber hinaus oftmals eine unterschiedliche Notation verwendet wird. Zudem sind die Übergänge zwischen den einzelnen Phasen nicht ausreichend definiert.

Eine weitere objektorientierte Vorgehensweise stellt die Enhanced Object Relationship Method (EORM) (Lange, 1994, 1996) dar, die aufgrund ihres zugrunde liegenden iterativen Prozesses die frühzeitige Entwicklung und Verfeinerung eines Prototyps für das Benutzungsinterface vorsieht. In Analogie zu OOHDM verwendet EORM die Object Modeling Technique und erweitert diese zur Beschreibung von Interaktionen zwischen den typischen Objekten einer Hypermedia-Anwendung. Ebenso wie alle bislang vorgestellten Vorgehensweisen unterstützt EORM nur Ausschnitte des Unified Process. Während HDM lediglich das konzeptuelle und navigationale Design abdeckt und nur teilweise die Modellierung der Darstellung einbezieht, wird dieser Aspekt sowohl bei der RM-Methode als auch bei OOHDM und EORM betrachtet. Darüber hinaus berücksichtigen diese Vorgehensweisen Teilaspekte der Implementierung, bieten jedoch keine Unterstützung der Testphase sowie der vorgelagerten Phasen Anforderungsdefinition und Analyse.

Die Scenario-based Object-oriented Hypermedia Design Methodology (SOHDM) (Lee et al., 1998, 1999) weist Ähnlichkeiten zu OOHDM, EORM und RMM auf, unterscheidet sich jedoch einerseits durch den Gebrauch von Szenarien mit Hilfe von Scenario Activity Charts und andererseits durch eine erweiterte Unterstützung der Implementierungsphase. Für diese Vorgehensweise existieren keine softwarebasierten Werkzeuge.

Die hohe Bedeutung der *Unified Modeling Language* (UML) (Booch et al., 1999) im Bereich der Softwaretechnik hat zu einer Vielzahl von Vorschlägen geführt, die eine Erweiterung von UML auf der Grundlage des integrierten Erweiterungsmechanismus vorsehen (Baumeister et al., 1999; Conallen, 1999; Dolog & Bielikova, 2002; Fuentes et al., 2002; Vilain et al., 2000). Allerdings existieren unterschiedliche Auffassungen über die Verwendung von UML zur Modellierung von Web-Anwendungen. Reinhartz-Berger, Dori und Katz (Reinhartz-Berger et al., 2002) kritisieren anhand von Beispielen vor allem die Komplexität von UML, der sich ein Web-Entwickler in diesem Fall stellen muss. Ihre Analyse bestätigt die Untersuchungen von Siau und Cau (Siau & Cau, 2001), die zwar die Komplexität der einzelnen UML-Diagrammtypen als nicht höher im Vergleich zu Techniken anderer objektorientierter Methoden sehen, jedoch die Gesamtkomplexität von UML deutlich höher beurteilen.

Neben diesen Methoden existieren jedoch auch Vorgehensweisen, die alle im Unified Process definierten Phasen des Entwicklungsprozesses berücksichtigen. Hierzu zählt zum einen der Engineering-Ansatz von Lowe und Hall, der keine Modellierungstechnik festschreibt, und zum anderen die von Olsina (Olsina, 1998) vorgeschlagene Methode des *Hypermedia Flexible Process Modeling* (HFPM), die alle wesentlichen Phasen und Aktivitäten eines Hypermedia-Projektes abdeckt. Ebenso wie andere spezifische Methoden zur Hypermedia-Anwendungsentwicklung basiert auch HFPM auf keiner formalen Spezifikation (Koch, 2001).

3.1.3 Ontologiebasierte Vorgehensweisen

Einen ontologiebasierten Ansatz stellt die Methode OntoWebber (Jin et al., 2001, 2002) dar. Der Fokus dieser Methode liegt auf der Integration heterogener Datenbestände unter Verwendung einer Referenzontologie[1] und eignet sich zur Darstellung datenintensiver Websites. OntoWebber unterstützt die Phasen *Anforderungsanalyse*, *Design der Domänenontologie*, *Site View Design*, *Personalisierungsdesign* und *Wartungsdesign*, für die insgesamt sechs verschiedene Modelle bereitgestellt werden. Aktivitäten des *Site View Design* liegen in der Modellierung von Navigation, Inhalt und Darstellung mit Hilfe eines *Site View Graph*. Dieser stellt ein vereinfachtes konzeptuelles Modell zur Beschreibung von Hypertext zur Verfügung und enthält eine minimale Menge von Design-Primitiven. Die auf den einzelnen Webseiten darzustellenden Inhalte werden in zwei unterschiedlichen Kategorien von Karten organisiert. Statische Karten können alle Typen statischer Elemente wie Text, Bild etc. enthalten, während dynamische Karten definierte Attribute von Instanzen darstellen. Jede Karte kann nur je einen Inhaltstyp enthalten. Die Navigation erfolgt darauf aufbauend ausschließlich durch Links, die Karten und Seiten miteinander

[1] Eine Erläuterung des Begriffs *Ontologie* findet sich in Kapitel 4.2. Für eine Definition sei auf Kapitel 5.1 verwiesen.

verbinden können. Durch Instanzierung der entsprechenden Site-Views mit Daten aus dem Repository lässt sich unter Verwendung einer Query-Engine die Web Site generieren. Ebenso wie bei allen anderen bisher genannten Vorgehensweisen sind jedoch auch hier die Gestaltungsmöglichkeiten der Darstellungs- und Navigationskomponenten sehr eingeschränkt.

3.1.4 Weitere Vorgehensweisen

Neben den hier aufgeführten Methodologien existieren noch weitere Vorgehensweisen, die sich nicht in ER-basierte, objektorientierte bzw. ontologiebasierte Methoden einordnen lassen und von denen eine begrenzte Auswahl im Folgenden kurz vorgestellt werden soll.

Bei der *Web Modelling Language* (WebML) (Ceri et al. 2000) handelt es sich um einen Ansatz, der anhand grafisch erzeugter Modelle die Generierung einer plattformunabhängigen, XML-basierten (Bray et al., 2000) Repräsentation ermöglicht. Die Methode sieht die Erstellung eines *Strukturmodells* zur Beschreibung der Entitäten und Beziehungen, eines *Kompositionsmodells*, das die einzelnen Elemente einer Webseite festlegt, eines *Navigationsmodells*, eines *Präsentationsmodells* und eines *Personalisierungsmodells* vor. Um einen schreibenden Zugriff seitens des Besuchers der Website zu ermöglichen, wurde die Methode durch Elemente zur Dateneingabe sowie entsprechender Operationen erweitert (Bongio et al., 2000). WebML ist ausschließlich für die Bereitstellung statischer Inhalte ausgelegt, die in relationalen Datenbanken abgelegt sind. Die Präsentation dynamischer Inhalte wird nicht unterstützt. Für die Methode existiert ein CASE-Tool, das die Entwicklung von Web-Anwendungen in den einzelnen Phasen unterstützt[2].

Die dem *Web Site Design Model* (WSDM) (De Troyer & Leune, 1997) zugehörige Methode stellt einen benutzerzentrierten Ansatz dar. Ausgangspunkt der Modellierung ist eine Menge verschiedener Klassen von Besuchern, denen entsprechende Informationsbedürfnisse zugewiesen werden. Zuletzt sollen die *Structural Hypermedia Design Technique* (SHDT) und dessen Nachfolger, die *World Wide Web Design Technique* (W3DT) (Bichler & Nusser, 1996b) erwähnt werden, die eine Unterstützung des Rapid Prototyping in den Vordergrund stellen und in einen Forschungsprototypen umgesetzt wurden.

3.1.5 Bewertung

Trotz der Vielzahl unterschiedlicher Methodologien, die teilweise über mehrere Jahre hinweg weiterentwickelt wurden, ist der Einsatz systematischer Entwicklungsmethoden bislang wenig verbreitet. Untersuchungen wie die von Barry und Lang (Barry & Lang, 2001) zeigen, dass insbesondere im Medienbereich spezifi-

[2] http://www.webratio.com

sche Methoden kaum eingesetzt werden und Unternehmen stattdessen auf eigene Vorgehensweisen zurückgreifen bzw. den Aufbau der Anwendung ad hoc betreiben. Selbst Standardmethoden wie UML werden bislang nur selten im Web-Bereich angewendet.

Eine wesentliche Ursache für die geringe Akzeptanz liegt in der Komplexität der jeweiligen Entwicklungsmethoden, die in den meisten Fällen auch für Web-Entwickler mit besonderen technischen Kenntnissen schwer zu erlernen sind. Oftmals weist die Dokumentation der Methoden zudem Lücken auf, vor allem die Übergänge zwischen den einzelnen Prozessschritten betreffend, oder wird in verschiedenen Veröffentlichungen unterschiedlich beschrieben. Auch wenn die den Entwicklungsmethoden zugehörigen Entwurfsmodelle in der Regel eine abstrahierte und systematische Vorgehensweise unterstützen, orientieren sie sich insgesamt zu wenig an der tatsächlichen Darstellung der Website. In der Konsequenz sind die Modelle nur sehr eingeschränkt zur Kommunikation mit anderen Entwicklern bzw. dem Kunden oder Auftraggeber geeignet.

Zusätzlich werden nur wenige Methoden der hohen Änderungsdynamik gerecht. Oftmals besitzen die vorgeschlagenen Vorgehensweisen ein Präsentationsmodell, das eine feingranulare Zerlegung einer Webseite nicht erlaubt. In der Folge ist die Aktualisierung von Teilbereichen einer einzelnen Seite nur sehr eingeschränkt möglich. Während einige Methoden durch entsprechende Softwareunterstützung einen dynamischen Seitenaufbau zur Laufzeit erlauben und somit Änderungen im Informationsbestand automatisch berücksichtigen, erfordern andere Methodologien den manuellen und folglich statischen Aufbau der Webseiten anhand der konzipierten Modelle.

Ein weiteres Defizit existierender Vorgehensweisen liegt in der Konzeption der für die Darstellung der Inhalte vorgesehenen Komponenten. Die Granularität der Komponenten ist durchweg nicht ausreichend, um der Flexibilität der Navigation und der Präsentation vor allem mehrerer gleichartiger Inhalte gerecht zu werden. Beispielsweise verfügen vorhandene Vorgehensweisen über keinen Mechanismus, Links auf konkrete Textabschnitte zu modellieren (Woukeu et al., 2003).

Bei der Einordnung der verschiedenen Vorgehensweisen in die einzelnen Phasen des Unified Process wird deutlich, dass die meisten Methodologien eine durchgängige Unterstützung des Entwicklungsprozesses bzw. die Betrachtung des gesamten Lebenszyklus der Anwendung vermissen lassen (vgl. Koch, 1999). Eine Ursache ist darin zu sehen, dass kein einheitliches Verständnis über die zu betrachtenden Aspekte einer Entwurfsmethode existiert. Rumbaugh (Rumbaugh, 1995) schlägt beispielsweise vier Komponenten vor, die eine Methodologie zur Systementwicklung enthalten sollte: eine Menge von Modellierungskonzepten zur Erfassung des Wissens über das Problem und dessen Lösung, eine Menge von Sichten

zur Darstellung der zugrunde liegenden Modellinformationen, einen iterativen Prozess zur Konstruktion und Implementierung der Modelle sowie eine Sammlung von Hinweisen und Regeln für die Entwicklung. Henderson-Sellers (Henderson-Sellers, 1995) hingegen betrachtet eine Vorgehensweise als eine erweiterte Sammlung aus folgenden Entwurfsaspekten und zugehörigen Komponenten: einem vollständigen Lebenszyklus, einer vollständigen Menge von Konzepten und Modellen, einer Sammlung von Regeln und Richtlinien, einer vollständigen Beschreibung der Ergebnisse, einer arbeitsfähigen Notation, einer Menge von Metriken, zusammen mit Hinweisen zu Qualität, Standards und Test-Strategien, verschiedenen Richtlinien zum Projektmanagement, Hinweisen zum Bibliotheksmanagement und zur Wiederverwendbarkeit, sowie der Identifikation organisationaler Regeln.

Schließlich ist die zu geringe CASE-Unterstützung fast aller Vorgehensweisen zu erwähnen, die sich zudem vorrangig auf das Design der Anwendung und weniger auf den dahinter stehenden Design-Prozess konzentrieren. Falls softwarebasierte Werkzeuge zur Unterstützung der Vorgehensweise existieren, sind diese darüber hinaus nicht ausreichend in den Entwicklungsprozess integriert (Lowe & Hall, 1999).

3.2 Werkzeuge für die Entwicklung von Web-Anwendungen

3.2.1 Kategorien softwarebasierter Werkzeuge

Parallel zur vorrangig im Forschungsbereich stattgefundenen Konzeption modellbasierter Vorgehensweisen, wurde eine Reihe softwarebasierter Werkzeuge zur Web-Anwendungserstellung für den Einsatz in der betrieblichen Praxis entwickelt. Im Kern gilt nach wie vor die von Fraternali vorgenommene Kategorisierung der Unterstützungswerkzeuge (Fraternali, 1999), die jedoch entsprechend Tabelle 3-1 durch sog. Web-Content-Management-Systeme zu ergänzen ist. Diese Systeme werden seit einigen Jahren insbesondere für den Aufbau größerer Internet- bzw. Intranet-Sites verwendet und erlauben die Berücksichtigung unterschiedlicher Entwicklerrollen, wie beispielsweise Autor oder Redakteur.

Im Folgenden sollen die Unterstützungswerkzeuge der unterschiedlichen Kategorien vor dem Hintergrund der definierten Anforderungen zusammengefasst werden. Eine ausführliche Diskussion der einzelnen Kategorien findet sich in dem durch Fraternali verfassten Überblick.

Die Werkzeuge der ersten Kategorie, die sich aus visuellen HTML-Editoren und Site-Managern zusammensetzt, konzentrieren sich auf das Design, die Implementierung und Wartung von Web-Anwendungen und bieten vielseitige Kontrollmöglichkeiten bzgl. der grafischen Ausgabe. Allerdings erfolgt der Seitenaufbau, dessen Inhalte statisch gebunden sind, Datei-basiert. Werkzeuge

Kategorie	Eigenschaften	Werkzeuge (Beispiele)
Visuelle HTML-Editoren und Site Managers	- Visuelle Erstellung und Bearbeitung von Webseiten, - Implementierung und Wartung von Websites, - Statischer Seitenaufbau	NetObject Fusion, Microsoft Frontpage
Webfähige Hypermedia Authoring Werkzeuge	- Offline Hypermedia Publishing - Fokussierung von Navigation und Präsentation - Datenbank-Anbindung	Macromedia Director, Allen Communication Quest, Asymetrix Toolbook II Assistant / Instructor
Web-DBPL-Integratoren	- Datenbank-basierter Aufbau der Web-Anwendung - hohe Skalierbarkeit - geringe Kontrollmöglichkeiten der grafischen Ausgabe	Oracle PL/SQL Web Toolkit, Sybase PowerBuilder Web.PB class library
Web-Formular Editoren, Bericht-Generatoren und Database Publishing Wizards	- Datenbank-zentrierter Ansatz - Unterstützen schelle Entwicklung datenintensiver Anwendungen - Formular-basierter Aufbau der Anwendung	Microsoft Visual Basic, Microsoft Access, Microsoft Visual InterDev, Oracle Developer 2000, Sybase PowerBuilder, Allaire Cold Fusion Application Wizards
Web Content Management-Systeme	- Trennung von Inhalt, Struktur und Darstellung - Redaktionsprozess mit unterschiedlichen Benutzerrollen - hohe Kontrollmöglichkeiten der grafischen Ausgabe	Vignette StoryServer
Modellbasierte Web-Generatoren	- Höchster Automatisierungsgrad und Lebenszyklus-Abdeckung - Generierung der Web-Applikation auf Basis der Erstellung eines konzeptuellen Modells	Oracle Web Development Suite, WebRatio

nach: Fraternali, 1999

Tabelle 3-1: Kategorien softwarebasierter Werkzeuge zur Web-Applikationsentwicklung [3]

dieser Kategorie eignen sich hauptsächlich für kleine bis mittlere Anwendungen ohne Datenbank-Anbindung.

Die zweite Gruppe von Werkzeugen, sog. webfähige Hypermedia-Authoring-Tools, unterstützt vor allem die Implementierung langlebiger Applikationen, die eine hohe Qualität hinsichtlich ihrer Benutzungsschnittstellen erfordern und nur seltenen Änderungen unterworfen sind. Zumeist erfolgt die Erstellung der Anwendung ad hoc und unterliegt keiner systematischen Vorgehensweise.

[3] Eine Übersicht der Werkzeuge findet sich im Anhang A.

Im Unterschied zu den bisher genannten Werkzeugkategorien basieren Web-DBPL-Integratoren auf einer Trennung von Informationsbestand und Webseiten. Es handelt sich hierbei jedoch weniger um tatsächliche Web-Entwicklungswerkzeuge als vielmehr um Programmierumgebungen.

Web-Formular-Editoren, Bericht-Generatoren und Database-Publishing-Wizards eignen sich zur schnellen Entwicklung datenintensiver Web-Anwendungen, die auf traditionellen Konzepten des Datenbank-Designs beruhen. Sie bieten in der Regel jedoch kaum Möglichkeiten, die grafische Darstellung der Benutzungsschnittstelle zu beeinflussen.

Diesem Mangel begegnen Content-Management-Systeme, die vielseitige Möglichkeiten zur Anpassung der grafischen Darstellung zur Verfügung stellen und ebenfalls auf der Trennung der Präsentation von den eigentlichen Inhalten und ihrer Struktur basieren. Allerdings konzentrieren sich diese Systeme vorrangig auf den Redaktionsprozess und bieten keine Unterstützung bei der Konzeption und Modellierung der Anwendung. Darüber hinaus unterstützen Content-Management-Systeme nur begrenzt die Entwicklung von Anwendungen, deren Datenmodell und Navigationsdesign komplexe Strukturen und Relationen aufweist.

Die letzte Gruppe bilden modellbasierte Web-Generatoren, die den Lebenszyklus von Web-Anwendungen am umfangreichsten abdecken. Im Gegensatz zu den Werkzeugen der anderen Kategorien werden die Anwendungen zunächst auf der konzeptuellen Ebene modelliert und anschließend durch automatische Quelltextgenerierung anhand der vom Benutzer erstellten grafischen Entwurfsmodelle implementiert. Bislang existieren jedoch nur sehr wenige kommerzielle Software-Werkzeuge, die dieser Gruppe zugeordnet werden können. Lediglich die Oracle Web Development Suite (Barnes & Gwyer, 1996), deren Schwächen vor allem in der Gestaltung der grafischen Ausgabe und insgesamt mangelnden Anpassungsmöglichkeiten liegt, sowie das auf WebML basierende Tool WebRatio bilden die Ausnahme.

3.2.2 Forschungsprototypen

Wenngleich kaum kommerzielle Werkzeuge zur systematischen und modellbasierten Entwicklung von Web-Anwendungen existieren, wurden bislang einige Forschungsprototypen entwickelt, die z. T. auf den vorgestellten Methodologien basieren. Eine ausführliche Diskussion der verschiedenen Web-Entwicklungswerkzeuge findet sich in den Arbeiten von Christodoulou (Christodoulou et al., 1998) und Fraternali (Fraternali, 1999). Einige ontologiebasierte Unterstützungswerkzeuge werden in einem technischen Bericht von Woukou, Carr, Wills und Hall (Woukou et al., 2003) zusammengefasst. Die entwickelten Prototypen konzentrieren sich zumeist auf unterschiedliche Aspekte des Web-Engineering-Prozesses und weisen insge-

samt Lücken hinsichtlich der systemtechnischen Unterstützung des gesamten Lebenszyklus auf. Im Folgenden werden die wichtigsten Systeme kurz vorgestellt.

Das *Araneus*-System (Atzeni et al., 1998a; Atzeni et al., 1998b) definiert eine Umgebung zur integrierten Betrachtung strukturierter und unstrukturierter Web-Inhalte und unterstützt sowohl die Erstellung eines konzeptuellen Modells als auch den Design-Prozess der Präsentation. Nachteile des Systems liegen jedoch in einer proprietären HTML-basierten Template-Sprache zur Spezifikation der Darstellung und in der fehlenden Möglichkeit, Applikationslogik in die Anwendung zu integrieren.

Eine Sammlung von Werkzeugen zur modellbasierten Entwicklung von Web-Anwendungen stellt das System *Autoweb* (Fraternali & Paolini, 2000) dar, aus dem die Konzeption der WebML-Vorgehensweise entstanden ist. Die Nachteile der WebML-Methodologie bzgl. der fehlenden dynamischen Content-Generierung gelten für dieses System ebenso, wie für das kommerzielle System *WebRatio*.

Zuletzt soll das Web-Management-Tool *Strudel* (Fernandez et al., 1998) erwähnt werden, dessen Kernidee in der Trennung von Inhalt, Struktur und Darstellung besteht. Der Seitenaufbau erfolgt mit Hilfe einer eigenen Abfragesprache (StruQL). Allerdings berücksichtigt auch dieses System keine dynamische Content-Erzeugung und erfordert darüber hinaus die manuelle Erzeugung von Skripten zur Einbindung der Inhalte.

3.2.3 Bewertung

Anhand der geringen Anzahl kommerziell verfügbarer Systeme zur modellbasierten Entwicklung von Web-Applikationen trotz einer Vielzahl existierender Prototypen zeigt sich der nach wie vor vorhandene Forschungsbedarf in diesem Themenfeld. Dieser bezieht sich in erster Linie auf die den softwarebasierten Werkzeugen zugrunde liegenden Methodologien, als auch auf den Automatisierungsgrad der Systeme. Vor allem in Bezug auf die Verständlichkeit und Anwendbarkeit, auch gegenüber Benutzern ohne besondere technische Kenntnisse, besteht immenser Verbesserungsbedarf, bevor modellbasierte Entwicklungswerkzeuge in der Praxis eingesetzt werden können.

Einen Ansatzpunkt zur systematischen Vorgehensweise bei der Entwicklung webbasierter Anwendungen bieten die bereits vielfach eingesetzten Web-Content-Management-Systeme, da sie umfangreiche Mechanismen zur Handhabung von Inhalten und zur feingranularen Strukturierung einzelner Webseiten zur Verfügung stellen. In Analogie zu den meisten Methodologien werden Inhalt, Struktur und Präsentation getrennt voneinander betrachtet. Allerdings berücksichtigen diese Systeme nur einen Ausschnitt des Lebenszyklus webbasierter Anwendungen, indem sie sich vornehmlich auf das Design der grafischen Darstellung, die Einbin-

dung und Aufbereitung der Inhalte und auf das Testen der Anwendung beschränken. Zumindest aus technischer Sicht stellen Web-Content-Management-Systeme eine in der betrieblichen Praxis getestete Grundlage für Erweiterungen in Bezug auf Werkzeuge zur Modellierung der Informationsbasis, der Navigationsstruktur sowie der unterschiedlichen Sichten für die verschiedenen Rollen und Benutzerinteressen dar, die zur modellbasierten Generierung von Inhalten geeignet ist.

4 Eine Vorgehensweise zur ontologiebasierten und komponentenorientierten Modellierung von Web-Anwendungen

Entscheidend für das Erreichen der in Kapitel 2.3 definierten Zielsetzungen im Zusammenhang mit der Entwicklung von Websites ist eine systematische Vorgehensweise, die den Erstellungsprozess in einzelne Phasen unterteilt. Auf der Grundlage eines durchgängigen Prozessmodells werden die unterschiedlichen Entwurfsbereiche der Website-Entwicklung den definierten Prozessschritten zugeordnet. Durch die Zerlegung des Gesamtprozesses in einzelne Teilschritte kann der Komplexität der Web-Anwendungsentwicklung begegnet werden, da dem Entwickler ermöglicht wird, sich auf einzelne Phasen des Entwurfs zu konzentrieren, deren Ergebnisse verifizierbar sind und im Einzelnen überarbeitet werden können.

Verschiedene Prozessmodelle für die Entwicklung von Websites wurden in den vergangenen Jahren vorgeschlagen, wie z. B. von Ginige (Ginige, 1998), Gellerson und Gaedke (Gellerson & Gaedke, 1999) und Koch (Koch, 1998). Gemein ist diesen Modellen eine übereinstimmende Berücksichtigung der Phasen *Anforderungsanalyse*, *Domänenmodellierung*, *Design*, *Implementierung* sowie *Test und Wartung*. Unterschiede existieren jedoch im Hinblick auf Detaillierung und Umfang der Aktivitäten in den einzelnen Prozessphasen, so dass an dieser Stelle, entsprechend Abbildung 4-1, ein Modell für den Entwicklungsprozess von Web-Anwendungen aufgezeigt werden soll, das alle relevanten Entwurfsbereiche und Prozessschritte abdeckt.

Ausgangspunkt ist eine Analyse grundsätzlicher Faktoren. Hierzu zählen die Definition der Ziele, die mit der Website erreicht werden sollen, die Identifikation relevanter Nutzergruppen, die Untersuchung der gegebenen Rahmenbedingungen und die Festlegung der strategischen Ausrichtung. In einem weiteren Schritt werden Inhalt und Funktionalität der Website spezifiziert und erste Richtlinien für die Strukturierung der Informationen sowie Bestimmungen zur Ergonomie festgelegt. Darauf aufbauend erfolgt der Entwurf verschiedener Modelle, die sich neben der Kategorisierung des Informationsbestands sowohl mit dem strukturellen Aufbau und dem Design der Website als auch mit der Definition der unterschiedlichen Benutzerrollen befassen. Die Modelle stellen den Ausgangspunkt für das Prototyping bzw. die Implementierung der Web-Anwendung dar und werden in einem iterativen Prozess in Bezug auf qualitative und ergonomische Merkmale verifiziert und ggf. optimiert.

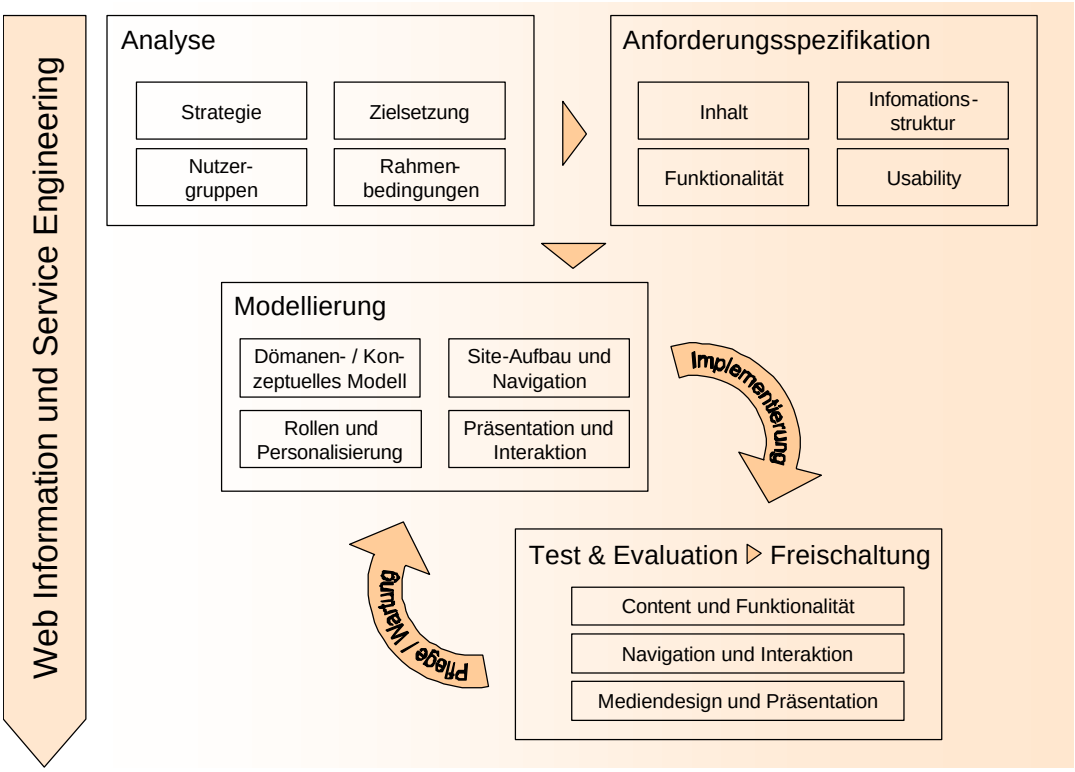


Abbildung 4-1: Phasen im Web Engineering

4.1 Entwurfsbereiche der Modellierungsphase

Für den systematischen Aufbau einer Web-Anwendung erweist sich eine Zerlegung der Modellierungsphase in unterschiedliche Entwurfsbereiche als sinnvoll. Dies erlaubt dem Entwickler, die verschiedenen Aspekte einer Web-Applikation getrennt voneinander zu betrachten, so dass eine methodische Vorgehensweise gefördert wird.

Zur einheitlichen Beschreibung des Modellierungsprozesses ist die Bereitstellung durchgängiger, umfassender und integrierter Modellierungsmethoden für die im Web-Kontext relevanten Entwurfsbereiche notwendig. Hierzu wird für jeden Bereich ein Modell erstellt, das einen bestimmten Aspekt der Web-Anwendung abstrahiert. Die einzelnen Modelle, die sich auf verschiedene Ebenen der Anwendung beziehen, ermöglichen den systematischen Aufbau der einzelnen Webseiten unter Einbeziehung des strukturellen Aufbaus, des Designs sowie der Definition rollenbasierter Zugriffe und bilden den Ausgangspunkt für die Implementierung der Anwendung. Die Konzepte und Techniken für die verschiedenen Modellierungsaspekte zielen auf eine systemunterstützte automatisierte Weiterverarbeitung und werden in einem Methodenverbund bereitgestellt, der folgende Entwurfsbereiche umfasst:

Die *konzeptuelle Ebene* analysiert vorhandene Strukturen eines Ausschnittes der Realität und dient der Erstellung von Katalogen, bestehend aus Objekten und Beziehungen. Hierbei handelt es sich in erster Linie um die Betrachtung von Ressourcen und Prozessabläufen, die in Form von Dokumenten vorliegen. Die ontologische Beschreibung des Informationsbestandes innerhalb eines konzeptuellen Modells, die vorhandene Konzepte und Relationen der zugrunde liegenden Ontologie (Lenat & Guha, 1990; Neches et al., 1991) als Themenbausteine einbezieht, dient als Grundlage für den systematischen Entwurf einer Navigations- und Sichtenstruktur. Darauf aufbauend beinhaltet die *navigationale Ebene* eine Beschreibung der darzustellenden Inhalte sowie der Navigationsstrukturen, die den Zugang zu den Informationen definieren. Das daraus resultierende Navigationsmodell wird innerhalb der *Sichtenebene* um die Definition der tatsächlich darzustellenden Inhalte in Abhängigkeit verschiedener Nebenbedingungen, wie beispielsweise die Rolle des Benutzers, seine Interessen, Ortsinformationen etc., erweitert. Zuletzt wird die grafische Präsentation der Inhalte in der *Präsentationsebene* festgelegt.

Aus der Betrachtung unterschiedlicher Ebenen in Verbindung mit der Generierung verschiedener Modelle am Ende jeder Phase ergeben sich Vorteile für den Entwickler bzgl. Aufbau und Wartung der Web-Anwendung. Im Hinblick auf Möglichkeiten zur Unterstützung der Kooperation und Koordination der an der Entwicklung beteiligten Akteure lassen sich frühzeitig Teilergebnisse kommunizieren. Darüber hinaus bestehen Verifikationsmöglichkeiten am Ende jeder Phase. Aufgrund der Trennung von Inhalt, Navigation und Darstellung ist es möglich, die zugrunde liegenden Daten unabhängig von Navigationspfaden oder Darstellungsmerkmalen zu warten, ohne die Web-Anwendung neu implementieren zu müssen. Dies gilt für alle Entwicklungsbereiche der Web-Applikation. Inhalt, Navigation und Präsentation können sowohl zur Laufzeit (z. B. für Fein-Tuning) als auch beim Redesign unabhängig voneinander modifiziert werden.

Die Modellierungsphase einer Web-Anwendung setzt sich demnach aus folgenden Schritten zusammen: Entwurf des Metadatenmodells (Themenstruktur), Aufbau des Navigationsmodells, formale Analyse der Benutzeranforderungen, Ableiten der Sichtenstruktur zur Modellierung personalisierter bzw. rollenbasierter Darstellungen auf der Grundlage des Navigationsmodells, Konfiguration externer Dienste und Definition der grafischen Darstellung und Interaktion (vgl. Abbildung 4-2). Die jeweiligen Metamodelle, die sich auf die verschiedenen Entwurfsbereiche einer Web-Anwendung beziehen, ermöglichen aufgrund ihrer Konzeption eine weitestgehend automatisierte Weiterverarbeitung, die eine systemunterstützte Prototypengenerierung der Web-Applikation einschließt. Dem Entwickler stehen hierdurch Methoden zur Verfügung, mit denen er die von ihm entworfenen Modelle anhand des erstellten Prototyps in Bezug auf Inhalt und Funktionalität prüfen sowie die Effizienz der Struktur, Navigation und Präsentation evaluieren kann. Die entsprechenden Unter-

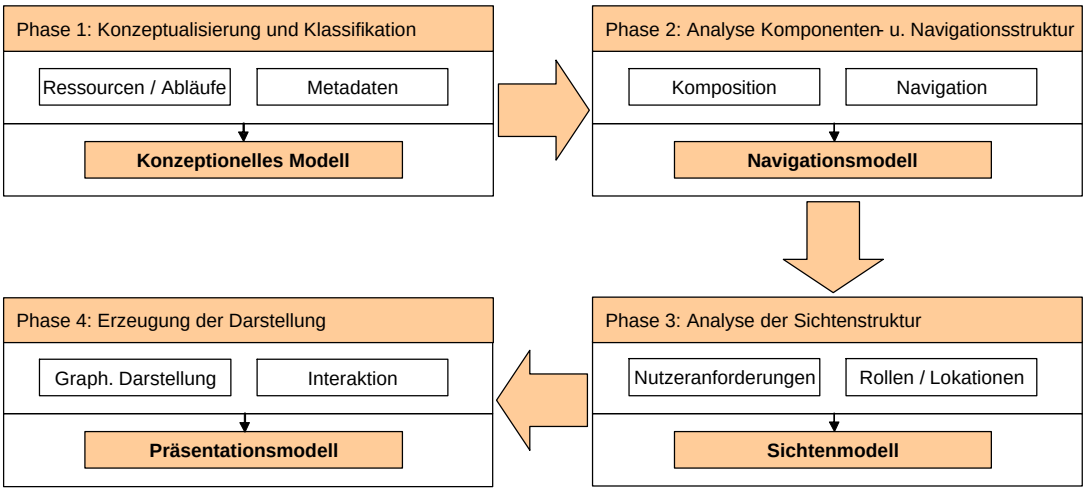


Abbildung 4-2: Entwurfsbereiche innerhalb der Modellierungsphase

suchungen können sich dabei auf einzelne Modelle beschränken, so dass im Sinne einer iterativen Vorgehensweise Teilbereiche des Modellierungsprozesses evaluiert und optimiert werden können.

4.2 Ontologien bei der Entwicklung von Web-Anwendungen – Methoden der konzeptuellen Ebene

Webbasierte Informationsangebote und Anwendungen stützen sich in der Regel auf Datenbestände, die in dem jeweiligen Unternehmen vorhanden sind oder von Dritten angeboten werden und bauen auf die zugrunde liegenden Organisationsstrukturen und Abläufe auf. Veränderungen an diesen Datenbeständen, Aufgaben oder Prozessen im Unternehmen bewirken im Allgemeinen auch eine Modifikation der in der zugehörigen Web-Anwendung dargestellten Informationen. Aus diesem Grund ist eine enge Kopplung zwischen den vorhandenen Ressourcen und der darauf aufbauenden Web-Applikation sinnvoll.

Ausgangspunkt für die Gestaltung von Web-Anwendungen ist daher die Analyse der Organisationsstrukturen und Abläufe in Unternehmen und die Herleitung entsprechender Begriffssysteme. Ziel ist eine formale Repräsentation des im Unternehmen vorhandenen Wissens, die auf einer *Konzeptualisierung* des Informationsbestandes aufbaut. Die Konzeptualisierung liefert Objekte, Konzepte und andere Entitäten einschließlich ihrer Beziehungen zueinander und stellt eine formalisierte, abstrakte und vereinfachte Sicht auf einen Ausschnitt der Realität dar. Bei einer Ontologie handelt es sich um eine explizite Spezifikation einer Konzeptualisierung (Gruber, 1993). Sie erlaubt eine konzeptuelle Strukturierung und Modellierung einer

Domäne und ermöglicht ein gemeinsames Verständnis der Strukturen, in denen Wissen abgelegt wird. Aus technischer Sicht geht es vor allem um den Anspruch, das vorhandene Wissen aus einem Gebiet derart darzustellen, dass es durch möglichst viele Programme verarbeitet werden kann.

Üblicherweise entsprechen die Konzepte, die aus einer Analyse der Organisationsstruktur und der Abläufe in einem Unternehmen hervorgehen, zumindest in Teilen der in der Website darzustellenden Kernthemen. So finden sich wesentliche Themenbereiche und Kompetenzen eines Unternehmens im Allgemeinen auch in dessen Web-Anwendung wieder. Beim Aufbau einer Website werden bei einer systematischen Vorgehensweise im ersten Schritt die wesentlichen Themenschwerpunkte und Kernbereiche im Unternehmen identifiziert und strukturiert. Hierzu eignen sich Ontologien in besonderer Weise, da sie im Sinne eines systematischen Wissensmanagements die Strukturierung des sprachlich verfassten Domänenwissens zulassen. Neben der formalsprachlichen Modellierung von Ausschnitten der Realität berücksichtigt die Semantik einer Ontologie ebenso die Erfassung natürlichsprachlicher Ausschnitte. Hierzu können innerhalb einer Ontologie semantische Regeln spezifiziert werden.

4.3 Methoden der navigationalen Ebene

Auch wenn sich Konzepte der Ontologie innerhalb einer Website wieder finden, werden bestenfalls nur Teile der ontologischen Relationen in die Web-Anwendung übernommen. In vielen Fällen ist es nicht wünschenswert, vorhandene Strukturen im Unternehmen in der Web-Applikation unmittelbar abzubilden. Es ist daher notwendig, eine von der Ontologie unabhängige Navigation über die Themen der Ontologie zu erstellen. Weiterhin müssen Informationen, die aus verschiedenen Informationsquellen stammen können, häufig parallel auf einer Webseite dargestellt werden. Aus diesen Gründen werden auf Basis der Ontologie Komponentenmodelle für die einzelnen Webseiten erstellt, die dann, zusammen mit einer übergeordneten Navigationsstruktur, das Navigationsmodell bilden. Die Komponentenmodelle definieren die Zusammensetzung jeder einzelnen Webseite aus unterschiedlichen Komponenten. Das zugrunde liegende Metamodell stellt eine Mindestmenge an Komponenten zur Verfügung, die zur Beschreibung aller wesentlichen Funktionen und Eigenschaften einer Webseite benötigt werden. Navigierbare Beziehungen zwischen den Inhalten der verschiedenen Komponenten werden mit sog. Navigationsrelationen beschrieben. Hieraus ergibt sich das Navigationsmodell.

Die grafische Beschreibung des Navigationsmodells, das vom Website-Entwickler zeichnerisch erstellt wird, basiert auf der Statechart-Notation (Harel, 1987). Diese stellt grundlegende Mechanismen zur Darstellung ineinander verschachtelter Komponenten zur Verfügung, so dass Ebenen entstehen, die die Navigationstiefe inner-

halb einer Website widerspiegeln. Komponenten derselben Ebene lassen sich entweder konjunktiv oder disjunktiv darstellen. Zusätzlich erlauben sie die Zuordnung verschiedener Zustände, so dass sich Zustandsveränderungen durch Übergangsrelationen definieren lassen. Hierdurch ist es möglich, die Navigationsstruktur einer Website mit entsprechender Strukturierung der Inhalte, ähnlich dem Erscheinungsbild der späteren Web-Anwendung zu modellieren. Dies stellt einen wesentlichen Unterschied zu existierenden Methodologien im Kontext der Website-Entwicklung dar, die eine vom strukturellen Aufbau der Inhalte getrennte Navigationsmodellierung vorsehen. Im Vergleich zu anderen Notationen wie UML, eignet sich der Statechart-Formalismus daher als Grundlage für die Beschreibung eines Navigationsmodells, das den Anforderungen der zugleich inhaltsorientierten und grafischen Darstellung entspricht. Darüber hinaus können durch die Betrachtung einzelner Komponenten Teile des Entwurfs wieder verwendet werden.

4.4 Methoden der Sichtenebene

Auf dieser Ebene wird definiert, auf welche konkreten Inhalte ein Nutzer in Abhängigkeit verschiedener Nebenbedingungen zugreifen kann bzw. darf. Zu diesen Nebenbedingungen gehören beispielsweise sein Interessenprofil, seine Rolle, der Aufenthaltsort oder das verwendete Gerät. Ziel der Sichtenebene ist es, durch eine rollenangepasste Darstellung der Web-Anwendung, den unterschiedlichen Informationsbedürfnissen der verschiedenen Benutzergruppen gerecht zu werden. Hierzu werden verschiedene Kontextfaktoren herangezogen, die technische Rahmenbedingungen, natürliche Faktoren wie Zeit und Ort und die soziale Umgebung beschreiben. Mit Hilfe der Sichtenebene können diese Faktoren in die Darstellung einzelner Bereiche und Inhalte einer Website einbezogen werden.

4.5 Methoden der Präsentationsebene

Anhand des Navigationsmodells werden der strukturelle Aufbau der einzelnen Seiten und ihre Inhalte, sowie die Struktur der Web-Anwendung im Gesamten festgeschrieben. Darauf aufbauend definieren im Hinblick auf eine personalisierte bzw. rollenbasierte Darstellung Zugriffsklassen eine Auswahl der zu präsentierenden Inhalte. Jedoch werden in den einzelnen Modellen bisher keine Aussagen über die genaue grafische Darstellung der einzelnen Komponenten und Inhalte getroffen, um den Anwender bei der Erstellung einer Web-Applikation zunächst nicht mit Details der Anwendung zu konfrontieren.

Die Präsentationsebene definiert nun unabhängig vom eigentlichen Content die Details der grafischen Darstellung. Auf dieser Modellierungsebene werden für jede Komponente Eigenschaften wie Position, Größe, Farbe etc. definiert. Des Weiteren

legt das Präsentationsmodell sowohl die Art als auch die Gestaltung der Interaktionskomponenten fest.

4.6 Modellbasierte Unterstützung bei der Implementierung von Websites

In der Praxis zeigt sich, dass die Verwendung zeichnerischer Modelle den Entwurfsprozess erleichtert. Die Bereitschaft des Entwicklers, derartige Modelle zu verwenden, hängt jedoch stark von der systemtechnischen Unterstützung bzw. dem Vorhandensein entsprechender Werkzeuge ab (Szwillus & Bomsdorf, 2002). Infolge dessen ist neben der grafischen Modellierung der den einzelnen Entwurfsbereichen zugeordneten Modelle eine formale Spezifikation erforderlich. Anhand dieser wird eine weitere systemunterstützte Verarbeitung ermöglicht, deren Ergebnis die Generierung der Website ist.

Die Konzeption der in dieser Arbeit vorgestellten Metamodelle fokussiert eine weitestgehend automatisierte Verwendung der vom Website-Entwickler zeichnerisch erstellten Modelle. Mit Hilfe der den Metamodellen zugrunde liegenden formalen Spezifikation lassen sich diese in ein XML-basiertes Austauschformat überführen. Auf dieser Grundlage können zur Laufzeit der Anwendung die Navigationsstruktur, die Sichten auf die festgelegten Inhalte und die Präsentation generiert werden.

4.7 Ontologien in der Anwendung von Websites

Neben der kontextbezogenen Suche lassen sich Informationen mit Hilfe definierter Regeln ableiten. Solche Inferenzregeln innerhalb einer Ontologie erlauben es, Schlussfolgerungen zu ziehen, die zwar plausibel und nachvollziehbar sind, jedoch nicht zwingend wahr sein müssen. Hierdurch lässt sich aus Dokumenten, die nur implizites Wissen enthalten, explizites Wissen erschließen (vgl. Zelewski, 2002).

Anhand der in einer Ontologie dargestellten Informationen können neue Informationen abgeleitet werden. Dieser als *Reasoning* bzw. *Inferenz* bezeichnete Vorgang stellt eine zentrale Komponente bei der Wissensrepräsentation dar, die Rückschlüsse beispielsweise darüber erlaubt, welche Informationen an welcher Stelle und zu welcher Zeit für einen Anwender von Interesse sein könnten oder welche Aufgabe in einem Prozess als nächstes bearbeitet werden sollte.

Als ein einfaches Beispiel könnte die Tatsache, dass ein bestimmter Mitarbeiter an einem Projekt X gearbeitet hat und dieses Projekt X sich mit dem Thema Y befasst hat, den Rückschluss zulassen, dass dieser Mitarbeiter sich in dem Thema Y auskennt, ohne dass diese Aussage explizit in der Wissensbasis enthalten ist:

$\forall$ mitarbeiter, projekt, thema hatKenntnis(thema) $\leftarrow$
mitarbeiter:Mitarbeiter[arbeitetAn ->> projekt] $\wedge$
projekt:Projekt[keywords ->> thema][4].

Hinsichtlich der Entwicklung von Webseiten lassen sich die in Kapitel 2.3 aufgezeigten Anforderungen bzgl. der Berücksichtigung von Kontextfaktoren dadurch erfüllen, dass konkrete Regeln für jede Bedingung erstellt werden. Diese Regeln, die sich in der Phase der Anforderungsanalyse auf Grundlage der aus der Betrachtung der Kontextfaktoren resultierenden Nebenbedingungen ergeben, lassen sich im Navigationsmodell bzw. im Sichtenmodell integrieren.

Die Kategorisierung des Informationsbestandes und die Zuordnung von konkreten Informationsobjekten in Form von Instanzen werden darüber hinaus zur Strukturierung der Informationsobjekte anhand sog. navigationaler Klassen genutzt. Der Eigenschaft, dass Klassen und Objekte in einer hierarchischen Struktur vernetzt gehalten werden, bedient sich eine navigationale Klasse, indem sie in verschiedenen Darstellungsformen (Baum, Sequenz, Liste etc.) die zu einem oder mehreren Konzepten zugehörigen Objekte darstellt.

[4] Ausdruck in Frame Logic (kurz: F-Logic) (Kifer & Lausen, 1989)

5 Konzeptualisierung und Klassifikation von Informationsbeständen

Der Entwurf von Web-Anwendungen basiert auf einer Analyse der Informationsinhalte, die in verschiedenen Ressourcen vorliegen und aus unterschiedlichen Datenquellen stammen. Zielsetzung bei der Erstellung eines konzeptuellen Schemas, das diese Informationsinhalte unabhängig von ihren Speicherstrukturen beschreibt, ist es, die Semantik der Domäne möglichst ohne die Berücksichtigung navigationaler Aspekte sowie der späteren Präsentation zu erfassen. Hierzu werden durch Betrachtung gemeinsamer Eigenschaften der Informationsinhalte zunächst Klassen gebildet, anhand derer eine Kategorisierung der Ressourcen vorgenommen werden kann. Darüber hinaus ist eine Analyse der Anforderungen aus Sicht der potentiellen Benutzer hilfreich. Die Anforderungsspezifikation ermöglicht es, basierend auf einer Beschreibung von Anwendungsfällen (Use Cases), Aktivitäten der Benutzer zu definieren und zugehörige Objekte zu identifizieren. Beim Entwurf von Web-Anwendungen ist die Spezifikation der Anforderungen von hoher Bedeutung, da sie über den Erfolg der zu entwickelnden Web-Anwendung entscheidet. Sie stellt somit den ersten Schritt bei der Entwicklung von Web-Applikationen dar und dient als Grundlage für die Schwerpunktbildung bei der Konzeptualisierung des Informationsbestandes.

Für die einzelnen Klassen werden zugehörige Attribute bzw. Eigenschaften und Operationen ermittelt, die diese beschreiben. Zusätzlich werden Relationen zwischen den Klassen definiert, die beliebige Beziehungen darstellen können. Bei dem auf diese Weise entstandenen konzeptuellen Modell handelt es sich um die Konzeptualisierung des Informationsbestandes, die eine Beschreibung der Informationsobjekte in Form von Dokumenten, Aufgaben, Prozessen etc. einschließlich deren Assoziationen enthält.

Die Erstellung eines konzeptuellen Modells kann neben der reinen Informationsklassifizierung auch die Analyse von Aufgaben und Prozessen beinhalten. Ansätze zur Aufgabenmodellierung im Kontext interaktiver Web-Applikationen, die sich auf die Modellierung der Aufgabendurchführung (Uhr, 2001; Biere et al., 1999) beziehen, wurden von Szwillus und Bomsdorf vorgeschlagen (Szwillus & Bomsdorf, 2002). Hinsichtlich der unterschiedlichen Navigationsstrukturen innerhalb der Web-Anwendung wird daher zwischen informationsbasierter und aufgabenbasierter Navigation differenziert. Die unterschiedlichen Vorgehensweisen bei der Analyse und der Modellbildung sollen im Folgenden aufgezeigt werden.

5.1 Konzeptualisierung des Informationsbestandes – Methoden des Knowledge Engineering

Bei der Konzeptualisierung des Informationsbestandes werden Ressourcen, die in Form von Dokumenten vorliegen, in Klassen strukturiert sowie deren Eigenschaften und vorhandenen Beziehungen zu anderen Klassen bestimmt und in einer Ontologie organisiert. Der Aufbau einer Ontologie stellt innerhalb der Vorgehensweise zum Entwurf von Web-Anwendungen einen notwendigen Schritt dar, um eine maschinenlesbare Strukturierung der Informationsinhalte, die in der Web-Anwendung zur Verfügung stehen sollen, zu erreichen.

Für die Entwicklung von Ontologien in einem Unternehmen existieren derzeit zwei Vorgehensweisen, die in der Praxis unterschiedlich häufig angewandt werden. Eine praxisnahe, jedoch zeitintensive Möglichkeit besteht darin, Begriffssysteme manuell zu erstellen. In einem Top-Down-Ansatz wird mit Beteiligung mehrerer Mitarbeiter aus verschiedenen Bereichen bzw. Disziplinen eine Ontologie erstellt, die die wesentlichen Wissenselemente und zugehörigen Dokumenttypen enthält. Die auf diese Art erstellte Themenstruktur kann jedoch nicht automatisch aktualisiert werden, so dass der Pflegeaufwand sehr hoch sein kann, falls sich die Themenstruktur ändert. Zudem muss in diesem Fall die Zuordnung der Informationsobjekte ebenfalls manuell angepasst werden.

Eine andere Möglichkeit bietet die automatische Klassifizierung der vorliegenden Dokumente zum Aufbau der Themenstruktur im Sinne einer Bottom-Up-Vorgehensweise (vgl. Lam et al., 1999; Yang, 1999). Diese basiert auf dem Grundproblem der Textkategorisierung, d. h. der Frage, wie sich Dokumente einer Menge vordefinierter Kategorien zuordnen lassen (vgl. Sebastiani, 2002) und stellt ein Kernproblem des Information Retrievals (Baeza-Yates, 1999) dar. Ein wichtiger Ansatz zur Lösung dieser Aufgabe setzt auf dem Paradigma des maschinellen Lernens (Mitchell, 1996; Morik et al., 2000) auf, bei dem versucht wird, mit Hilfe von Beispielen allgemeinere Zusammenhänge zu generieren. Die Klassifikation erfolgt dabei anhand der Eigenschaften der einzelnen Kategorien. Innerhalb eines selbstlernenden Verfahrens werden die Eigenschaften auf der Grundlage bereits klassifizierter Dokumente ermittelt und auf nachfolgende Dokumente angewandt. Voraussetzung für eine effektive automatische Klassifizierung ist jedoch eine homogene Menge der zu analysierenden Dokumente. In der Praxis hat sich gezeigt, dass eine manuelle Nachbearbeitung trotz verbesserter Analysewerkzeuge nach wie vor notwendig ist (Spies, 2002). Allerdings können neu erstellte Informationsobjekte systemunterstützt und somit in vielen Fällen einfacher in die Ontologie eingefügt werden.

Bei der Analyse der Organisationsstrukturen und Abläufe in einem Unternehmen lassen sich oftmals Ähnlichkeiten zu anderen Unternehmen feststellen. Anstatt

Teilbereiche, die wiederkehrende Muster aufweisen, erneut zu modellieren, können existierende Standardmodelle genutzt und ggf. angepasst werden (Hays, 1996; Fowler, 1998; Parsons & Wand, 1997). Unter diesem Aspekt bietet die ontologiebasierte Vorgehensweise die Möglichkeit, bereits existierende Ontologien bestimmter Domänen zumindest teilweise nutzen zu können bzw. auszutauschen.

Auch wenn für den Entwurf von Ontologien aufgrund der Tatsache, dass es sich hierbei um eine noch relativ junge Disziplin handelt, noch keine einheitliche Vorgehensweise bei der Modellierung existiert, wurden seit Mitte der neunziger Jahre verschiedene Ansätze zur Ontologieentwicklung vorgeschlagen, die einen strukturierten Erstellungsprozess zugrunde legen. Neben Uschold und King (Uschold & King, 1995), die als erste den Ontologieaufbau in Form eines Entwicklungsprozesses darstellten, sind hier vor allem die TOVE-Methodologie (Grüninger & Fox, 1998) und die Vorgehensweise METHODOLOGY (Fernandez-Lopez et al., 1999) zu nennen. Da Ontologien modelliert werden, müssen Designentscheidungen getroffen werden. Beispielsweise stellt sich die Frage, ob zwischen Objekten, Eigenschaften und Relationen unterschieden werden soll, oder sich das eine nicht durch das andere ausdrücken lässt. So können durchaus Eigenschaften durch Relationen beschrieben und Relationen durch Objekte ersetzt werden. Zusätzlich besteht auch die Möglichkeit, Vererbung auf Objekt- statt auf Klassenebene zu definieren (Sciore, 1989).

Weiterhin werden objektive Kriterien benötigt, die bei der Erstellung von Ontologien Hilfestellung geben. In diesem Zusammenhang wurden von Gruber frühzeitig einige Kriterien für deren Entwurf aufgestellt. Hierzu zählen Klarheit, Kohärenz, Erweiterbarkeit, Kodier-Unabhängigkeit und minimale Verpflichtungen (Gruber, 1993). Außerdem sollte beim Erstellen einer Ontologie im Sinne einer späteren Wiederverwendung eine zu starke Spezifizierung vermieden werden. In der Praxis zeigt sich, dass die Erstellung begrenzter Ontologien von Dokumenten bestimmter Fertigungs- und Dienstleistungsprozesse einer „Unternehmensontologie“ vorzuziehen ist. Dies gilt vor allem für Unternehmen, die in stark dynamischen Märkten tätig sind, da hier die Strukturierung der Themen und Metainformationen schnell veraltet sein kann.

Bevor Beschreibungsmöglichkeiten von Ontologien betrachtet werden, soll an dieser Stelle eine formale Definition einer Ontologie vorgestellt werden, die auf der formalen Beschreibung einer ontologischen Struktur von Motik, Maedche und Volz (Motik et al., 2002) basiert und sich für eine Vielzahl von Ontologiebeschreibungssprachen nutzen lässt.

Definition 5.1 *Eine* Ontologie *ist ein 8-Tupel* $\boldsymbol{O} = \left(\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{L}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}^{\boldsymbol{c}}, rel, \boldsymbol{A}^{\boldsymbol{o}}\right)$

bestehend aus

- *einer Menge* $\boldsymbol{C}$ *von Konzepten und einer Menge* $\boldsymbol{R}$ *von Relationen,*
- *einem Lexikon* $\boldsymbol{L}$*:* $\boldsymbol{L}$ *enthält eine Menge* $\boldsymbol{L^C}$ *von Symbolen für Konzepte und eine Menge* $\boldsymbol{L^R}$ *von Symbolen für Relationen, so dass* $\boldsymbol{L} = \boldsymbol{L^C} \cup \boldsymbol{L^R}$ *,*
- *zwei Relationen* $\boldsymbol{F} \subseteq \boldsymbol{L^C} \times \boldsymbol{C}$ *und* $\boldsymbol{G} \subseteq \boldsymbol{L^R} \times \boldsymbol{R}$ *die Referenzen für Konzepte und Relationen darstellen. Für* $L \in \boldsymbol{L^C}$ *seien* $\boldsymbol{F}(L) = \{C \in \boldsymbol{C} \mid (L,C) \in \boldsymbol{F}\}$ *und* $\boldsymbol{F}^{-1}(C) = \{L \in \boldsymbol{L^C} \mid (L,C) \in \boldsymbol{F}\}$ *sowie für* $L \in \boldsymbol{L^R}$ *seien* $\boldsymbol{G}(L) = \{R \in \boldsymbol{R} \mid (L,R) \in \boldsymbol{G}\}$ *und* $\boldsymbol{G}^{-1}(R) = \{L \in \boldsymbol{L^R} \mid (L,R) \in \boldsymbol{G}\}$*,*
- *einer Konzepthierarchie* $\boldsymbol{H^C}$*:* $\boldsymbol{H^C}$ *ist eine gerichtete Relation* $\boldsymbol{H^C} \subseteq \boldsymbol{C} \times \boldsymbol{C}$*. Für* $\boldsymbol{H^C}(\boldsymbol{C}_1, \boldsymbol{C}_2)$ *gilt,* $\boldsymbol{C}_1$ *ist Subkonzept von* $\boldsymbol{C}_2$*,*
- *einer Funktion* $rel : \boldsymbol{R} \rightarrow \boldsymbol{C} \times \boldsymbol{C}$*, die nicht-taxonomische Relationen zwischen Konzepten beschreibt. Die Funktion* $dom : \boldsymbol{R} \rightarrow \boldsymbol{C}$ *mit* $dom(R) = \Pi_1(rel(R))$ *bestimmt die Domäne von* R*,* $range(R) = \Pi_2(rel(R))$ *seinen Wertebereich (für* $rel(R) = (C_1, C_2)$ *kann man auch* $R(C_1, C_2)$ *schreiben),*
- *einer Menge* $\boldsymbol{A^O}$ *von Axiomen (Ausdrücke in einer wählbaren Logiksprache).*

Die Struktur einer Ontologie soll am folgenden Beispiel verdeutlicht werden. Seien $\boldsymbol{C} = \{c_1, c_2, c_3\}$ eine Menge von Konzepten einer Ontologie $\boldsymbol{O}$ und $\boldsymbol{R} = \{r_1\}$ eine Menge von Relationen in $\boldsymbol{O}$. Weiter seien $\boldsymbol{H^C}(c_1, c_2)$ eine hierarchische Relation und $r_1(c_2, c_3)$ eine nicht-taxonomische Relation. Das Lexikon $\boldsymbol{L} = \boldsymbol{L^C} \cup \boldsymbol{L^R}$ ist definiert als $\boldsymbol{L^C}$ = {'Unternehmen', 'Organisation', 'Produkt'} und $\boldsymbol{L^R}$ = {'stellt_her'}. Die lexikalischen Einträge lassen sich durch die Funktionen $\boldsymbol{F}$ und $\boldsymbol{G}$ wie folgt referenzieren: $\boldsymbol{F}$ ('Unternehmen') = c_1, $\boldsymbol{F}$ ('Organisation') = c_2, $\boldsymbol{F}$ ('Produkt') = c_3 und $\boldsymbol{G}_1$ ('stellt_her') = r_1.

Auf der Struktur einer Ontologie aufbauend, lässt sich eine Wissensbasis, die zusätzlich zur ontologischen Struktur konkrete Inhalte in Form von Instanzen enthält, wie folgt definieren:

Definition 5.2 *Eine* Wissensbasis $\boldsymbol{KB}$ *ist ein Quadrupel* $\boldsymbol{KB} = (\boldsymbol{O}, \boldsymbol{I}, inst, instr)$ *bestehend aus*

- *einer Ontologie* $\boldsymbol{O}$ *mit* $\boldsymbol{O} = (\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{L}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H^C}, rel, \boldsymbol{A^O})$*,*
- *einer Menge* $\boldsymbol{I}$ *von Instanzen,*

- *einer Konzeptinstanzierungsfunktion inst mit* $inst : \boldsymbol{C} \rightarrow 2^{\boldsymbol{I}}$,
- *einer Relationsinstanzierungsfunktion instr mit* $instr : \boldsymbol{R} \rightarrow 2^{\boldsymbol{I} \times \boldsymbol{I}}$.

Zur Beschreibung von Ontologien steht eine Vielzahl von Repräsentationssprachen zur Verfügung. Im Prinzip lässt sich jede Beschreibungssprache nutzen, die zur konzeptuellen Modellierung geeignet ist. Hierzu zählen beispielsweise UML, PL1 (Prädikatenlogik erster Stufe), ER-Modelle sowie logische Sprachen wie LOOM (MacGregor & Bates, 1987) und Frame Logic (kurz: F-Logic) (Kifer & Lausen, 1989). Unterschiede zwischen den Sprachen existieren vor allem hinsichtlich ihrer Ausdrucksstärke. Im Zusammenhang mit der Idee des *Semantic Web* (Berners-Lee et al., 2001) haben sich vor allem XML und XML Schema[5] als Grundlage für weitere Sprachen etabliert. Ziel des Semantic Web ist es, Inhalte des WWW derart zu beschreiben, dass sie auch für Computer verständlich sind und verarbeitet werden können. Dazu werden den Inhalten Metainformationen zugefügt, so dass diese eine Semantik erhalten. Ontologien lassen sich, aufbauend auf XML, neben SHOE (Heflin et al., 1999), XOL (Karp et al., 1999) und OML/CKML (Kent, 1999) vor allem durch RDFS (Brickley & Guha, 2000) und durch die RDF-basierte Sprache DAML+OIL (Hendler & McGuinnes, 2001) beschreiben.

5.2 Prozessanalyse – Business Integration

Zusätzlich zu den in Unternehmen vorliegenden Ressourcen in Form von Dokumentenbeständen lassen sich auch Aufgaben und Prozesse semantisch beschreiben. Durch eine Analyse der im Unternehmen vorherrschenden Wissensflüsse können Arbeitsabläufe und zugehörige Themenfelder identifiziert werden. Neben dem reinen informationsbasierten Untersuchungsansatz ist durch eine prozessorientierte Analyse eine Übertragbarkeit der Abläufe im Unternehmen und zugehörigen Aktionen auf die Darstellung in einer Web-Anwendung möglich. Die semantische Beschreibung der Arbeitsabläufe kann auf dieser Grundlage den Ausgangspunkt für die Modellierung und Bereitstellung einer aufgabenbasierten Navigation bilden. Hierbei werden dem Benutzer der Website Präsentation und Navigation abhängig von der Aufgabenstellung, d. h. vom aktuellen Stand der Aufgabenbearbeitung, angeboten. Zusätzlich zu einer formalen Beschreibung der Arbeitsabläufe ist eine enge Anbindung der Web-Präsentation und Navigation an prozessunterstützende Systeme erforderlich, die den Zustand des zugehörigen Workflow erfassen und darlegen können.

Für die Bereitstellung einer aufgabenbasierten Navigation innerhalb einer Web-Applikation existieren zwei mögliche Vorgehensweisen. Zum einen besteht die

[5] http://www.w3.org/XML/Schema

Möglichkeit, Darstellung, Navigation und Interaktion zu einzelnen Prozessschritten zu modellieren und ggf. die Übergänge von einem Prozessschritt zum nächsten automatisiert durchzuführen. Zur automatischen Unterstützung ist jedoch die Anbindung eines Workflow-Management-Systems erforderlich.

Zum anderen ist auch eine weitergehende automatisierte Systemunterstützung möglich. Hierzu übernehmen spezielle navigationale Komponenten, abhängig von der Aufgabenstellung und dem aktuellen Prozesszustand, die Darstellung der entsprechenden Informationen sowie die Bereitstellung zugehöriger Navigations- und Interaktionskomponenten. Die Konzeption dieser navigationalen Komponenten, die mit weitgehenden Funktionalitäten ausgestattet sind und die Modellierung einer entsprechenden webbasierten Aufgabennavigation entbehrlich machen, soll nicht Gegenstand dieser Arbeit sein. Jedoch sollen im Folgenden grundlegende Mechanismen zur Integration einer aufgabenbasierten Navigation angesprochen werden.

Für die Bereitstellung einer aufgabenbasierten Navigation ist es erforderlich, die zur Lösung der Aufgabe notwendigen Aktivitäten, also den zugrunde liegenden Prozess, zunächst zu modellieren und dann semantisch zu beschreiben. Für die Modellierung von Prozessen existiert inzwischen eine Vielzahl von Werkzeugen. Im Zusammenhang mit der Entwicklung von Web-Anwendungen soll jedoch davon ausgegangen werden, dass die Aufgaben bzw. Prozesse auf die Web-Applikation lediglich portiert werden, d. h. bereits modelliert wurden und als Prozessmodell zur Verfügung stehen. Zur semantischen Beschreibung von Prozessen, die sich vornehmlich auf Web Services (Florescu & Kossmann, 2001), also Dienste, die über das Web bereitgestellt werden, konzentrieren, wurde DAML-S (Ankolenkar et al., 2001) entwickelt. Es handelt sich dabei um eine Ontologie, deren Ziel die maschinelle und weitestgehend automatisierte Verarbeitung von Web Services ist. Mit Hilfe von DAML-S lassen sich Funktionalitäten vor allem zum Finden eines „passenden“ Dienstes, dessen Aufruf, zur Zusammensetzung mehrerer Web Services zu einer Prozesskette und zur Überwachung der Ausführung automatisiert durchführen. Die Ontologie verfügt dabei über ein *Service Profile*, ein *Service Model* und über ein *Service Grounding*. Das *Service Profile* liefert eine Beschreibung des Dienstes an sich und seine Funktion, während das Service Model Auskünfte darüber erteilt, was bei der Ausführung des Dienstes geschieht. Den Zugriff auf den Dienst durch Agenten klärt im Detail das *Service Grounding*. Die einzelnen Prozessschritte einschließlich zugehöriger Kontroll- und Datenflussstrukturen eines Web Service werden im *Process Model* repräsentiert, das eine Erweiterung des *Service Model* darstellt. Zur Modellierung der aufgaben- bzw. prozessbezogenen Navigation können vor allem diese Informationen für die Beschreibung der einzelnen Prozessschritte genutzt werden.

Einen weiteren interessanten Ansatz zur Beschreibung von Geschäftsprozessen stellt die auf XML bzw. WSDL[6] basierende Sprache BPEL (Curbera et al., 2002) dar, die weite Übereinstimmungen mit dem Process Model in DAML-S aufweist. BPEL unterscheidet zwischen abstrakten und ausführbaren Geschäftsprozessen. Während abstrakte Prozesse lediglich das Verhalten darstellen, bilden ausführbare Prozessbeschreibungen die Grundlage für die Implementierung und Bereitstellung des entsprechenden Prozesses. Darüber hinaus ermöglicht BPEL auch die Definition und Modellierung komplexer Prozesse. Diese Prozessbeschreibungen lassen sich ebenfalls für die Modellierung einer aufgabenbasierten Navigation nutzen.

5.3 Bereitstellung ontologiebasierter Ressourcen

Die Konzeptualisierung der Informationsbestände und Prozesse führt zur Modellierung einer Ressourcenontologie und einer Prozessontologie. Beide Ontologien können einerseits zur Unterstützung des Entwurfsprozesses einer Web-Anwendung herangezogen werden, andererseits dienen sie zur Laufzeit der Web-Applikation als Informationsbasis für den Zugriff auf die modellierten Inhalte.

Prinzipiell ist die Verfügbarkeit der Informationen in einer Datenbank für den Zugriff seitens der Website ausreichend. Aus verschiedenen Gründen, die z. T. bereits in Kapitel 4.7 angesprochen wurden, bietet sich die Nutzung eines Inferenzsystems an. Hierzu müssen die Informationen der beiden Ontologien in Fakten bzw. Regeln eines Inferenzsystems konvertiert werden, was nachfolgend erläutert werden soll. Zuvor soll jedoch die grundsätzliche Funktionsweise von Inferenzsystemen skizziert werden.

5.3.1 Inferenzsysteme

Die Bearbeitung der Regeln erfolgt durch regelbasierte Inferenzsysteme. Hier gilt es, zwischen Produktionssystemen und Frame-basierten Systemen zu unterscheiden. In Produktionssystemen besitzen die Regeln einen IF-Teil (Antezedens) und einen THEN-Teil (Konsequenz). Regeln können zu Clustern oder Blöcken zusammengefügt werden, die einen Ast des Entscheidungsbaumes darstellen. Ist das Antezedens erfüllt, wird die Regel ausgelöst und die in der Konsequenz beschriebene Aktion ausgeführt. Entweder wird nun eine weitere Regel bearbeitet oder der Regelblock beendet. Letzteres veranlasst das Inferenzsystem zu einer Auswahl von Schlussfolgerungen oder Empfehlungen. Produktionssysteme erlauben die Interaktion mit dem Nutzer, so dass sie sich zur Durchführung von Aufgabenselektionen eignen.

Frame-basierte Systeme hingegen fassen alle nötigen Informationen für die Behandlung einer Information oder eines Objektes in einem sog. Frame zusammen

[6] http://www.w3.org/TR/wsdl

(vgl. Minsky, 1975). Sie repräsentieren Wissen in Form hierarchisch organisierter vernetzter Knoten, die Eigenschaften ihrer übergeordneten Knoten erben. Ein Knoten repräsentiert eine Klasse oder ein Objekt mit seinen Attributen bzw. Werten, die in Slots gespeichert werden. Werte können durch das Auslösen von Regeln ermittelt werden. Frame-basierte Systeme eignen sich daher insbesondere zur Bestimmung, welche Informationen für einen Nutzer relevant sind. Prinzipiell lassen sich die Vorteile beider Systeme auch kombinieren (Tanzer & Shasha, 1999).

5.3.2 Aufbau von Fakten und Regeln aus Ontologien

Inzwischen existieren verschiedene Werkzeuge zur Generierung von Fakten, ausgehend von einer Ontologie. Da die Expertensystemen zugrunde liegenden Sprachen prinzipiell zur konzeptuellen Modellierung ebenso geeignet sind wie Ontologien, stellt die Konvertierung von Informationen keinen aufwändigen Schritt dar. Dies gilt zunächst für die Erzeugung von Fakten anhand einer Ressourcenontologie. Einen Ansatz zur Abbildung einer DAML-Ontologie in Fakten und Regeln des Produktionssystems Jess (Friedman-Hill, 1999) stellt DAMLJessKB[7] dar. Die Transformation von Konzepten und Instanzen in F-Logic ermöglicht das Tool FloraTab[8]. Das Ergebnis beider Ansätze ist eine Menge von Fakten, die anhand von Queries abgerufen werden. Zusätzlich zu diesen Fakten können Regeln erzeugt werden, die die Bereitstellung von Informationen beispielsweise anhand von Ereignissen automatisiert durchführen.

Für die Transformation von Aufgaben und Prozessen einer Prozessontologie in ein Expertensystem müssen für die einzelnen Prozessschritte Regeln erzeugt werden, die auf dem Faktenwissen der Ressourcenontologie aufbauen. Derzeit existieren noch keine Konzepte oder Werkzeuge, die diese Umwandlung unterstützen, obwohl sich Vorteile sowohl für die Modellierung von Prozessen als auch für die Prozessabläufe zur Laufzeit der Anwendung ergeben.

5.4 Anwendungen des Inferenzsystems

Für den Entwurf und die Anwendung von Web-Applikationen erweist sich der Einsatz eines Inferenzsystems in zwei Bereichen als sinnvoll. Zum einen können die beim Entwurf einer Website erzeugten Modelle verifiziert werden. Zum anderen ist es möglich, zur Laufzeit anhand verschiedener Situationen und Ereignisse, entsprechende Informationen zu generieren, die nicht explizit in der Datenbasis enthalten sind. Durch eine Art automatischen Schließens entscheidet das System, welche Informationen unter Berücksichtigung verschiedener Faktoren wie Ort, Zeit, Nutzerinteresse etc. relevant sind.

---

[7] http://edge.mcs.drexel.edu/assemblies/software/damljesskb/damljesskb.html
[8] http://www.dfki.uni-kl.de/~sintek/FloraTab

Die Verifikation der Modelle bezieht sich in erster Linie auf die Plausibilitätsprüfung des konzeptuellen Modells. Verschiedene Editoren für den Entwurf von Ontologien verfügen bereits über eine Anbindung an ein Inferenzsystem, das neben der Überprüfung der Ontologie auch die Zuordnung von Konzepten und Beziehungen auf Basis gemeinsamer Eigenschaften automatisch unterstützt. Eine weitere Möglichkeit besteht darin, die zur Navigationsmodellierung eingesetzten Modelle zur verifizieren (s. Kapitel 10.2).

Ebenfalls können die zur Bearbeitung einer Aufgabenstellung erforderlichen Prozessschritte systemunterstützt auf Plausibilität hin überprüft oder ggf. sogar optimiert werden. Im Falle eines konsultativen Inferenzsystems lassen sich zudem automatisiert Vorschläge bzgl. der als nächstes auszuführenden Aufgabe in einer bestimmten Situation generieren. Ergänzend ließe sich das Faktenwissen sowie das Wissen über einzelne Prozessschritte nutzen, um bei der Modellierung der Zielsetzung einer Aufgabenstellung, ausgehend von einer Startkonfiguration, automatisiert eine mögliche Schrittfolge zur Lösung der Aufgabe vorzuschlagen.

5.5 Zusammenfassung

Die Konzeptualisierung stellt anhand der Relationen zwischen den enthaltenen Klassen bzw. Konzepten eine erste Navigationsmöglichkeit über Themen zur Verfügung. Für eine definierte Navigation in der späteren Web-Anwendung ist dies jedoch nicht ausreichend, da es an dieser Stelle noch nicht möglich ist, gezielt auf Informationskomponenten zuzugreifen, d. h. bestimmte Merkmale einer definierten Instanz zu adressieren. Allerdings lassen sich mit Hilfe einer Ontologie explorative Zugriffe innerhalb der Web-Applikation realisieren und darüber hinaus Navigationsmöglichkeiten auf dynamische Mengen und Hierarchien erweitern.

6 Komponentenentwurf und Navigationsmodellierung

Webseiten setzen sich aus einer Vielzahl unterschiedlicher Informationen zusammen, die typischerweise aus verschiedenen Quellen stammen. Die Inhalte werden gewöhnlich nicht nur auf die einzelnen Seiten einer Website verteilt, sondern finden sich, nicht notwendigerweise semantisch miteinander in Verbindung stehend, in verschiedenen Darstellungsbereichen einer Seite wieder. Hierdurch lassen sich auf einer Webseite verschiedene Nutzerinteressen ansprechen bzw. mehrere, auch verschiedenartige Inhalte übersichtsartig präsentieren, die dann auf weiteren Webseiten detaillierter dargestellt werden können.

Dies hat zur Folge, dass die Darstellung von Websites, wie auch bereits der Aufbau einzelner Webseiten, komplexe Strukturen beherbergen kann. Die Komplexität wird dadurch verstärkt, dass sich die einzelnen Teilbereiche einer Webseite unabhängig voneinander aktualisieren oder mit neuen Informationen füllen lassen. Bei der Navigation kann sich daraus ein unerwünschtes Darstellungsverhalten ergeben, wie beispielsweise die wiederholte Einbettung einer Gesamtseite in einen in ihr enthaltenen Teilbereich (Rekursion) oder eine inhaltliche Darstellung im Hauptansichtsbereich einer Webseite, die aufgrund der Weiterverfolgung von Links mit der umgebenden Menüstruktur in keinem Zusammenhang mehr steht.

Grundlegend für die Entwicklung übersichtlicher und verständlicher Websites ist daher die abstrahierte und formale Betrachtung der Navigations- und Darstellungsstrukturen mit Hilfe eines Entwicklungsmodells. In diesem Kapitel sollen zunächst Komponenten für die Darstellung der verschiedenen Inhalte einer Website, wie z. B. Texte, Bilder, dynamisch generierte Informationen, externe Applikationen, Multimedia-Streams etc. formal beschrieben werden. Ziel der formalen Definition ist es, einerseits die Erstellung der Entwurfsmodelle zu systematisieren und andererseits den softwarebasierten Automatisierungsprozess zur Generierung der Website anhand der erstellten Modelle zu ermöglichen.

Die Zusammensetzung von Komponenten zu Teilbereichen einer Webseite bzw. einer Gesamtseite erfolgt mit Hilfe von Kompositionsmodellen. Diese legen die Struktur einzelner Webseiten fest, d. h. die Aufteilung einer Webseite in unterschiedliche Bereiche, die unabhängig voneinander modifiziert und aktualisiert werden können. Darauf aufbauend definieren Übergänge zwischen einzelnen Komponenten bzw. Teilbereichen einer Webseite die eigentliche Navigationsstruktur. Das Ergebnis ist ein Navigationsmodell, das die Zusammensetzung von Komponenten zu konjunktiven oder disjunktiven Sichten beschreibt und Übergänge zwischen den Sichten in Form von Navigationsrelationen definiert. Die grafische Darstellung des

Navigationsmodells orientiert sich dabei weitestgehend am späteren Erscheinungsbild der Website.

6.1 Formale Definition und grafische Beschreibung der Komponenten

Bei der Gestaltung von Webseiten werden zwei unterschiedliche Arten von Komponenten eingesetzt. Strukturelle Komponenten bestimmen den strukturellen Aufbau einer Seite, d. h. sie definieren die Zusammenstellung von Komponenten zu Teilbereichen und deren konzeptionelle Anordnung innerhalb der Seite. Inhaltsspezifische Komponenten sind für die Präsentation der Inhalte verantwortlich. Diese Komponenten legen die Darstellung statischer Inhalte, d. h. Inhalte, die bereits zur Modellierungszeit bekannt sind, sowie dynamischer Inhalte, die sich erst zur Laufzeit der Web-Anwendung ergeben, fest.

6.1.1 Container und Partitionen

Die Elemente, die eine einzelne Webseite beschreiben, werden in sog. *Containern* organisiert. Diese haben die Aufgabe, sowohl die gesamte Website als auch einzelne Webseiten zu strukturieren und die darzustellenden Inhalte getrennt voneinander zu betrachten. Daraus ergibt sich die Anforderung, dass sich in einen Container weitere Container einbetten lassen, so dass verschachtelte Webseiten darstellbar sind. Jeder Container kann beliebig viele Subcontainer beinhalten, die wiederum als eigenständige Container betrachtet werden.

Durch die Möglichkeit, Container ineinander einzubetten, entstehen Ebenen. Mehrere Subcontainer innerhalb eines Containers, die auf der gleichen Ebene liegen, werden gemäß einer XOR-Verknüpfung dargestellt, d. h. in diesem Fall ist immer genau ein Subcontainer sichtbar (vgl. Abbildung 6-1 (a)). Liegen mehrere Subcontainer auf gleicher Ebene eines Containers, so muss festgelegt werden, welcher Subcontainer zunächst sichtbar ist. Dazu wird gemäß Abbildung 6-2 (a) ein Startzustand definiert. Um verschiedene Container parallel darstellen zu können, werden Subcontainer als einzelne *Partitionen* angeordnet. Alle Partitionen in einem Container werden so in der Web-Anwendung gleichzeitig angezeigt (vgl. Abbildung 6-1 (b)).

Die Inhalte, die in einem Container beschrieben werden können und später in der Web-Darstellung präsentiert werden, lassen sich in zwei Kategorien einteilen. Zum einen stehen sie zur Modellierungszeit bereits fest, es handelt sich in diesem Fall um statische Inhalte. Zum anderen ist es möglich, dass sich die auf einer Webseite anzuzeigenden Informationen erst zur Laufzeit einer Web-Anwendung ergeben, also dynamisch erzeugt werden. Sowohl statische als auch dynamische Inhalte werden nicht unmittelbar in einem Container gehalten, sondern durch sog. Modell-

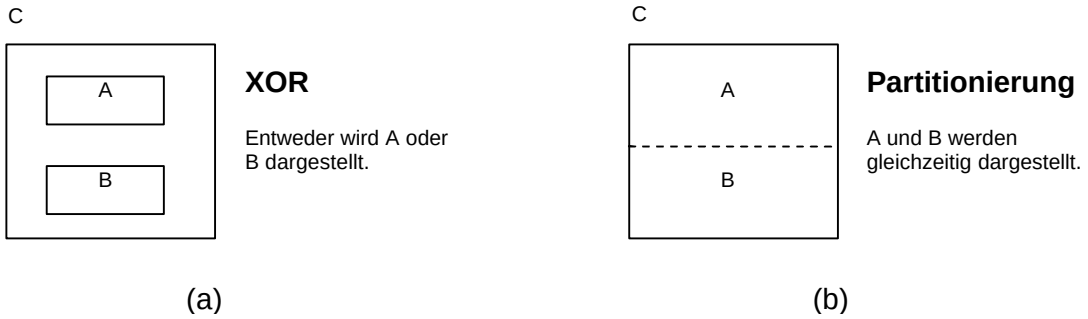


Abbildung 6-1: Container und Partitionen

klassen beschrieben, die einem Container zugeordnet sind. Prinzipiell kann jeder Container neben Subcontainern beliebig viele Modellklassen enthalten.

Aufgabe der Modellklassen ist es, die Klassifizierung der Inhalte innerhalb der Ontologie auch im Navigationsmodell zu erhalten. Hinsichtlich statischer Inhalte wird daher ihre Art im Navigationsmodell festgelegt, d. h. es wird definiert, ob es sich beispielsweise um Texte, Bilder, Services oder andere Inhalte handelt. Werden Inhalte dynamisch generiert, geht ihre Art aus der Ressourcenontologie hervor, d. h. sie können über den in der Ontologie definierten Wertebereich der entsprechenden Eigenschaft eines Konzeptes ermittelt werden. Zusätzlich übernehmen hinsichtlich dynamischer Inhalte die Modellklassen die Aufgabe, die darzustellenden Informationen zu gruppieren, falls mehrere Instanzen eines Konzeptes oder weitere, in Beziehung stehende Konzepte der Ontologie dargestellt werden müssen.

Dynamische Inhalte sind im Unterschied zu den statischen Inhalten kontextabhängig, d. h. die darzustellenden Informationen ergeben sich erst zur Laufzeit aus dem aktuellen Zustand der Web-Anwendung. Anhand des Navigationsverhaltens des Benutzers, beispielsweise durch Aufzeichnung des Navigationspfades oder der durchgeführten Interaktion, lässt sich der gültige Kontext ermitteln. Hierdurch sind Rückschlüsse über den Zusammenhang möglich, in dem die Informationen präsentiert werden sollen.

Vor diesem Hintergrund ist es erforderlich, den aktuellen Kontext einem Container während der Navigation mitzuteilen. Da alle Inhalte in Modellklassen und daher nicht unmittelbar im Container abgelegt werden, erfolgt die Navigation stets von einer Modellklasse zu einem Container oder einer anderen Modellklasse. Der aktuelle Kontext eines Containers setzt sich somit aus der Ursprungsmodellklasse, der aktuellen Instanz und ihrem zugehörigen Konzept in der Ontologie zusammen.

Definition 6.1 *Sei* $KB = (\boldsymbol{O}, \boldsymbol{I}, inst, instr)$ *mit* $\boldsymbol{O} = (\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{L}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}^{\boldsymbol{C}}, rel, \boldsymbol{A}^{\boldsymbol{O}})$

eine Wissensbasis und $\boldsymbol{M}$ *die Menge der Modellklassen. Ein* Container *ist ein Tupel* $B=(m,c,i)$*, bestehend aus einer Ursprungsmodellklasse* $m\in \boldsymbol{M}$*, einem Ursprungskonzept* $c\in \boldsymbol{C}$ *und einer Ursprungsinstanz* $i\in \boldsymbol{I}$*.*

Die grafische Beschreibung des Navigationsmodells, das die Grundlage für die spätere Darstellung der Web-Applikation bildet, beruht auf einem Higraphen-ähnlichen Ansatz (vgl. Harel, 1988). Es definiert die Anordnung von Containern zu beliebig komplexen Sichten, die durch flächenmäßige Erschließung von Teilsichten erzeugt werden können. Einzelne Bereiche werden durch gerichtete Navigationsbeziehungen, die eine Übergangsrelation zwischen den einzelnen Containern darstellen, miteinander verbunden. Für den Aufbau dieser Sichten anhand von Containern und Subcontainern ergibt sich entsprechend der formalen Beschreibung von Higraphen folgende Definition:

Definition 6.2 *Sei* $\boldsymbol{B}$ *eine endliche Menge von Containern. Die* Subcontainer-Funktion k *ist definiert als*

$$k:\boldsymbol{B}\rightarrow 2^{\boldsymbol{B}}.$$

Für $x\in \boldsymbol{B}$ *seien weiter*

$$k^{0}(x)=\{x\},\ k^{i+1}(x)=\bigcup_{y\in k^{i}(x)}k(y)\ \textit{und}\ k^{+}(x)=\bigcup_{i=1}^{\infty}k^{i}(x)\ \textit{mit}\ x\notin k^{+}(x),$$

d. h. k *erzeugt keinen Zyklus. Die* Partitionierungsfunktion p *ist definiert als*

$$p:\boldsymbol{B}\rightarrow 2^{\boldsymbol{B}\times\boldsymbol{B}}$$

und stellt eine Äquivalenzfunktion dar, die die Menge $k(x)$ *in disjunkte Äquivalenzklassen, d. h. in sog.* Partitionen $p_1(x),...,p_k(x)\in \boldsymbol{B}$ *zerlegt.*

Die Subcontainer-Funktion weist jedem Container eine Menge weiterer, unmittelbar enthaltener Container zu. Jeder Container definiert einen Zielbereich, in dem die Inhalte der Web-Anwendung mittels einer Modellklasse angezeigt werden können. Startcontainer und Partitionen werden zur Laufzeit mit den Werten ihres übergeordneten Containers initialisiert.

6.1.2 Platzhalter

Mit Hilfe von Platzhaltern können Teilbereiche der Web-Anwendung zur besseren Überschaubarkeit schematisch dargestellt werden (vgl. Abbildung 6-2 (b)). Eine

detaillierte Modellierung des durch einen Platzhalter nur stellvertretend dargestellten Bereiches kann an einer anderen Stelle erfolgen.

Abbildung 6-2: (a) Festlegung des Startzustandes; (b) Platzhalter für einen an anderer Stelle definierten Container

6.1.3 Primitive statische Klassen

Primitive statische Klassen beschreiben einfache Objekte, die nicht weiter zerlegt werden können. Hierzu zählen beispielsweise Texte, Bilder, Multimediaobjekte, externe Links, einfache HTML-Seiten etc. Derartige Objekte werden als Instanzen ihres zugehörigen primitiven statischen Konzeptes innerhalb der zugrunde liegenden Ontologie betrachtet und dienen in erster Linie zur einheitlichen Beschreibung der verschiedenen darzustellenden Objekte innerhalb des Navigationsmodells. Die Menge der primitiven statischen Typen wird durch T^P, die der primitiven statischen Klassen durch $\boldsymbol{P}_S \subseteq \boldsymbol{M}$ definiert.

Definition 6.3 *Sei* T^P *die Menge primitiver Typen. Eine* primitive statische Klasse *ist ein Tupel* $P_S = (t, i)$ *mit* $t \in T^P$, $i \in I$.

6.1.4 Primitive dynamische Klassen

Nicht alle darzustellenden Inhalte müssen bereits zur Modellierungszeit feststehen. Vielmehr sollen Veränderungen im Datenbestand nicht notwendigerweise zu einer Neuentwicklung der Web-Applikation oder von Teilen der Anwendung führen. Darüber hinaus ist die Präsentation von Inhalten in manchen Fällen abhängig vom Navigationsverhalten des Benutzers bzw. von bereits erfolgten Benutzereingaben.

Aus diesem Grund müssen Modellkomponenten zur Verfügung gestellt werden, die es dem Entwickler einer Web-Anwendung erlauben, Inhalte zur Laufzeit dynamisch generieren zu lassen, die darüber hinaus vom aktuellen Kontext der Applikation abhängig sein können. Primitive dynamische Klassen sind derartige Modellkomponenten, die ein Attribut einer erst zur Laufzeit definierten Instanz eines Konzeptes der Ontologie darstellen können. Hierzu sind die Angabe eines Zielkonzeptes der

Ontologie, die entsprechende Beziehung zum Ausgangskonzept und die Angabe des Attributs, d. h. der darzustellenden Eigenschaft der Instanz notwendig.

Die Auswahl der zur Laufzeit darzustellenden Instanz ist von dem gegenwärtigen Kontext abhängig oder genauer, vom Ursprungselement, bei dem es sich entweder um die aktuelle Instanz oder um das derzeit gültige Konzept handelt. Das Ursprungselement kann entweder bereits zur Modellierungszeit festgelegt oder mittels Rückverfolgung zur Laufzeit berechnet werden. Mit diesem sowie dem Zielkonzept und der zugehörigen Relation ergibt sich die darzustellende Instanz, was zu folgender Definition führt:

Definition 6.4 *Sei* T^P *die Menge primitiver Typen. Eine* primitive dynamische Klasse *ist ein Quadrupel* $P_D = (t, j, c, r)$ *mit einem Typ* $t \in T^P$, *einem Ursprungselement* $j \in I \cup \boldsymbol{C}$, *einem Zielkonzept* $c \in \boldsymbol{C}$ *und einer Relation* $r \in \boldsymbol{R}$.

Die Menge der primitiven dynamischen Klassen wird mit $\boldsymbol{P}_D \subseteq \boldsymbol{M}$ bezeichnet.

Folgendes Beispiel soll die Funktionsweise verdeutlichen: Ein Online-Händler stellt auf seiner Website Produktkataloge verschiedener Hersteller zur Verfügung. Um zu einem bestimmten Produkt zu gelangen, kann der Benutzer zunächst unter den verschiedenen Herstellern wählen und dann die gewünschte Produktkategorie angeben. Es werden nun in einem Bereich der Webseite alle Produkte der entsprechenden Kategorie aufgelistet. Zusätzlich soll jedoch das Firmenlogo des zuvor gewählten Herstellers neben den aufgeführten Produkten angezeigt werden. Diese Information ist abhängig von der Wahl des Herstellers, die jedoch außerhalb der aktuellen Webseite erfolgte und somit nicht mehr im Sichtbarkeitsbereich des gegenwärtigen Containers liegt. Zur Laufzeit sind der dynamischen Klasse zunächst nur die Konzepte ‘Hersteller’ und ‘Firmenlogo’ einschließlich der Eigenschaft ‘besitzt’ bekannt. Die Ursprungsinstanz, also die Information über den aktuell gültigen Hersteller, wird aus dem Kontext des übergeordneten Containers gewonnen. Mit diesen Informationen kann nun die Instanz des Konzeptes ‘Firmenlogo’ mit dem entsprechenden Inhalt ermittelt werden.

Die Bestimmung des benötigten Kontextes, also der gesuchten Ursprungsinstanz, zur Laufzeit gestaltet sich vergleichsweise einfach. Da jedem Container bekannt ist, von welcher Modellklasse er initialisiert wurde und ihm zusätzlich die zu dieser Zeit gültige Ursprungsinstanz zugewiesen wurde, kann durch Rückverfolgung der Navigationsschritte die gewünschte Instanz ermittelt werden. Ob eine solche Instanz tatsächlich auffindbar ist, kann durch Analyse der möglichen Navigationspfade zur dynamischen Klasse und den assoziierten Konzepten bereits zur Modellierungszeit verifiziert werden.

6.1.5 Navigationale dynamische Klassen

In einem Zielbereich kann entweder eine einzelne Instanz eines Konzeptes dargestellt werden oder, falls zu einem Konzept mehrere Instanzen vorhanden sind, eine Übersicht der vorhandenen Instanzen, mit der Möglichkeit, diese an definierter Stelle im Einzelnen zu betrachten.

Zu diesem Zweck werden *navigationale Klassen* eingeführt, die die Navigation auf einer Menge von Instanzen mit Hilfe einer Übersicht ermöglichen. Die Art und Weise, in der die Übersicht dargestellt wird, ist im zugrunde liegenden Navigationsmodell bereits festgelegt, kann jedoch auch von der navigationalen Klasse selbst bestimmt werden. Im ersten Fall wird eine Darstellungsart aus einer Menge T^N vordefinierter Navigationstypen gewählt. Zu dieser Menge gehören Darstellungstypen wie beispielsweise Baumstruktur, Listenstruktur, Sequenz etc. Wird die Darstellungsart von der navigationalen Klasse dynamisch ermittelt, werden hierzu zur Laufzeit die Kardinalität der darzustellenden Instanzmenge und der zur Verfügung stehende Bereich innerhalb der Webseite herangezogen.

Die Menge der zu repräsentierenden Instanzen kann mit Hilfe der Instanz des Ausgangskonzeptes, die zur Laufzeit ermittelt wird, und der in der navigationalen Klasse angegebenen Relation zwischen dem Ausgangskonzept und dem Zielkonzept beschrieben werden. Zur Modellierungszeit ist daher die Angabe der Eigenschaft des Ausgangskonzeptes und die Festlegung des Zielkonzeptes notwendig. Ist der übergebene Kontext konzeptbezogen und enthält somit keine Ursprungsinstanz, wird zur Bestimmung der Instanzmenge das dem Kontext mitgegebene Konzept betrachtet.

Für jede Instanz wird beim Aufbau der Webseite zur Laufzeit ein Link generiert, dessen Darstellung durch die Attributmenge der Zielinstanz festgelegt wird. Die Sortierung der Instanzmenge erfolgt anhand der zur Modellierungszeit festgelegten Kriterien durch die Sortierungsfunktion V .

Neben der Darstellung einer Menge von Instanzen können navigationale Klassen auch zur Organisation von Konzepten, die mit einem bestimmten anderen Konzept in Beziehung stehen, verwendet werden. In diesem Fall ist die Angabe des entsprechenden Konzeptes $c \in \boldsymbol{C}$ zur Modellierungszeit notwendig, da die Navigation auf Konzeptebene kontextunabhängig, d. h. ohne die Berücksichtigung einer konkreten Instanz erfolgen kann. Mit Hilfe von c und der Eigenschaft $p \in \boldsymbol{L^R}$ werden alle Konzepte ermittelt, die mit c in Verbindung stehen. Ein weiterer Unterschied gegenüber der Betrachtung von Instanzen liegt hinsichtlich der Modellierung darin, dass keine Attributmenge angegeben werden muss. Die Darstellung der Konzepte erfolgt ausschließlich anhand des jeweiligen Konzeptnamens.

Wird beim Einsatz einer navigationalen Klasse zur Verwaltung von Konzepten auf die Festlegung des Konzeptes c verzichtet, wird dieses zur Laufzeit anhand des

dem Container mitgegebenen Kontextes bestimmt. Hierdurch wird erreicht, dass die Bestimmung der darzustellenden navigierbaren Konzeptmenge aufgrund der Festlegung des Ausgangskonzeptes zur Laufzeit dynamisch erfolgen kann. Falls nicht sichergestellt werden kann, dass der Kontext für die navigationale Klasse sichtbar ist, z. B. falls sich diese in einer unabhängigen Komponente befindet, kann der Kontext unter Zuhilfenahme der Zustandserfassung durch Navigationsrelationen (s. Kapitel 6.2.2) bestimmt werden. Hierzu muss statt des Konzeptes der entsprechende Bezeichner in der Modellklasse angegeben werden, mit dem sich zur Laufzeit das erforderliche Konzept bestimmen lässt.

Die Unterscheidung zwischen der Navigation auf Instanzebene oder auf Konzeptebene wird durch die Angabe des Objekttyps zur Modellierungszeit festgelegt.

Mehrstufige Navigation

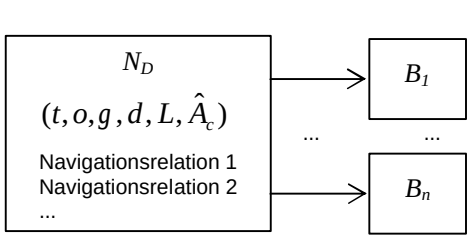
Des Weiteren besitzen navigationale Klassen die Eigenschaft, dass sie nicht nur zur Navigation auf einer einem einzigen Konzept zugeordneten Instanzmenge bzw. einer Menge von Konzepten, die mit diesem Konzept in unmittelbarer Beziehung stehen, verwendet werden können. Sie lassen sich ebenfalls nutzen, um Instanzen benachbarter Konzepte zu verwalten oder, falls auf einer Menge von Konzepten navigiert wird, alle Konzepte bis zu einer definierten Entfernung vom Ausgangskonzept innerhalb der Themenstruktur darzustellen. In beiden Fällen kann für jeden Suchschritt, d. h. für jede weitere Entfernung vom Ausgangskonzept, die Eigenschaft bestimmt werden, über die weitere Konzepte betrachtet werden sollen.

Definition 6.5 *Seien* $\boldsymbol{O} = (\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{L}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}^{\boldsymbol{C}}, rel, \boldsymbol{A}^{\boldsymbol{O}})$ *eine Ontologie,* T^N *eine Menge von Navigationstypen und* Z *eine Menge von Zustandsbezeichnern*[9]*. Eine* navigationale dynamische Klasse N_D *ist ein 9-Tupel* $N_D = (t, o, j, g, d, L, \hat{A}_c, \Xi, \hat{B})$*, bestehend aus einem Navigationstyp* $t \in T^N$*, einem Objekttyp* $o \in$ *{'Instanz', 'Konzept'}, einem Ursprungselement* $j \in I \cup \boldsymbol{C}$*, einem Element* $g \in \boldsymbol{C} \cup Z$*, einer Tiefe* d*, einer nach* d *sortierten Menge* $L \subset \boldsymbol{L}^{\boldsymbol{R}}$ *von Eigenschaften, einer Attributmenge* $\hat{A}_c$*, einer nach* d *sortierten Menge* Ξ *von Sortierungsfunktionen* $V : \hat{A}_c \to \hat{A}_c$ *und einer nach* d *sortierten Menge* $\hat{B}$ *von Zielcontainern* $B \in \boldsymbol{B}$*.*

$\boldsymbol{N}_D \subseteq \boldsymbol{M}$ bezeichnet die Menge der navigationalen dynamischen Klassen.

[9] Die Funktionsweise der Zustandserfassung und die Verwendung von Zustandsbezeichnern wird in Kapitel 6.2.2 erläutert.

Die grafische Darstellung einer navigationalen Klasse besteht aus zwei Komponenten, der navigationalen Klasse, die einem Container zugeordnet ist und einem Zielcontainer. Beide Komponenten sind in Abhängigkeit von der Tiefe des Suchfeldes durch eine oder mehrere Navigationsrelationen miteinander verbunden. Bei den Navigationsrelationen handelt es sich um parametrisierbare Links, die durch die einzelnen Instanzen der navigationalen Klasse beim entsprechenden Aufruf initialisiert werden. Abbildung 6-3 zeigt die grafische Darstellung einer navigationalen Klasse einschließlich der Parameter, die zur Modellierungszeit angegeben werden müssen. Die Navigationsrelationen sind in der grafischen Darstellung durch eine offene Pfeilspitze gekennzeichnet.



Navigationale dynamische Klasse N_D

Zustandsänderung in *B* bei Interaktion innerhalb von N_D

t = Typ des Navigators (Index, Baum, ...)
o = Objekttyp
g = Konzept bzw. Bezeichner
d = Tiefe
L = sort. Menge von Eigenschaften
$\hat{A}_c$ = Attributmenge
$\hat{B} = B_1,...,B_n$ = sort. Menge von Zielcontainern

Abbildung 6-3: Grafische Darstellung einer navigationalen dynamischen Klasse

Darstellungsformen navigationaler Klassen

Eine navigationale Klasse kann die Präsentation einer Menge von Instanzen bzw. Konzepten auf unterschiedliche Arten gestalten. Die Form der Darstellung wird im Kompositionsmodell zur Modellierungszeit festgelegt. Dabei kann zwischen verschiedenen Typen navigationaler Klassen entschieden werden, die unterschiedliche Arten für die Navigation auf einer Menge von Instanzen bzw. Konzepten zur Verfügung stellen. Grundformen navigationaler Klassen wurden bereits zum Aufbau von Navigationsstrukturen in Form von Erreichbarkeitsgrafen eingeführt (vgl. Ziegler, 1997).

Die meisten Typen navigationaler Klassen erzeugen für die Menge der darzustellenden Instanzen bzw. Konzepte eine Übersicht in Form von Navigationsrelationen, die zur Laufzeit von der Webseite als Links interpretiert werden. Die Navigationsrelationen repräsentieren einen Teil der Instanz bzw. einen Ausschnitt der zu einem Konzept zugeordneten Informationen und stellen somit lediglich eine Referenz dar. Die eigentlichen Inhalte, die über den Link erreichbar sind, werden innerhalb des Zielcontainers der navigationalen Klasse bereitgestellt. Allerdings existieren auch navigationale Klassen, die keine Übersicht mittels Referenzen erstellen, sondern

die entsprechenden Informationen unmittelbar in einem Container präsentieren, der in der Modellklasse eingebettet ist. In diesem Fall kann bei der Modellierung auf die Angabe des Zielbereiches verzichtet werden.

Tabelle 6-1 zeigt die wesentlichen Darstellungsformen, die in Web-Anwendungen vorzufinden sind:

- Index: Es wird eine Übersicht in Form einer Liste über eine einem Konzept zugewiesene Menge von Instanzen bzw. eine Menge assoziierter Konzepte erstellt. Die Auflistung kann entweder vertikal oder horizontal erfolgen.
- Group: Es wird eine Übersicht in Form einer Liste über eine einem Konzept zugewiesene Menge von Instanzen bzw. eine Menge assoziierter Konzepte erstellt. Größere Mengen von Elementen können zur besseren Übersicht in Gruppen organisiert werden. Die Auflistung kann entweder vertikal oder horizontal erfolgen.
- Tree: Es wird eine baumartige Übersicht über eine einem Konzept zugewiesene Menge von Instanzen bzw. eine Menge assoziierter Konzepte erstellt. Je nach angegebener Tiefe wird die Übersicht durch Instanzen von Unterklassen bzw. durch weitere assoziierte Konzepte ergänzt. Die Auflistung kann in Abhängigkeit der Hierarchieebene vertikal oder horizontal erfolgen.
- Panel: Es wird keine Übersicht erstellt. Die Inhalte werden parallel unmittelbar innerhalb des in der Modellklasse enthaltenen Zielcontainers dargestellt. Dabei ist die Anzahl der parallel darzustellenden Instanzen bzw. Konzepte anzugeben. Die Darstellung kann entweder vertikal oder horizontal erfolgen.
- Sequence: Es wird keine Übersicht erstellt. Die Inhalte werden sequenziell unmittelbar innerhalb des in der Modellklasse enthaltenen Zielcontainers präsentiert. Die Interaktionskomponenten zur Navigation (vor, zurück etc.) werden von der navigationalen Klasse bereitgestellt.
- Tabs: Es wird eine Übersicht über eine einem Konzept zugewiesene Menge von Instanzen bzw. eine Menge assoziierter Konzepte erstellt, die für jede Referenz einen Karteireiter generiert, mit der die zugehörige Präsentation der Inhalte im definierten Zielbereich erfolgen kann.

Die Menge der Navigationstypen ist prinzipiell beliebig erweiterbar. Es können Klassen entwickelt werden, die komplexere Darstellungs- und Navigationsformen beinhalten. Die Menge der Instanzen bzw. assoziierten Konzepte ist dabei nicht nur auf ein einziges Konzept der Ontologie beschränkt. Navigationale Klassen stellen zudem eine Möglichkeit dar, im Sinne von Design Patterns wiederkehrende Muster bzgl. der Navigation durch eine Modellkomponente zu beschreiben und innerhalb der Website aufgrund ihrer Parametrisierbarkeit beliebig oft wieder zu verwenden.

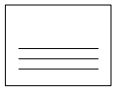
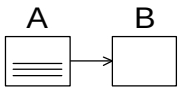
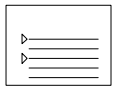
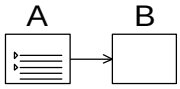
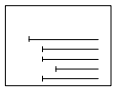
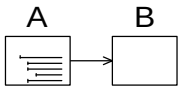
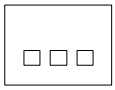
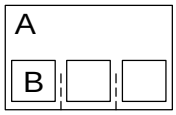
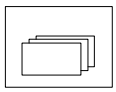
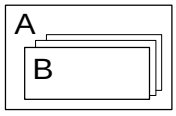
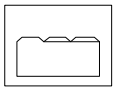
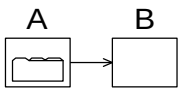
Typ	Erläuterung	Modellierung	Tiefe	Ziel-bereich	# Inst. im Zielber.	Varianten
 Index	Auflistung der Instanz- bzw. Konzeptmenge	 A = Modellklasse B = Container	1	B	1	- Anordnung horizontal / vertikal
 Group	Instanz- bzw. Konzeptmenge gruppiert	 A = Modellklasse B = Container	1	B	1	- Anordnung horizontal / vertikal für jede Ebene - Gruppierung manuell oder automatisch
 Tree	Hierarchische Anordnung der Elemente in einer Baumstruktur	 A = Modellklasse B = Container	≥ 1	B	1	
 Panel	Parallele Darstellung der Instanz- bzw. Konzeptmenge	 A = Modellklasse mit Container B = Container	1	-	n	- Anordnung horizontal / vertikal
 Sequence	Instanz- bzw. Konzeptmenge mit sequentieller Navigation	 A = Modellklasse mit Container B = Container	1	-	1	
 Tabs	Instanz- bzw. Konzeptmenge mit Navigation über Karteireiter	 A = Modellklasse B = Container	1	B	1	

Tabelle 6-1: Darstellungsformen navigationaler Klassen

6.1.6 Navigationale statische Klasse

Eine vereinfachte Art einer navigationalen Klasse stellt die *navigationale statische Klasse* dar, die sich dahingehend von navigationalen Klassen unterscheidet, dass sie die Menge der Instanzen nicht zur Laufzeit automatisch ermittelt und dynamisch darstellt, sondern ihr bereits zur Modellierungszeit eine statische Menge von Navigationsrelationen zugeordnet wird. Hierdurch lassen sich die Darstellungsformen für navigationale Klassen auch auf eine vordefinierte Menge statischer Inhalte anwenden. Während bei einer navigationalen Klasse die den aufgelisteten Instanzen zugeordneten Links dynamisch generiert werden und sich lediglich bzgl. ihrer Ursprungsinstanz unterscheiden, können die Elemente der statischen Klasse zudem verschiedene Zielcontainer aufweisen. Navigationale statische Klassen setzen sich daher aus einem Navigationstyp und einer sortierten Liste von Navigationsrelationen zusammen. Sie können nur für Navigationstypen mit Tiefe eins verwendet werden.

Definition 6.6 *Eine* navigationale statische Klasse N_S *ist ein Tripel* $N_S = (t, L, V_{N_S})$, *bestehend aus einem Navigationstyp* $t \in T^N \setminus \{Tree\}$, *einer Menge* L *von Navigationsrelationen und der Sortierungsfunktion* V_{N_S}.

$\boldsymbol{N}_S \subseteq \boldsymbol{M}$ bezeichnet die Menge der navigationalen statischen Klassen.

6.1.7 Abstrakte Serviceklassen

Abstrakte Serviceklassen dienen der Anbindung externer Applikationen bzw. Dienste und ermöglichen somit die Integration bereits existierender Applikationsmodule oder Web Services in eine Web-Anwendung. Zur Modellierungszeit wird über eine definierte Schnittstelle der externen Anwendung eine Methode der Applikation festgelegt, die zur Laufzeit von der Serviceklasse aufgerufen wird. Falls für den Methodenaufruf die Angabe von Parametern erforderlich ist, können diese auf drei Arten festgelegt werden:

1. *Eingabe*: Für einen Parameter wird von der Serviceklasse ein Feld für die Benutzereingabe generiert.
2. *Statisch*: Ein Parameter wird zur Modellierungszeit statisch festgelegt.
3. *Dynamisch*: Ein Parameter wird erst zur Laufzeit ermittelt.

Prinzipiell ist die Kombination der unterschiedlichen Arten von Parametern im Methodenaufruf möglich. Ein Parameter setzt sich aus einem Parameternamen, einer Parameterart (Eingabe, statisch, dynamisch), einem Parametertyp und einem Parameterwert zusammen.

Der Rückgabewert des Methodenaufrufes kann auf zwei unterschiedliche Arten von der Serviceklasse verarbeitet werden. Entweder wird das Ergebnis anhand eines zuvor in der Serviceklasse definierten Ausgabefeldes dargestellt, oder es wird ein Ereignis ausgelöst, das entsprechend dem Broadcast-Mechanismus (vgl. Abschnitt 6.2.1) von anderen Komponenten aufgenommen und verarbeitet werden kann. Die unterschiedlichen Arten der Verarbeitung werden in der Menge Y beschrieben.

Durch die Angabe eines Servicetyps, der bestimmt, ob es sich bei der Serviceklasse um einen Web Service oder eine andere externe Applikation handelt, lässt sich zur Modellierungszeit die Auswahl eines Dienstes anhand eines Verzeichnisses realisieren. Beispielsweise kann die Wahl eines Web Service auf der Grundlage eines UDDI[10]-Verzeichniseintrags erfolgen. Die Menge der Servicetypen einer abstrakten Serviceklasse wird mit T^S, die Menge der Methoden in Abhängigkeit des Servicetyps $t \in T^S$ mit F_t bezeichnet.

Definition 6.7 *Sei T^S die Menge abstrakter Servicetypen, $w = w_1,...,w_n$ eine Folge von Parametern und $f \in F_t$ eine Methode. Eine* abstrakte Serviceklasse *ist ein Quadrupel $S = (t, f, w, y)$ mit $t \in T^S$ und $y \in Y$.*

Die Menge der abstrakten Serviceklassen wird mit $\boldsymbol{S \subseteq M}$ bezeichnet. Abbildung 6-4 zeigt die grafische Darstellung einer abstrakten Komponente.

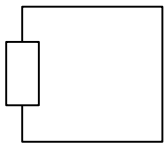


Abbildung 6-4: Grafische Darstellung einer abstrakten Serviceklasse

6.2 Kompositionsmodell

Das Kompositionsmodell definiert die Zusammensetzung beliebiger Komponenten zu einzelnen Webseiten. Es spezifiziert mit Hilfe von Containern und Partitionen die Struktur einer Webseite sowie die assoziierten Modellklassen der einzelnen Container. Als Modellklassen werden primitive Klassen, navigationale Klassen und Serviceklassen bezeichnet.

Die Zuweisung einer Modellklasse zu einem Container erfolgt durch die Typisierungsfunktion h. Einem Container können mehrere Modellklassen zugeordnet werden.

[10] http://www.uddi.org

Definition 6.8 *Seien* $\boldsymbol{M} = \boldsymbol{P}_S \cup \boldsymbol{P}_D \cup \boldsymbol{N}_S \cup \boldsymbol{N}_D \cup \boldsymbol{S} \cup \boldsymbol{P}_Z$ *die Menge der Modellklassen und* $\boldsymbol{B}$ *eine Menge von Containern. Die* Typisierungsfunktion *ist definiert als* $h : \boldsymbol{B} \rightarrow 2^{\boldsymbol{M}}$.

6.2.1 Ereignisse und Navigationsrelationen

Eine Veränderung der auf einer Webseite dargestellten Inhalte wird durch ein *Ereignis* hervorgerufen, das von einem Benutzer durch Interaktion über einen Link ausgelöst wird. Zusätzlich besteht die Möglichkeit, dass Ereignisse beispielsweise durch Überschreitung einer Zeitspanne oder durch vorangegangene Ereignisse automatisch initiiert werden.

Aufgrund eines Ereignisses wird eine *Navigationsrelation* aktiviert, was zu einer Veränderung der Darstellung in der Webanwendung führt. Die Zielkomponente einer Navigationsrelation definiert den Container, in dem die Änderungen durchgeführt werden sollen oder die nach Aktivierung des Links gültige Modellklasse. In der Navigationsrelation wird zudem der aktuelle Kontext in Form der Ursprungsinstanz mitgeführt. Diese ist die im aktuellen Web-Kontext sichtbare Instanz, die den Link auf der generierten Webseite darstellt. Die Übergabe des Kontextes ist notwendig, um Abhängigkeiten der neu aufzubauenden Informationen mit dem aktuellen Kontext zu bedienen.

Die Darstellung der kontextabhängigen Informationen erfolgt ausschließlich über primitive dynamische Klassen und navigationale Klassen. Bereits zur Modellierungszeit werden in diesen Modellklassen das Zielkonzept und die Relation zwischen dem Ursprungskonzept und dem Zielkonzept festgelegt, so dass zur Laufzeit lediglich die Ermittlung der zugehörigen Instanzmenge aussteht. Diese ist abhängig von der Ursprungsinstanz, die erst zur Laufzeit bekannt ist. Anhand der Ursprungsinstanz, dem Zielkonzept und der zugehörigen Relation ist es möglich, die darzustellende Instanz bzw. Instanzmenge zu ermitteln. Den hierzu notwendigen Kontext erhalten die Modellklassen über ihren assoziierten Container.

Abbildung 6-5 soll diesen Zusammenhang verdeutlichen. Seien $i_k, i_l \in I$ Instanzen des Konzeptes $c_x \in \boldsymbol{C}$ und $i_p, i_q, i_r \in I$ Instanzen des Konzeptes $c_y \in \boldsymbol{C}$. Seien weiter i_l die Ursprungsinstanz, c_y das Zielkonzept und $p \in \boldsymbol{L}^{\boldsymbol{R}}$ eine Eigenschaft von c_x. Mit Hilfe des zugehörigen Konzeptes der Ursprungsinstanz, dem Zielkonzept und der Eigenschaft p erhält man die Relationen r_1^i und r_2^i und somit die darzustellenden Instanzen i_p und i_q.

Definition 6.9 *Eine* Navigationsrelation L *ist ein 7-Tupel* $L = (i_{c_s}, c_t, p, k_s, k_t, e, v)$, *bestehend aus einer Ursprungsinstanz* $i_{c_s} \in I$, *einem Zielkonzept* $c_t \in \boldsymbol{C}$, *einer*

Eigenschaft $p \in \boldsymbol{L^R}$, *einer Ursprungskomponente* $k_s \in \boldsymbol{B} \cup \boldsymbol{M}$, *einer Zielkomponente* $k_t \in \boldsymbol{B} \cup \boldsymbol{M}$, *einem Ereignis* $e \in E$ *und einer textuellen Beschriftung* $v \in V$.

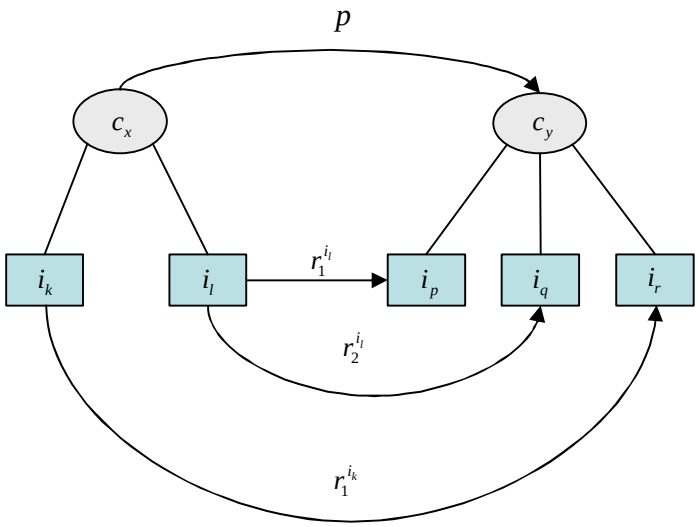


Abbildung 6-5: Ermittlung der Zielinstanzen

Die Funktionsweise der Navigationsrelationen unterscheidet sich in Abhängigkeit davon, ob es sich im Entwurfsmodell bei der Ursprungskomponente um eine Modellklasse oder um einen Container handelt. Im ersten Fall werden innerhalb des durch die Modellklasse repräsentierten Inhalts alle textuellen Vorkommen der Beschriftung v mit einem Link versehen. Ist $v = \varnothing$, bezieht sich der Link auf den gesamten Inhalt der Modellklasse. Im anderen Fall wird die in der Navigationsrelation definierte Beschriftung v als Link in den Container eingefügt. Es handelt sich hierbei um eine Vereinfachung der grafischen Modellierung: Statt eine von einem Container $b \in \boldsymbol{B}$ ausgehende Navigationsrelation $l \in L$ zu zeichnen, müsste eine primitive statische Klasse, deren Inhalt die textuelle Beschriftung von l enthält, in den Container b eingefügt werden und dieser Modellklasse dann die Navigationsrelation l zugeordnet werden.

In der grafischen Repräsentation des Navigationsmodells definieren Navigationsrelationen den Übergang von einer Komponente zu einer anderen. Entsprechend Abbildung 6-6 (a) wird eine Navigationsrelation mit einem beschrifteten Pfeil dargestellt. Soll nach der Ausführung der Navigationsrelation im Sinne eines *Broadcast* ein weiteres Ereignis ausgelöst werden, wird der zusätzliche Link bei der Modellierung mit in die Beschriftung des Pfeils aufgenommen (vgl. Abbildung 6-6 (b)).

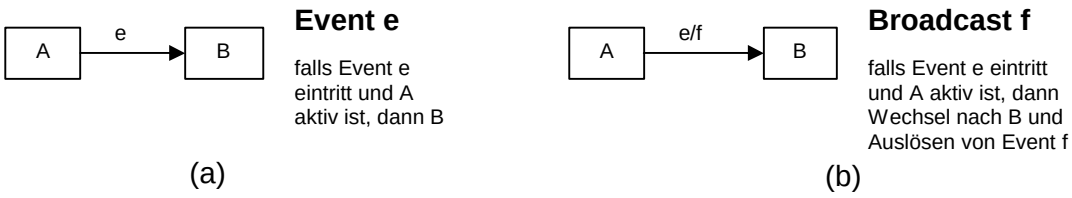


Abbildung 6-6: (a) Aufruf von B aufgrund der Navigationsrelation e; (b) Aktivierung der Navigationsrelation f nach Aufruf von e

6.2.2 Zustandserfassung durch Navigationsrelationen

Der Broadcasting-Mechanismus lässt sich des Weiteren zur Festlegung von Anwendungszuständen nutzen. Hierdurch kann das Navigationsverhalten des Besuchers aufgezeichnet werden und sowohl die inhaltliche als auch die grafische Darstellung in Abhängigkeit des bisherigen Navigationsverlaufes erfolgen. Die Abhängigkeit eines Anwendungszustandes von einer Navigationsrelation wird anhand einer *Zustandstabelle* festgehalten, in der für jedes Broadcast-Ereignis der Zustand eines bestimmten Bezeichners definiert werden kann. Tabelle 6-2 zeigt beispielhaft die Festlegung einer Zustandsänderung für den Bezeichner ‘ROLE’ in Abhängigkeit des Broadcast-Ereignisses.

Ereignis	Bezeichner	Wert
f	ROLE	Investor
g	ROLE	Kunde
h	LOCATION	München

Tabelle 6-2: Zustandsänderungen durch Navigationsrelationen

6.2.3 Primitive Zustandsklassen

Primitive Zustandsklassen können dazu verwendet werden, den aktuellen Kontext in Form des zu dieser Zeit gültigen Konzeptes des konzeptuellen Modells darzustellen, oder über den gegenwärtigen Zustand eines bestimmten Bezeichners zu informieren.

Definition 6.10 *Seien* $\boldsymbol{B}$ *eine Menge von Containern und* Z *eine Menge von Zustandsbezeichnern. Dann heißt* $P_Z = (z)$ *mit* $z \in \boldsymbol{B} \cup Z$ primitive Zustandsklasse.

Die Menge der primitiven Zustandsklassen wird mit $\boldsymbol{P}_Z \subseteq \boldsymbol{M}$ bezeichnet.

6.2.4 Linkkollektionen

Mehrere von einer Modellklasse ausgehende Links zu einer anderen Modellklasse oder einem anderen Container können in der grafischen Darstellung des Navigationsmodells zur besseren Übersicht in sog. Linkkollektionen zusammengefasst werden. Dies ist jedoch nur möglich, falls es sich bei der Ursprungskomponente um eine primitive statische Klasse handelt, da navigationale Klassen in der grafischen Beschreibung nur einen Link zu einem Zielbereich enthalten, Serviceklassen nicht mehrere unterschiedliche ausgehende Links besitzen können und bzgl. primitiver dynamischer Modellklassen sich eine Modellierung statischer Inhalte per Definition ausschließt.

Diese Notation ist vor allem dann vereinfachend, wenn Inhalte primitiver statischer Modellklassen mehrere Links enthalten, wie beispielsweise Hyperlinks innerhalb größerer Textpassagen. Linkkollektionen werden grafisch durch einen nicht ausgefüllten geschlossenen Pfeil dargestellt.

6.2.5 Popup-Fenster

Stellt der Zielbereich einer Navigationsrelation ein Popup-Fenster dar, wird dies bei der Modellierung mit Hilfe des Ereignis-Parameters der Navigationsrelation spezifiziert. Zusätzlich muss festgelegt werden, welche Aktion bzgl. des Popup-Fensters auszuführen ist. Die Aktion entspricht entweder dem Öffnen oder dem Schließen eines Popup-Fensters. Eine Navigationsrelation, die zu einem Container führt, der als Popup-Fenster dargestellt werden soll, wird gemäß Abbildung 6-7 modelliert.

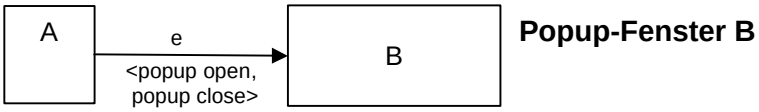


Abbildung 6-7: Modellierung eines Popup-Fenster mit Hilfe einer Navigationsrelation

6.3 Navigationsmodell

Das Navigationsmodell setzt sich aus den Kompositionsmodellen der einzelnen Webseiten und einer zusätzlichen übergeordneten Navigationsstruktur zusammen. Diese legt die Navigation zwischen den einzelnen Komponenten und Webseiten fest.

Die grafische Beschreibung des Navigationsmodells verwendet einen auf der Statechart-Notation basierenden Ansatz[11]. Hierbei werden atomare und nicht-atomare Komponenten sowie Container und Partitionen durch gerichtete Navigationsrelationen miteinander verbunden. Der Bereich, der später im Webbrowser für die Darstellung einer oder mehrerer Komponenten zur Verfügung gestellt wird, lässt sich mit Hilfe von Containern und Partitionen festlegen. Hierbei können sowohl konjunktive als auch disjunktive Sichtenstrukturen beschrieben werden. Konjunktive Sichtenstrukturen, bei denen alle unmittelbar enthaltenen Komponenten gleichzeitig sichtbar sind, entstehen durch Partitionierung eines Containers. Hingegen bilden mehrere Komponenten gleicher Ebene in einem unpartitionierten Bereich eine disjunktive Sichtenstruktur.

Abbildung 6-8 zeigt beispielhaft das Navigationsmodell einer Portalseite, das den

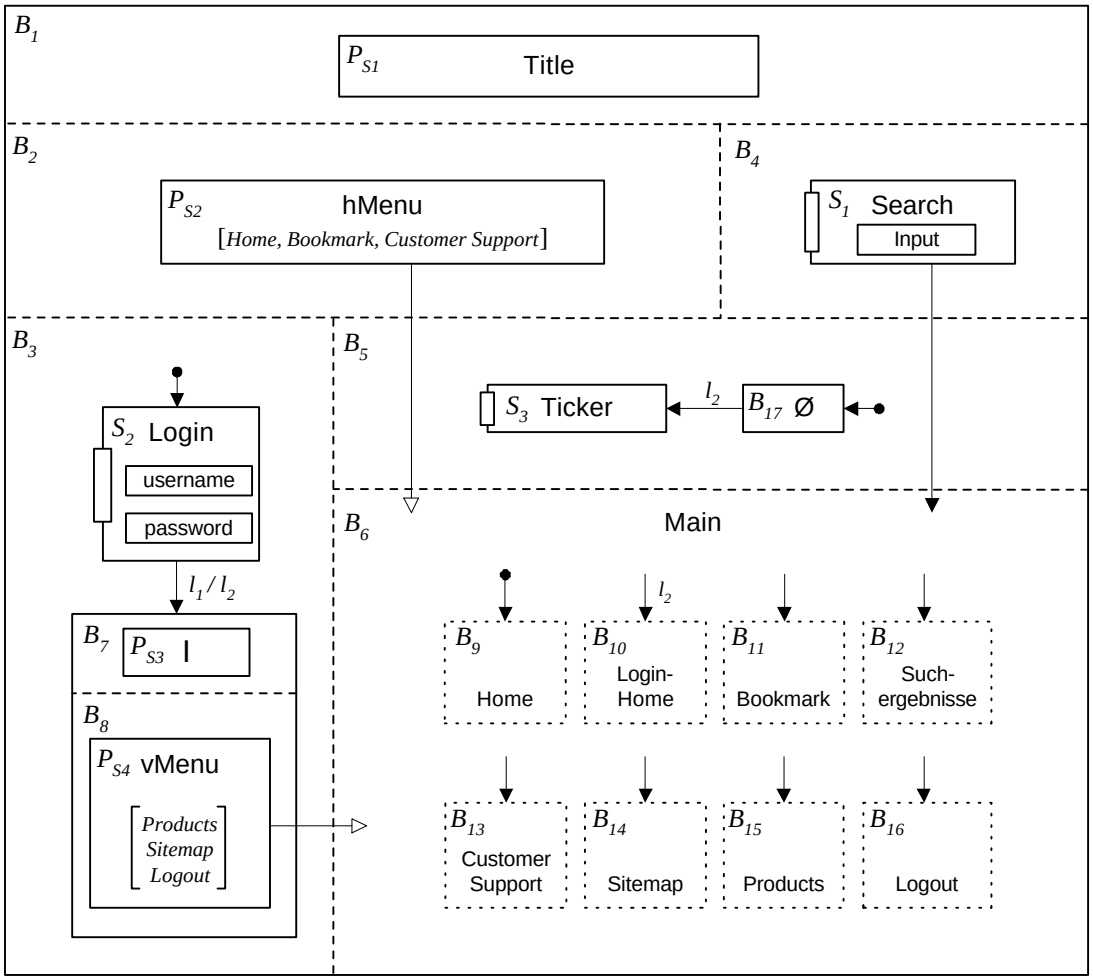


Abbildung 6-8: Navigationsmodell (Auszug aus einem Portal)

[11] Eine Übersicht über die Verwendung der grafischen Notation befindet sich im Anhang D.

Rahmen der entsprechenden Anwendung darstellt. In der gesamten Web-Applikation findet sich im oberen Bereich eine Titelzeile, unmittelbar darunter liegt eine durch eine Linkkollektion realisierte Menüleiste mit einem rechts angehefteten Suchfeld in Form einer Applikation. Unterhalb dieser beiden Komponenten befindet sich eine weitere Partition. Diese enthält im Startzustand links zunächst ein Login-Feld, das nach erfolgreicher Anmeldung durch ein vertikales Menü mit einem darüber liegenden Bild ('I' steht hier für 'Image') ersetzt wird. Rechts davon wird im oberen Bereich zunächst nichts angezeigt ('Ø' steht hier für die leere Menge). Erst nachdem die Anmeldung abgeschlossen ist, wird der Ticker aktiviert. Der untere Bereich stellt den Hauptbereich der Anwendung dar. Er ist gleichzeitig der Zielbereich für die beiden Menüleisten und das Suchfeld. Die verschiedenen Komponenten, die in diesem Bereich dargestellt werden, können nun an anderer Stelle genauer spezifiziert werden.

Bei der Portalseite handelt es sich in diesem Beispiel um die Homepage eines Unternehmens. Die Ursprungsinstanz des konkreten Unternehmens ist auf der Einstiegsseite und somit im obersten Container bekannt. In einem weiteren Beispiel soll nun die Modellierung dynamischer Inhalte, die als Instanzen in der zugrunde liegenden Ontologie vorliegen, verdeutlicht werden. Hierzu werden primitive dynamische Klassen als auch navigationale Klassen eingesetzt. Abbildung 6-9 zeigt die grafische Beschreibung des Kompositionsmodells für die Komponente 'Products' aus Abbildung 6-8.

Ausgehend von der Homepage des Unternehmens erhält der Benutzer im linken Bereich eine Übersicht über die Produkte in Form der Produktnamen. Im rechten Bereich wird zu einem im Produktnavigator ausgewählten Produkt zunächst die Produktbeschreibung angezeigt. Von hier aus können über die Navigationsrelation l_2 stellvertretend auch Testergebnisse zusammen mit einer Auflistung von Anwendungen präsentiert werden. Für eine ausführlichere Dokumentation einer im Anwendungsnavigator gewählten Anwendung steht die Navigationsrelation l_4 zur Verfügung, die dem Benutzer die gewünschte Anwendungsdokumentation liefert.

Über den Link l_5 erhält man wieder die Darstellung der Testergebnisse mit den zugehörigen Anwendungen. Mittels l_3 gelangt man zurück zur Produktbeschreibung.

In diesem Beispiel steht bereits zur Modellierungszeit fest, dass ausschließlich die Produkte der Firma X dargestellt werden sollen. Zur Laufzeit ergibt sich daher zunächst folgende Initialisierung:

$B_{15} = B_{18} = B_{19} =$ (main, Unternehmen, Firma X :: Unternehmen)
$N_{D1} =$ (Index, 'Instanz', Firma X :: Unternehmen, Produkt, 1, {'produces'}, {Produktname}, $\{V\}$, $\{B_{19}\}$).

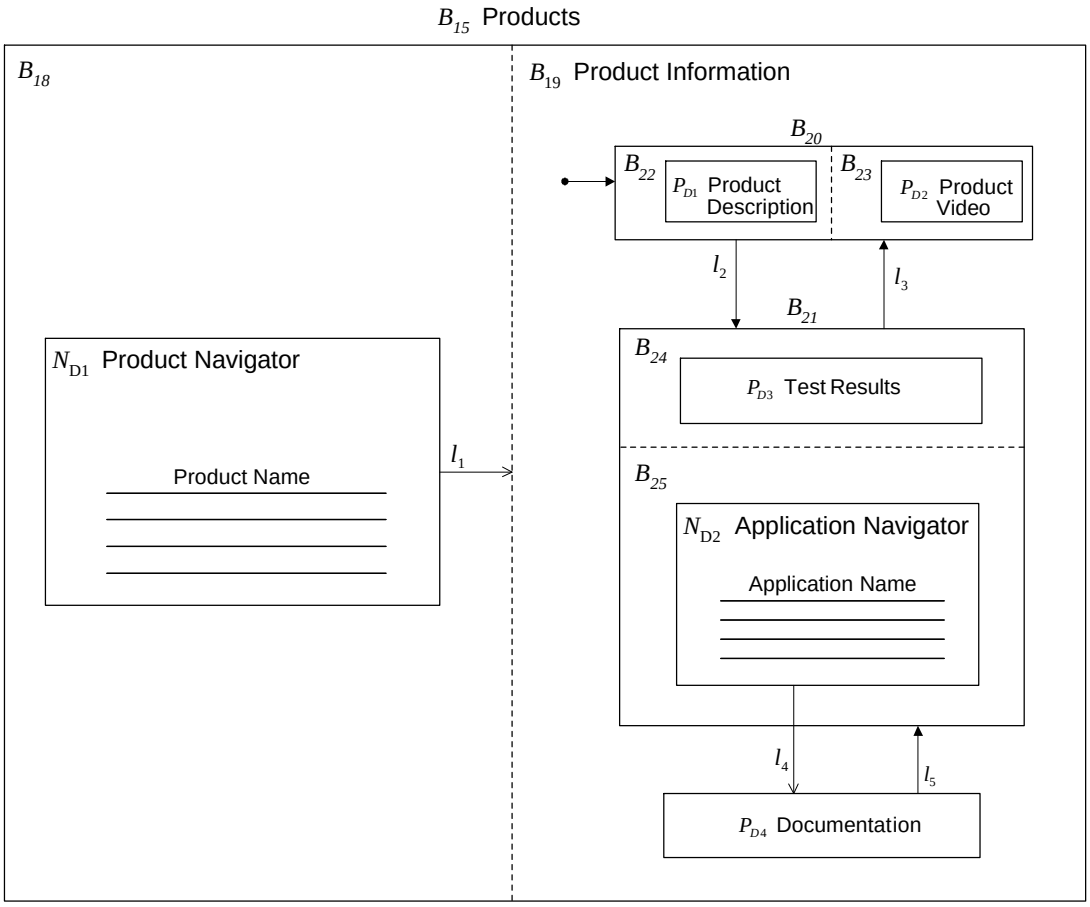


Abbildung 6-9: Grafische Beschreibung der Komponente „Products“

Mit dieser Information werden in der navigationalen Klasse N_{D1} alle Produkte wie folgt als Link aufgelistet:

$l_{1,1}$ = (Produkt1 :: Produkt, Produkt, $\varnothing$, N_{D1}, B_{19}, ‘click‘, ‘Produkt 1’),
$l_{1,2}$ = (Produkt2 :: Produkt, Produkt, $\varnothing$, N_{D1}, B_{19}, ‘click‘, ‘Produkt 2’),
…

Der Zielbereich der navigationalen Klasse sowie der darin enthaltene Startcontainer werden mit der ersten Instanz, also mit ‘Produkt1’ initialisiert:

$B_{19} = B_{20}$ = (N_{D1}, Produkt, Produkt1 :: Produkt).

Für die in B_{22} enthaltene Modellklasse und der zugeordneten Navigationsrelation l_2 folgt:

P_{D1} = (String, Produkt1 :: Produkt, Beschreibung, ‘has_a’),

$l_2 =$ (Beschreibung1 :: Beschreibung, Testergebnis, $\varnothing$, P_{D1}, B_{21}, ‘click‘, ‘Tests’).

Bei Aktivierung von l_2 werden B_{21} und die darin enthaltenen Partitionen B_{24} und B_{25} mit

$B_{21} = B_{24} = B_{25} =$ (P_{D1}, Testergebnis, Beschreibung1 :: Beschreibung)

initialisiert. Allerdings beziehen sich die Testergebnisse nicht auf die Produktbeschreibung, sondern auf das Produkt selbst. Die dem Container B_{24} übergebene Ursprungsinstanz ‘Beschreibung1’ ist an dieser Stelle nutzlos, da in der Ontologie keine Assoziation zwischen dem Konzept ‘Beschreibung’ und dem in der primitiven dynamischen Klasse P_{D3} modellierten Konzept ‘Testergebnis’ existiert. Durch Betrachtung der im übergebenen Kontext enthaltenen Ursprungsmodellklasse P_{D1} mittels Rückverfolgung kann die Instanz ‘Produkt1’ gefunden werden, die einen Bezug zu dem Konzept ‘Testergebnis’ besitzt, so dass die der Instanz ‘Produkt1’ zugeordneten Testergebnisse dargestellt werden können. Die Ermittlung der Inhalte für die verbleibenden Modellklassen ergibt sich analog.

7 Kontext- und Sichtenmodellierung

Immer mehr Websites bieten eine personalisierte Darstellung ihrer Inhalte an. Vor allem für den Bereich des E-Commerce ist die benutzerbezogene Aufbereitung der Inhalte von hoher Bedeutung, da hierdurch direktes Marketing auf der Basis eines individuell an den Kunden angepassten User Interfaces realisiert werden kann. So lassen sich z. B. Produkte, von denen sich der Anbieter ein besonderes Interesse seitens bestimmter Kunden verspricht, an zentraler Position hervorheben bzw. die Navigationspfade zu diesen Produkten vereinfachen. Ferner ermöglicht die Berücksichtigung situationsbezogener Informationen eine Anpassung der darzustellenden Inhalte beispielsweise an örtliche oder zeitliche Gegebenheiten.

Soll eine Website verschiedene Versionen für die unterschiedlichen Benutzerinteressen bereitstellen, existieren zwei Vorgehensweisen, die sich jedoch nicht gegenseitig ausschließen. Zum einen können beim Entwurf einer Web-Anwendung Kategorien von Benutzern und deren unterschiedliche Interessen analysiert werden. Für jede dieser Benutzergruppen lassen sich dann verschiedene Versionen der Website generieren. Voraussetzung ist, dass die wesentlichen Merkmale jeder Benutzergruppe zunächst modelliert und festgehalten werden. Zum anderen können Interessen einzelner Benutzer berücksichtigt werden. Hierzu ist ein sog. *User Profile* erforderlich, das entweder der Benutzer nach der Registrierung erstellen bzw. modifizieren kann oder das von der Web-Anwendung mit Standardwerten initialisiert wird. Durch Erfassen (sog. Tracking) der Benutzerinteraktion wird das User Profile dem Verhalten des Benutzers angepasst, so dass sich hieraus Interessen des Benutzers erschließen lassen.[12]

Des Weiteren besteht die Möglichkeit, einem Benutzer aufgrund von Ereignissen bestimmte Informationen anzubieten. Beispielsweise könnte der Besucher über Neuigkeiten zu für ihn interessante Themen informiert werden. Ebenfalls wäre es möglich, ihn auf Personen aufmerksam zu machen, die er treffen möchte und die sich in seiner Nähe befinden.

Um die verschiedenen Möglichkeiten zur Personalisierung beim Entwurf einer Web-Applikation methodisch zu unterstützen, bedarf es der Modellierung von Zugriffs- und Sichtenstrukturen. Das Navigationsmodell liefert eine grafische und formale Beschreibung der Struktur und der thematischen Inhalte einer Website. Es repräsentiert eine einheitliche Sicht auf die darzustellenden Informationen, unabhängig vom Kontext, in dem der Benutzer auf die Inhalte zugreift. Die Berücksichtigung von

[12] Bei der Erfassung des Benutzerverhaltens sind jedoch in den meisten Ländern rechtliche Bestimmungen zu beachten. In den Ländern der Europäischen Union müssen die benutzerbezogenen Daten vor dem Zugriff Dritter sicher sein und es bedarf darüber hinaus einer Genehmigung zur Datenerfassung seitens des Benutzers.

Kontextfaktoren wie beispielsweise die Rolle des Benutzers, sein gegenwärtiger Aufenthaltsort, der Zeitpunkt und das von ihm zur Interaktion genutzte Gerät, ist jedoch notwendig, um ein den Anforderungen des Benutzers angepasstes Informationsangebot zu erzeugen.

Die Kontextmodellierung lässt sich prinzipiell in drei Bereiche gliedern (vgl. Schmidt et al., 1999): Der *technische Kontext* beinhaltet technische Rahmenbedingungen, wie Hard- und Softwareeigenschaften des benutzten Endgerätes, Eigenschaften des Netzwerks und den Zustand der Anwendung selbst. Diese Angaben sind notwendig, um bei der Bereitstellung von Inhalten durch die Web-Anwendung Faktoren wie z. B. Displaygröße und Übertragungsrate zu berücksichtigen. Orts- und zeitbezogene Informationen sind im *natürlichen Kontext* enthalten. Hierdurch erhält die Web-Anwendung einerseits die Möglichkeit, Informationen in Abhängigkeit des gegenwärtigen Standortes zu sortieren, und andererseits Gültigkeitszeiträume in die Auswahl der darzustellenden Inhalte einzubeziehen. Der *soziale Kontext* umfasst schließlich die Beschreibung des Benutzers bzw. der Rollen, die der Benutzer einnimmt.

Jeder Modellklasse des Navigationsmodells kann eine *Zugriffsklasse* zugeordnet werden. Die Modellklassen definieren lediglich die Inhalte, die in den entsprechenden Containern dargestellt werden sollen, jedoch keine Bedingungen für den Zugriff auf diese Inhalte. Dies bedeutet, dass Nebenbedingungen, die sich aus der Berücksichtigung des Kontextes ergeben, im Navigationsmodell nicht aufgenommen werden. Die Aufgabe der Zugriffsklassen liegt nun darin, diese Nebenbedingungen festzulegen und dementsprechend den Zugriff auf die Inhalte und somit ihre Sichtbarkeit in Abhängigkeit von unterschiedlichen Benutzerrollen zu definieren. Zur Betrachtung verschiedener Kontexte in der Web-Anwendung werden ggf. mehrere Zugriffsklassen für eine Modellklasse benötigt.

Die Modellierung der Kontextfaktoren mit Hilfe der Zugriffsklassen erfolgt anhand des Komponenten- bzw. Navigationsmodells. Für jede darin enthaltene Modellklasse können Bedingungen hinzugefügt werden, die über ihre Sichtbarkeit in Abhängigkeit des Kontextes entscheiden. Beispielsweise ist es möglich, die Darstellung ganzer Teilbereiche für eine bestimmte Person, Benutzerrolle oder ein spezielles Endgerät zu verhindern.

7.1 Zugriffsklassen und Sichtenmodell

Eine Zugriffsklasse setzt sich aus zwei Teilen zusammen: einer Zuordnungsangabe und einem Bedingungsteil. Die Zuordnungsangabe legt den Benutzer bzw. die Rolle fest, auf den sich die Zugriffsklasse bezieht. Der Bedingungsteil enthält einen beliebigen booleschen Ausdruck, der erfüllt sein muss, um den in der zugehörigen

Modellklasse festgelegten Inhalt darzustellen. Eine Zugriffsklasse ist demnach wie folgt definiert:

Definition 7.1 *Seien* $\boldsymbol{U}$ *eine Menge von Rollen und* $\boldsymbol{J}$ *eine Menge von Bedingungen. Eine* Zugriffsklasse Γ *ist ein Tripel* $\Gamma = (m, d, b)$*, bestehend aus einer Modellklasse* $m \in \boldsymbol{M}$*, einer Rolle* $d \in \boldsymbol{U}$ *und einer Bedingung* $b \in \boldsymbol{J}$ *.*

Prinzipiell ist es möglich, für verschiedene Rollen bzw. Benutzer jeweils eine Zugriffsklasse zu erstellen. Dadurch lassen sich rollenbasierte bzw. benutzerspezifische Darstellungen innerhalb der Web-Applikation realisieren. Abbildung 7-1 zeigt die grafische Beschreibung einer Zugriffsklasse.

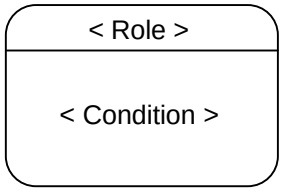


Abbildung 7-1: Grafische Beschreibung einer Zugriffsklasse

Das *Sichtenmodell* setzt sich aus einer Menge von Zugriffsklassen für jede einzelne Modellklasse zusammen. Jede Zugriffsklasse ist genau einer Modellklasse zugeordnet. Die grafische Modellierung kann auf zwei Arten erfolgen: Sollen nur wenige Sichten modelliert werden, können die Zugriffsklassen neben den zugehörigen Modellklassen positioniert und durch eine Linie verbunden werden. Im anderen Fall, d. h. bei einer umfangreichen Sichtenmodellierung, kann eine unmittelbare Integration der Zugriffsklassen in das Navigationsmodell zu einer unübersichtlichen Darstellung führen. Das Sichtenmodell ist dann als eigenständiges grafisches Modell zu betrachten, das alle Modellklassen des Navigationsmodells als Liste aufführt und jede dieser Modellklassen mit den zugeordneten Zugriffsklassen verbindet.

Einen Sonderfall bilden Navigationsrelationen, bei deren Ursprungskomponente es sich um einen Container handelt. Da in diesem Fall auf die Modellierung einer zusätzlichen primitiven statischen Klasse verzichtet wird (vgl. Kapitel 6.2.1), kann im Entwurfsmodell eine Zugriffsklasse der entsprechenden Navigationsrelation direkt zugeordnet werden.

Wird beim Entwurf der Web-Anwendung auf eine Einbeziehung sowohl personalisierter bzw. rollenbasierter Darstellungen als auch kontextbezogener Aspekte verzichtet, ist eine explizite grafische Modellierung der Zugriffsklassen nicht notwendig.

In diesem Fall wäre jeder Modellklasse eine Zugriffsklasse zugeordnet, deren Rolle mit ‘ALL’ und Bedingung mit ‘TRUE’ gekennzeichnet ist.

7.2 Modellierung kontextbezogener Bedingungen

Mit Hilfe der Zugriffsklassen ist es möglich, beliebige Bedingungen in den Entwurfsprozess einer Web-Anwendung aufzunehmen. Da die Bedingungen, die sich aus der Betrachtung der verschiedenen Kontextfaktoren ergeben frei wählbar bzw. modellierbar sind, wird mit den Zugriffsklassen eine weitestgehend generische Modellierung ermöglicht. Die Aufgabe des Sichtenmodells liegt dementsprechend in der Bereitstellung einer Grundstruktur zur Definition beliebiger kontextbezogener Bedingungen.

Für die Berücksichtigung der den Modellklassen zugeordneten Bedingungen zur Laufzeit ist jedoch eine konkrete Beschreibung erforderlich. Bevor anhand einer Auswahl von Beispielen zur Einbindung von Kontextfaktoren die konkrete Beschreibung der Bedingungen verdeutlicht wird, soll zunächst der grundlegende Mechanismus für den Zugriff auf Informationen ohne die Berücksichtigung von Bedingungen erläutert werden.

7.2.1 Grundstruktur der Zugriffsklasse

Für die Angabe kontextbezogener Nebenbedingungen ist es notwendig, die grundsätzliche Funktionsweise der Zugriffsklassen zu bestimmen. Im Normalfall, also ohne Berücksichtigung des Kontextes, ermittelt eine Zugriffsklasse die in der zugeordneten Modellklasse abstrakt beschriebenen Inhalte. Dabei muss unterschieden werden, ob es sich um statische oder dynamische Modellklassen handelt, d. h. ob die Inhalte bereits zur Modellierungszeit feststehen oder sich erst zur Laufzeit ergeben.

Im Prinzip kann der Zugriff unter Verwendung einer SQL-ähnlichen Anfragesprache erfolgen. Um die in Kapitel 4.7 aufgezeigten Vorteile regelbasierter Systeme zu nutzen, sollen die Zugriffe durch die deduktive, objektorientierte Datenbanksprache F-Logic ausgedrückt werden.

Zugriff auf statische Inhalte

Primitive statische Modellklassen zeichnen sich dadurch aus, dass der Inhalt, der später durch diese Klasse dargestellt wird, zur Modellierungszeit bestimmt wird. Die Ermittlung des Inhaltes durch die Zugriffsklasse ist trivial, da die entsprechende Instanz in der Modellklasse $P_s = (t, i)$ mit $t \in T^{P_s} \subset \boldsymbol{C}$ und $i \in I$ definiert ist:

$$\forall X \leftarrow X : t \wedge equals(X, i).$$

Der Ausdruck beschreibt den Zugriff auf alle Instanzen des Konzeptes t der Ontologie, die gleich der Instanz i sind.

Zugriff auf dynamische Inhalte

Soll die Instanzmenge dynamisch, also zur Laufzeit, ermittelt werden, ist die Angabe der Ursprungsinstanz $i_s \in I$, des Zielkonzeptes $c_t \in \boldsymbol{C}$ und der entsprechenden Eigenschaft $p \in \boldsymbol{L}^{\boldsymbol{R}}$ der Ursprungsinstanz notwendig. Sowohl die primitiven dynamischen Modellklassen als auch die navigationalen Klassen enthalten diese Angaben. Der Zugriff auf die Instanzen ergibt sich wie folgt:

$$\forall X \leftarrow X : c_t \wedge i_s[p \rightarrow X].$$

Die aufgeführten Regeln für den Zugriff auf statische bzw. dynamische Inhalte bilden die Grundlage für Erweiterungen hinsichtlich modellierbarer Nebenbedingungen. Die Modellierung der verschiedenen kontextbezogenen Faktoren und deren Definition in den Zugriffsklassen werden im Folgenden näher erläutert.

7.2.2 Modellierung und Integration der Kontextfaktoren

Grundsätzlich ist es möglich, beliebige Kontextfaktoren in einer Web-Anwendung zu berücksichtigen und somit auch zu modellieren. Diese Kontextfaktoren sind von den Zielen der Web-Anwendung abhängig, so dass die Modellierung der Kontextfaktoren beim Entwurf einer Web-Applikation stattfindet. Aus diesem Grund soll an dieser Stelle unter Verwendung des Metamodells der Sichtenbeschreibung der prinzipielle Mechanismus der Kontextmodellierung anhand einiger Beispiele für die verschiedenen Kontextarten erläutert werden.

Technischer Kontext

Der technische Kontext enthält Angaben über das verwendete Endgerät, dessen Kommunikationsmöglichkeiten mit der Web-Anwendung und den Zustand der Anwendung selbst. Beim Entwurf der Web-Applikation muss darauf geachtet werden, dass zur Laufzeit diese Informationen ermittelt werden können. Dies könnte z. B. durch Übertragung der Geräteinformationen des Endgerätes oder durch die Verwaltung von Geräteprofilen seitens der Web-Anwendung erfolgen.

Die Einbeziehung technischer Kontextfaktoren ermöglicht eine gerätespezifische Informationsdarstellung und -übermittlung. Eine Web-Anwendung kann beispielsweise darüber entscheiden, ob umfangreiche Multimedia-Dateien bei geringer Bandbreite übertragen werden sollen, ob bestimmte Dateitypen (Bilder, Streams etc.) überhaupt auf dem Endgerät darstellbar sind oder ob zusätzliche Informationen einer Seite auf einem kleinen Display ausgelassen werden sollen.

Das folgende Beispiel soll die Modellierung einer kontextabhängigen Darstellung mit Hilfe einer Zugriffsklasse und die Erstellung der zugehörigen Regel für die an-

gepasste Informationsdarstellung in der Web-Anwendung verdeutlichen. Ausgehend von Abbildung 6-9 soll das dort modellierte Produktvideo nur dann gezeigt werden, wenn die Übertragungsrate zum Endgerät über 33000 Bits pro Sekunde liegt.

Entsprechend Abbildung 7-2 wird für die Modellklasse P_{D2} eine Zugriffsklasse erzeugt, die für alle Besucher der Website gültig ist und den zugehörigen Inhalt nur dann darstellt, wenn die Bedingung erfüllt ist. Hieraus lässt sich für den Zugriff auf das Video eines zur Laufzeit gewählten Produktes p folgende Regel ableiten:

$$\forall X \leftarrow p[\text{has\_Video} \rightarrow X] \wedge \mathit{greater}(\text{device.BPS}, 33000)$$

Da erst zur Laufzeit bekannt ist, für welches Produkt das zugehörige Video bereitgestellt werden soll, ist ein dynamischer Zugriff notwendig. Die Basisregel für den Zugriff auf dynamische Inhalte wird daher um die entsprechende Bedingung erweitert.

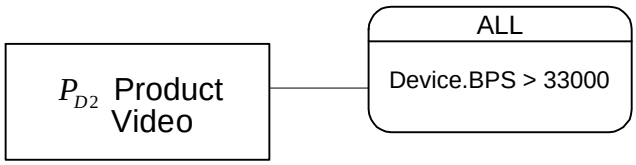


Abbildung 7-2: Zugriffsklasse zur Modellklasse P_{D2}

Natürlicher Kontext

Zeit- und ortsbezogene Informationen sind im natürlichen Kontext enthalten. Diese Informationen können genutzt werden, um für bestimmte Inhalte einer Website eine Gültigkeitsdauer festzulegen bzw. Inhalte ortsabhängig zu präsentieren oder zu sortieren. Auch hier existieren beliebige Anwendungsmöglichkeiten. Als Beispiel sollen auf einer Webseite alle Restaurants aufgelistet werden, die sich in unmittelbarer Umgebung des Benutzers befinden und gleichzeitig noch geöffnet sind (Abbildung 7-3).

Die der Zugriffsklasse zugrunde liegende Bedingung enthält zwei Funktionen. *Near* verifiziert die örtliche Nähe des Benutzers zur angegebenen Position, in diesem Fall den Ort des Restaurants, während die Funktion *During* überprüft, ob die lokale Zeit mit der Zeitspanne der Öffnungszeiten des Restaurants übereinstimmt. Die Methoden müssen zur Laufzeit verfügbar sein, d. h. eine explizite Implementierung der Funktionalitäten ist notwendig.

Daneben können zusätzliche Methoden weitere orts- oder zeitbezogene Relationen zur Verfügung stellen, beispielsweise zur Überprüfung der exakten Position des

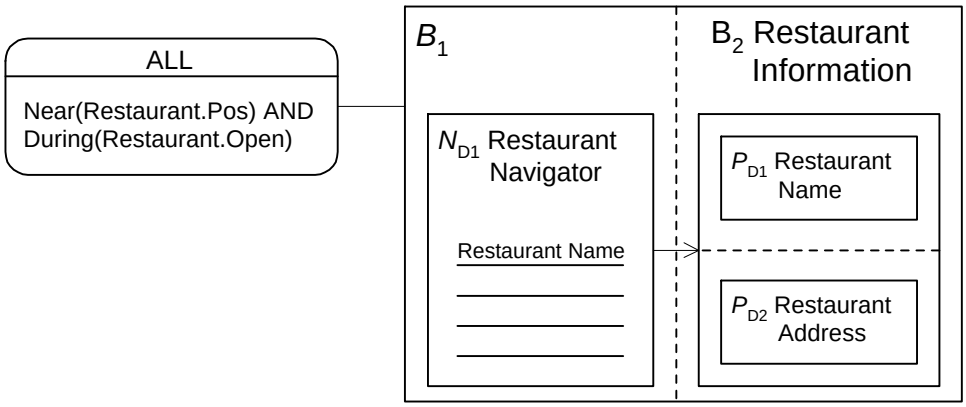


Abbildung 7-3: Zugriffsklasse zur orts- und zeitabhängigen Präsentation

Benutzers, seinem räumlichen Aufenthaltsort innerhalb eines Gebäudes oder zum Vergleich der lokalen Zeit mit einem festgelegten Zeitstempel.

Für den Zugriff auf die Inhalte, in diesem Beispiel also die Menge der präferierten Restaurants, kann unter Berücksichtigung der orts- und zeitbezogenen Bedingungen folgende Regel erstellt werden:

$$\forall X \leftarrow X : \mathrm{Restaurant}[\mathrm{position} = \mathrm{p}] \wedge near(\mathrm{position}) \wedge during(time)$$

Sozialer Kontext

Dem sozialen Kontext werden benutzerbezogene Informationen zugeordnet, die zur Personalisierung des Informationsangebotes der Website herangezogen werden können. Grundlage für die benutzerspezifische Informationszuordnung sind die Rollendefinition des Benutzers und eine Beschreibung seines persönlichen Profils, das sich seinen Vorlieben und Interessen dynamisch anpasst. Aus der Betrachtung der sozialen Umgebung können beispielsweise Informationen darüber gewonnen werden, ob sich weitere, dem Benutzer bekannte Personen, in unmittelbarer Nähe aufhalten.

Zur Definition der verschiedenen möglichen Benutzerrollen werden in der Analysephase des Web-Engineering-Prozesses zunächst alle potentiellen Benutzer der Web-Anwendung identifiziert und nach Aktivitäten sortiert. Die spezifischen Informationsbedürfnisse jeder dieser Gruppen werden jeweils einer Rolle zugeordnet, die gemeinsame Aufgaben und Inhalte abdeckt.

Informationen, die aus der Betrachtung des sozialen Umfeldes des Benutzers resultieren, können zur Laufzeit ebenfalls mit Hilfe von Inferenzregeln generiert werden. Möchte z. B. ein Benutzer eine bestimmte Person treffen, kann er durch das Auslö-

sen eines Ereignisses an das Treffen erinnert werden, falls sich die entsprechende Person in unmittelbarer Nähe befindet.

Die Modellierung derartiger Regeln müsste nicht zwingend mit Hilfe der Zugriffsklassen erfolgen, sondern könnte auf Basis spezieller navigationaler Klassen durchgeführt werden, die ausgehend von einem Benutzer dessen Aufgaben darstellen. Die Funktionalitäten sowohl zum Empfangen eines ausgelösten Ereignisses aufgrund einer aktiven Regel als auch zur Darstellung der zugehörigen Aufgabe, müssten von der navigationalen Klasse abgedeckt werden. Eine andere Möglichkeit bestünde darin, eine entsprechende Applikation mit diesen Funktionalitäten in die Web-Anwendung zu integrieren.

Neben der Betrachtung der Benutzerrolle bzw. seiner Umgebung lassen sich User Profiles erstellen, die das Nutzerverhalten innerhalb einer Web-Anwendung analysieren und daraus Interessen des Benutzers ableiten. Auch in diesem Fall würde die Berücksichtigung des User Profiles nicht durch die Modellierung von Zugriffsklassen erfolgen. Vielmehr wären spezielle navigationale Klassen notwendig, die einerseits die Auswahl der Instanzmenge anhand des Profils durchführen und andererseits zusätzliche Navigationsmöglichkeiten zu besonderen Interessensgebieten bereitstellen.

8 Präsentationsmodellierung

In vielen existierenden Methodologien zur Website-Entwicklung definiert das Navigationsmodell ausschließlich die Navigationsstruktur, während das Präsentationsmodell als zusätzliches Modell festlegt, wie die Navigationsstruktur dem Besucher präsentiert wird. Das Präsentationsmodell stellt in diesem Fall ein schematisches User Interface dar und enthält Angaben über Positionierung, Größe und Aussehen der Komponenten. Beispielsweise arbeitet die Methode OOHDM mit sog. Advanced Data Views, WebML benutzt Stylesheets[13] für die Modellierung der Präsentation, UHML (Hennicker & Koch, 2000) definiert spezielle Stereotypen für Klassen und RMM wird hinsichtlich eines Präsentationsdiagramms erweitert. Zumeist stehen dabei strukturierende Techniken im Vordergrund. Einige Ansätze, wie z. B. UHML, verwenden ein Präsentationsmodell zur strukturellen Organisation der Präsentation, das jedoch keine Angaben über die physikalische Erscheinung hinsichtlich Formatierung, Farben etc. enthält. Die Autoren verweisen für diese Angaben auf die Implementierungs-Phase, deren Inhalte dort allerdings nicht weiter modelliert werden.

Im Rahmen dieser Arbeit wurde mit dem Kompositions- und Navigationsmodell der strukturelle Aufbau der zu erstellenden Website definiert. Dabei wurden bereits Angaben zur relativen Positionierung der einzelnen Komponenten gemacht. Der Grund hierfür ist darin zu sehen, dass eine konzeptionelle Trennung des Navigations- und Präsentationsmodells zwar allgemein als sinnvoll und notwendig erachtet wird, die formalen Modelle jedoch ein Kommunikationsproblem darstellen, da sie nicht nur den Anwender überfordern können, sondern insbesondere die Kommunikation mit nicht-technischen Zielgruppen erschweren und für diese in der Regel ungeeignet sind.

Aus diesem Grund enthält das Navigationsmodell bereits relative Positionsangaben der einzelnen Komponenten, so dass schon frühzeitig eine erste Orientierung bzgl. der strukturellen Gestaltung der Website gegeben wird. Weitere Angaben zum User Interface werden analog zu existierenden Vorgehensweisen in einem separaten Modell, dem Präsentationsmodell, festgelegt. Dieses Modell enthält vor allem Vorschläge zur relativen Größe der einzelnen Komponenten, der Farbwahl sowie der zugehörigen Interaktionskonzepte. Jedoch ist zu beachten, dass das Präsentationsmodell nicht das endgültige Aussehen vorschreibt, vielmehr ist dies wesentlicher Teil der späteren Implementierung bzw. des Customizing[14].

---

[13] Bei sog. Stylesheets handelt es sich um die Zusammenstellung von Formatierungsregeln, die festlegen, wie einzelne Elemente darzustellen sind.

[14] Unter dem Begriff Customizing wird die häufig durch Skriptsprachen realisierte Anpassung von Softwarekomponenten an unterschiedliche Bedürfnisse verstanden.

Eine konzeptionelle Trennung zwischen Navigations- und Präsentationsmodell ist auch in dieser Arbeit zu finden: die strukturelle Konzeption des Navigationsmodells wird durch das Präsentationsmodell in Bezug auf Angaben erweitert, die näher an den Anforderungen der Implementierung liegen, jedoch unabhängig von den eigentlichen Inhalten und Ressourcen sind.

8.1 Präsentationsklassen

Das Präsentationsmodell definiert verschiedene Präsentationsmöglichkeiten für die gleiche Navigationsstruktur für den Fall, dass unterschiedliche Benutzerrollen bzw. personalisierte Darstellungen existieren. Die Modellierung und der Aufbau eines Präsentationsmodells erfolgt in Anlehnung an das Sichtenmodell, d. h. auch hier kann zu jeder Modellklasse eine Präsentationsklasse erzeugt werden, die u. a. folgende Angaben über die der Zugriffsklasse zugeordneten Modellklasse enthält:

- Relative Größe: Die relative Größe einer Komponente bezieht sich auf die Breite und Höhe einer Partition im Verhältnis zur Größe des übergeordneten Containers, der die Partition enthält. Bei der Angabe der relativen Größe handelt es sich um eine Mindestgröße, da benachbarte Partitionen bei entsprechender Definition in den zugehörigen Zugriffsklassen evtl. nicht dargestellt werden. In diesem Fall wird der nicht genutzte Bildbereich auf die darzustellenden Partitionen aufgeteilt.
- Farbwahl: Für jede Modellklasse können Vorgaben sowohl für die Wahl der Hintergrundfarbe, als auch für die Vordergrundfarbe festgelegt werden. Fehlen die Angaben, werden die Werte der übergeordneten Komponente benutzt.
- Interaktionskonzept: Dieser Wert bestimmt das Verhalten bei einer Interaktion durch den Benutzer. Für eine Modellklasse kann festgelegt werden, durch welchen Mechanismus ein Ereignis ausgelöst werden soll. Zum Beispiel kann das Aktivieren eines Links durch Mechanismen wie ‘MouseClick’ oder ‘MouseOver’ ein Ereignis auslösen. Das Interaktionsverhalten wird bereits innerhalb der Navigationsrelationen modelliert, jedoch lässt es sich auf diese Art und Weise auch innerhalb navigationaler Klassen festlegen.

Neben diesen Informationen zur grafischen Gestaltung der einzelnen Komponenten können weitere Angaben im Präsentationsmodell festgelegt werden. Das Präsentationsmodell stellt hierzu eine offene Struktur zur Verfügung, die es erlaubt, beliebige Merkmale der grafischen Gestaltung zu bestimmen. Beispielsweise lassen sich zusätzlich Angaben über die zu verwendende Schriftart, die Schriftgröße etc. festlegen.

Definition 8.1 *Eine* Präsentationsklasse Π *ist ein Tripel* $\Pi = (m, d, \Psi)$*, bestehend aus einer Modellklasse* $m \in \boldsymbol{M}$*, einer Rolle* $d \in \boldsymbol{U}$ *und einer Menge* Ψ *von Darstellungszuordnungen. Eine* Darstellungszuordnung $y \in \Psi$ *ist ein Tupel* $y = (t, w)$*, bestehend aus einem Präsentationstyp* t *und einem Wert* w*.*

8.2 Präsentationsmodell

Die grafische Modellierung einer Präsentationsklasse erfolgt gemäß Abbildung 8-1 analog zur Verwendung von Zugriffsklassen. Präsentationsklassen beziehen sich ebenfalls auf bestimmte Benutzergruppen bzw. einzelne Benutzer. Prinzipiell können jeder Modellklasse beliebig viele Präsentationsklassen zugeordnet werden. Wird eine Präsentationsklasse statt einer Modellklasse einem Container zugewiesen, so beziehen sich die Darstellungseigenschaften auf die unmittelbar im entsprechenden Container enthaltenen Modellklassen. Navigationsrelationen mit einem Container als Ursprung können in Analogie zur Verwendung von Zugriffsklassen (vgl. Kapitel 7.1) auch Präsentationsklassen zugeordnet werden.

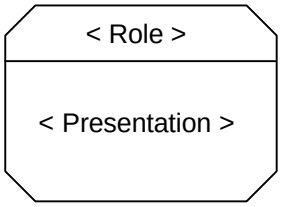


Abbildung 8-1: Grafische Beschreibung einer Präsentationsklasse

Abbildung 8-2 zeigt beispielhaft die grafische Modellierung zweier Präsentationsklassen, die einer Modellklasse bzw. einem Container zugeordnet wurden. Die grafischen Merkmale sowohl des Navigators auf der linken Seite als auch des Informationsbereiches auf der rechten Seite gelten für alle Benutzer. Unterschiede zwischen den beiden Partitionen bestehen in der Größendefinition und der Farbwahl. Während in diesem Beispiel dem Navigationsbereich dreißig Prozent des horizontalen Bereichs der Komponente zugewiesen werden, bleiben dem Informationsbereich die übrigen siebzig Prozent zur Darstellung des Namens und der Adresse eines gewählten Restaurants.

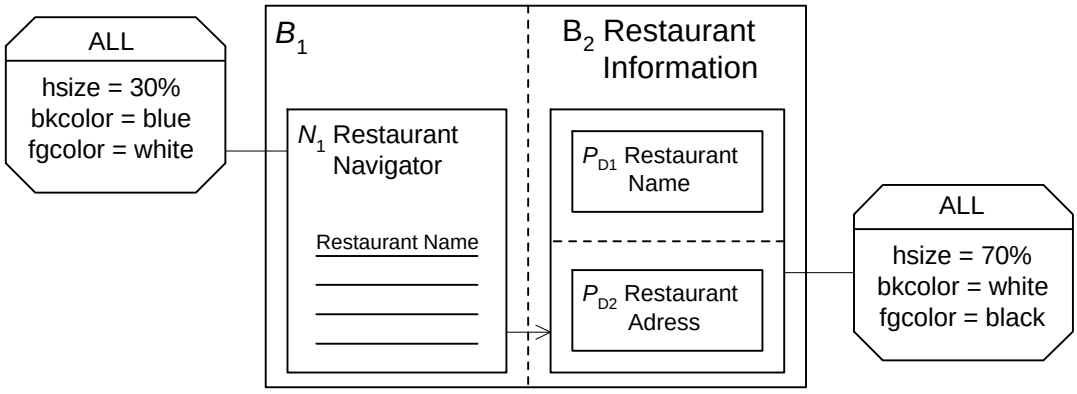


Abbildung 8-2: Verwendung von Präsentationsklassen

9 Anwendung der Vorgehensweise zur Erstellung von Web-Anwendungen

In den vorangegangenen Kapiteln wurde eine Vorgehensweise aufgezeigt, die den Entwicklungsprozess für Websites systematisiert. Der Entwurfsprozess wurde dabei in verschiedene Bereiche unterteilt, deren zugehörige Modelle die unterschiedlichen Anforderungen der Website aufgreifen und darstellen. Im Folgenden wird das Zusammenspiel der einzelnen Bearbeitungsschritte mit ihren zugeordneten Entwurfsmodellen anhand eines Anwendungsbeispiels erläutert, das die Neukonzeption der Web-Präsentation eines Dienstleistungsunternehmens thematisiert. Neben der Modellierung soll gleichzeitig die Funktionsweise der automatisierbaren Generierung der Website berücksichtig werden, als Grundlage für die nachfolgende Diskussion der technischen Unterstützungsmöglichkeiten im Hinblick auf den Entwurf und die Implementierung der Web-Anwendung.

Das Anwendungsbeispiel konzentriert sich auf die partielle Beschreibung der wesentlichen Entwurfsprozesse der Website-Entwicklung mit dem Ziel, die Vorgehensweise und die Anwendung der einzelnen Modelle zu verdeutlichen. Eine vollständige Konzeption und Modellierung des Beispiels ist nicht vorgesehen.

9.1 Analyse

In der Analysephase wird zunächst die Ausrichtung der neuen Website bestimmt. Die Untersuchungen konzentrieren sich auf die Zielsetzungen, die mit der Neugestaltung der Web-Präsenz verfolgt werden, die potentiellen Nutzergruppen, die erreicht werden sollen, sowie die technischen Rahmenbedingungen.

Das als Beispiel dargestellte Dienstleistungsunternehmen bietet Leistungen in den Bereichen Versicherung, Vorsorge und Vermögensbildung an. Die grundlegenden Zielsetzungen lassen sich dementsprechend wie folgt aufschlüsseln:

- Bereitstellung aktueller und ausführlicher Informationen für Kunden,
- Werben von Kunden und Investoren durch professionelle Außendarstellung,
- Online-Verwaltung von Verträgen seitens der Kunden sowie Möglichkeit zur Online-Schadensmeldung zur Entlastung des Kundensupports,
- Werben neuer Mitarbeiter.

Als potentielle Nutzergruppen können anhand dieser Überlegungen einerseits Privat- und Geschäftskunden, andererseits Investoren und Bewerber identifiziert werden. Weitere Evaluationsfaktoren wie technische Rahmenbedingungen, Nutzungssituationen bzgl. der Besucher, Zielerreichbarkeit etc. sollen in diesem Beispiel unbeachtet bleiben.

9.2 Anforderungsspezifikation und Themenmodellierung

Die Themen, die für die Web-Präsentation von Bedeutung sind, bestehen im Wesentlichen aus den vielfältigen Produkten des Versicherers sowie vertiefenden Informationen zu den einzelnen Produkten. Weiterhin sollen Formulare und Dienste, beispielsweise zur Schadensabwicklung, zur Verfügung gestellt werden und dem Kunden die Möglichkeit bieten, einfache Anfragen und Bearbeitungen online durchzuführen.

Von diesen Überlegungen ausgehend, befasst sich der nächste Schritt mit dem Aufbau des konzeptuellen Modells. Abbildung 9-1 zeigt auszugsweise die Kategorisierung der für die Web-Präsenz relevanten Inhalte in Form eines solchen Modells,

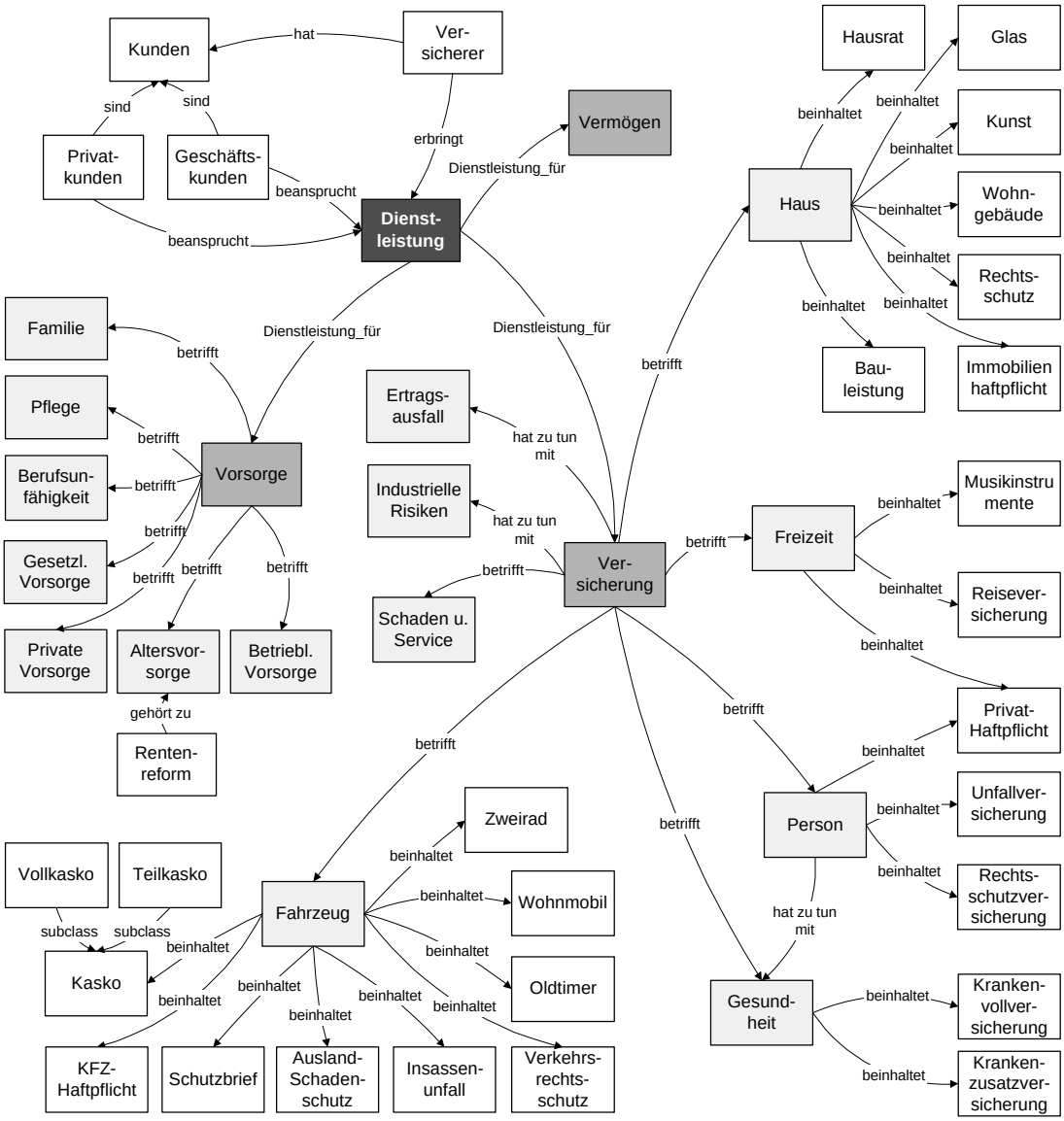


Abbildung 9-1: Konzeptuelles Modell

das in Anlehnung an die Themennavigation des Allianz[15]-Webauftrittes erstellt wurde.

In diesem Anwendungsbeispiel soll ausschließlich der Bereich ‘Versicherung’, der sich auf der obersten Hierarchiestufe (Ebene 0) wieder findet und sich in die Kategorien ‘Fahrzeug’, ‘Gesundheit’, ‘Freizeit’, ‘Haus’ und ‘Person’ (Ebene 1) unterteilt, berücksichtigt werden. Jedes dieser Themengebiete lässt sich in weitere Unterkategorien (Ebene 2) zerlegen, denen verschiedene Informationen, Produkte und Dienste zugeordnet werden können. Das konzeptuelle Modell gibt an dieser Stelle bereits einen ersten Hinweis auf spätere Navigationsmöglichkeiten innerhalb der Web-Anwendung: Über die Hauptkategorien kann aus einer Menge zugeordneter Themenfelder der gewünschte Bereich ausgewählt werden, für den dann die entsprechenden Informationen bereitgestellt werden.

Die Präsentation der Inhalte soll für Themen gleicher Hierarchiestufe jeweils einheitlich erfolgen. Beim Entwurf des konzeptuellen Modells lässt sich dies durch eine gemeinsame Bezeichnung von Themenfeldern, die inhaltlich mehreren Konzepten zugeordnet werden können, erreichen. Abbildung 9-2 zeigt die Rubriken, die neben

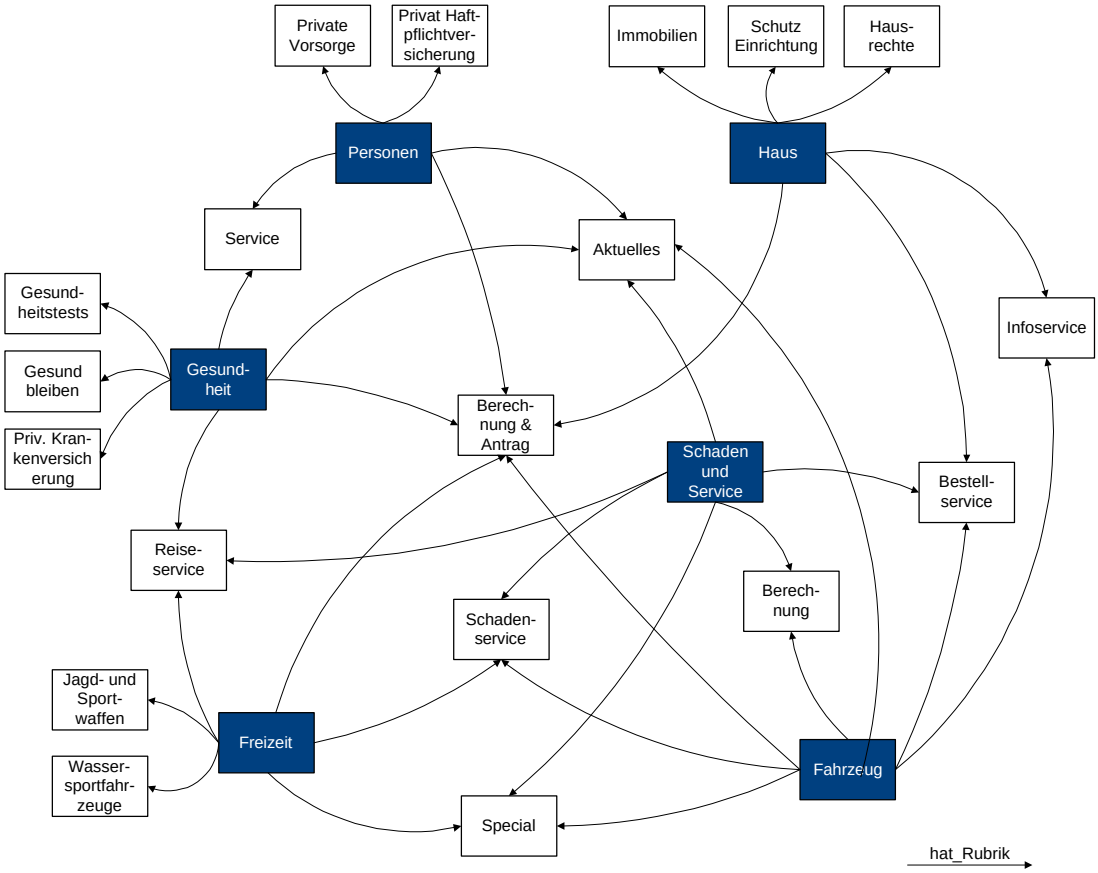


Abbildung 9-2: Themenmodell für Ebene 1

[15] http://www.allianz.de

einer Auswahlmöglichkeit der untergeordneten Kategorien zu den Themen der Ebene 1 dargestellt werden sollen.

Erkennbar ist, dass zu einigen Unterkategorien gleichnamige Rubriken existieren. Da es sich hier jedoch um das Themenmodell handelt, ist an dieser Stelle noch keine Aussage darüber enthalten, ob und welche Inhalte zu den einzelnen Rubriken für verschiedene Subkategorien identisch sind. Beispielsweise können Inhalt und Darstellung bzgl. des Themas ‘Schadenservice’ für alle Subkategorien gleich sein, während die Rubrik ‘Aktuelles’ unterschiedliche Inhalte aufweist.

9.3 Klassifikation und Instanzierung der Inhalte

Entsprechend der in Kapitel 5.1 aufgezeigten Vorgehensweise besteht das weitere Vorgehen in der Klassifizierung der darzustellenden Inhalte auf Grundlage der modellierten Themenstruktur. Hierzu werden Eigenschaften für die einzelnen Konzepte definiert, anhand derer eine Zuordnung von Inhalten auf Basis zugehöriger Metadaten erfolgen kann. Im Detail werden die darzustellenden Inhalte automatisch oder manuell mit Metainformationen ergänzt, die dann ebenfalls automatisiert oder von Hand durch eine Analyse gemeinsamer Eigenschaften den einzelnen Konzepten der Themenstruktur zugeordnet werden können.

Bei der Darstellung der Inhalte, d. h. der Auswahl bestimmter Instanzen eines Konzeptes, wird die entsprechende Instanz bzw. Instanzmenge anhand der Werte ihrer Eigenschaften bestimmt. Die Funktionsweise soll Abbildung 9-3 verdeutlichen, die neben den Konzepten ‘Fahrzeug’, ‘Haus’, ‘Aktuelles’, ‘Berechnung’ etc. die Zuordnung der Instanzen darstellt. So werden beispielsweise in den Instanzen des Konzeptes ‘Bestellservice’ verschiedene Bestellmöglichkeiten zu einem Themengebiet angeboten. Auf welches Themengebiet sich diese Informationen beziehen, d. h. in welche Rubrik diese eingeordnet werden können, wird mit einer Relation zwischen der Instanz und dem entsprechenden Konzept festgelegt. Instanzen des Konzeptes ‘Bestellservice’ besitzen dementsprechend zumindest folgende Eigenschaften (s. Tabelle 9-1):

Name	Typ	Wertebereich
Titel	String	
Inhalt	String	
gehört_zu_Kategorie	Class	parent={Kategorie}

Tabelle 9-1: Eigenschaften des Konzeptes „Aktuelles“

Der Zugriff auf Instanzen der Klasse ‘Bestellservice’ erfolgt durch die Angabe und den Vergleich von Werten, die bestimmten Eigenschaften der Klasse zugewiesen sein müssen. Zum Beispiel lassen sich alle Informationen, die sich auf die Kategorie ‘Fahrzeug’ beziehen, durch die Betrachtung aller Instanzen des Konzeptes ‘Bestellservice’, deren Eigenschaft ‘gehört_zu_Kategorie’ den Wert ‘Fahrzeug’ aufweist, ermitteln.

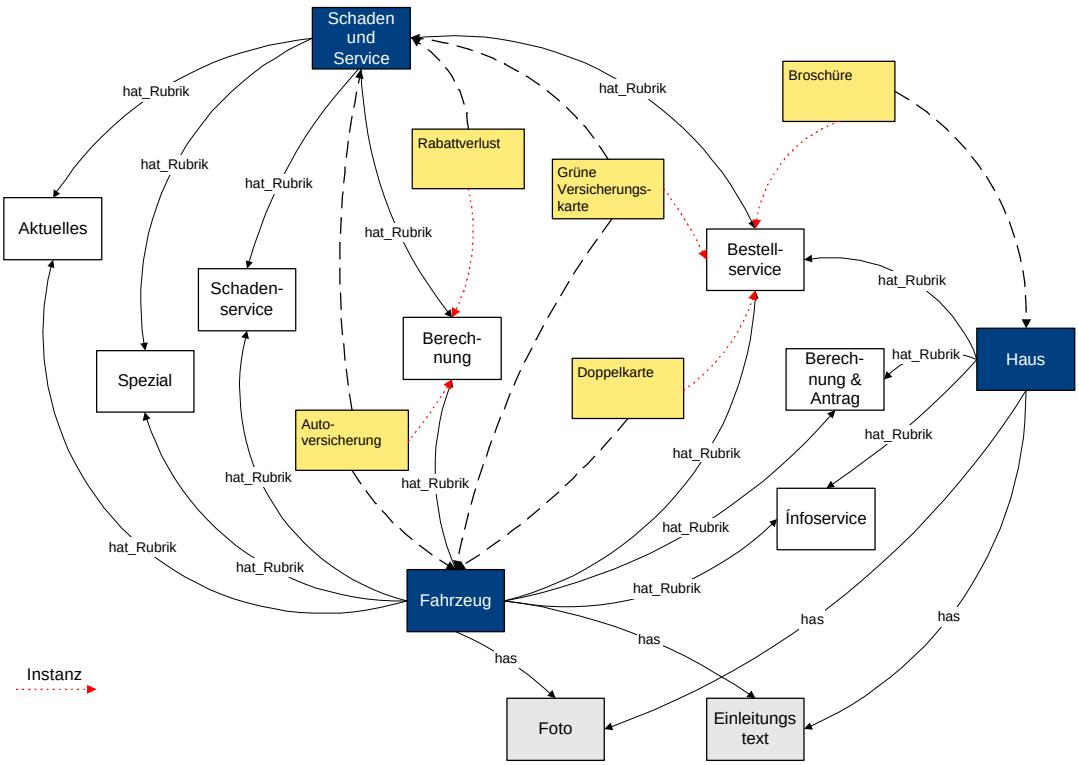


Abbildung 9-3: Zugriff auf Instanzen

9.4 Entwurf des Navigationsmodells

Der nächste Schritt sieht den Entwurf der Komponentenmodelle für die einzelnen Webseiten vor, die das Navigationsmodell bilden. Entsprechend Abbildung 9-4 besteht die Hauptkomponente der Website aus einer Einstiegsseite, die abhängig von der Auswahl des Besuchers im weiteren Navigationsverlauf entweder durch die Hauptseite oder durch eine der Komponenten ‘Investor Relations’, ‘Presse/News’, ‘Karriere’ bzw. ‘International’ ersetzt wird. Die Einstiegsseite soll gemäß der Themenstruktur in Abbildung 9-1 eine Unterscheidung zwischen Privatkunden und Geschäftskunden erlauben und darüber hinaus Navigationsmöglichkeiten zu

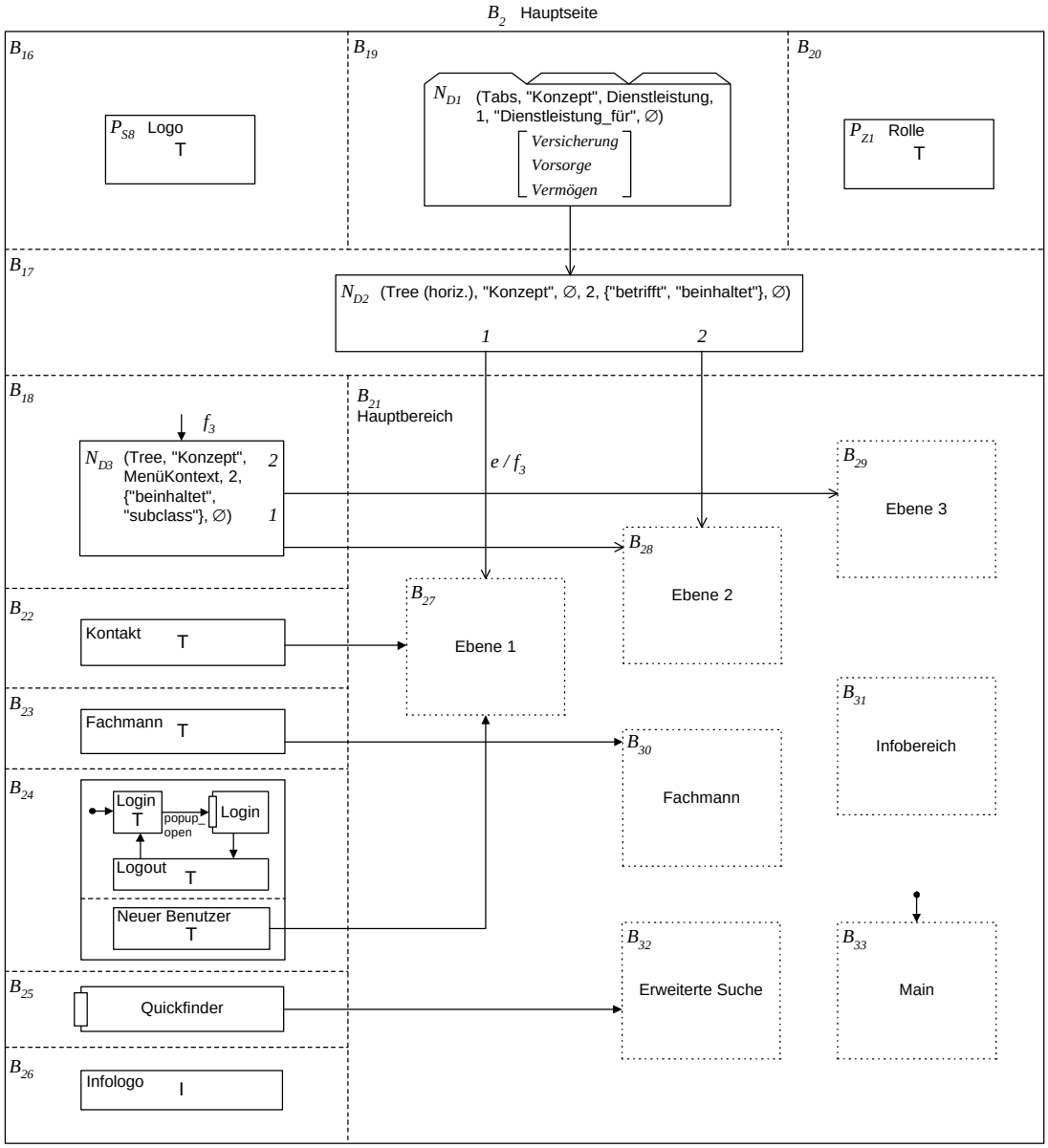


Abbildung 9-4: Komponentenmodell der Einstiegsseite

unternehmensbezogenen Informationen bereitstellen. Sie setzt sich aus einer Kopf- und einer Fußzeile für die Angabe des Firmennamens, sowie einem Hauptbereich zusammen, der wiederum aus den drei horizontal angeordneten Bereichen ‘Privatkunden’, ‘Geschäftskunden’ und ‘Unternehmen’ besteht. Sowohl über die darin enthaltenen Bildkomponenten als auch mittels der Textkomponenten kann auf die assoziierte Webseite, in diesem Fall die Hauptseite, navigiert werden. Diese befindet sich innerhalb des Navigationsmodells in derselben Komponente wie die Einstiegsseite und wird aus diesem Grund bei Aktivierung des entsprechenden Links

ersetzt. Im Vergleich zu den beiden Kundenbereichen beinhaltet der Bereich ‘Unternehmen’ statt der Textkomponente eine Menükomponente, die aus einer statischen Menge von fünf Navigationspunkten besteht. Dieses Menü wird durch eine statische navigationale Klasse des Typs ‘Group’ realisiert, d. h. die Instanzmenge wird zur Modellierungszeit manuell festgelegt. Der Zielbereich dieser navigationalen Klasse ist ebenfalls die mit ‘Hauptseite’ bezeichnete Komponente und ersetzt die Einstiegsseite in gleicher Weise.

Den Rahmen für die Informationsbereitstellung sowohl für Privat- und Geschäftskunden als auch für die Unternehmensdarstellung bildet die Hauptseite (s. Abbildung 9-5). Alle übrigen Informationen, die von der Einstiegsseite aus navigiert werden können, werden in einem Popup-Fenster dargestellt. Hierbei handelt es sich um Inhalte, die über die Menükomponente N_{S1} erreichbar sind.

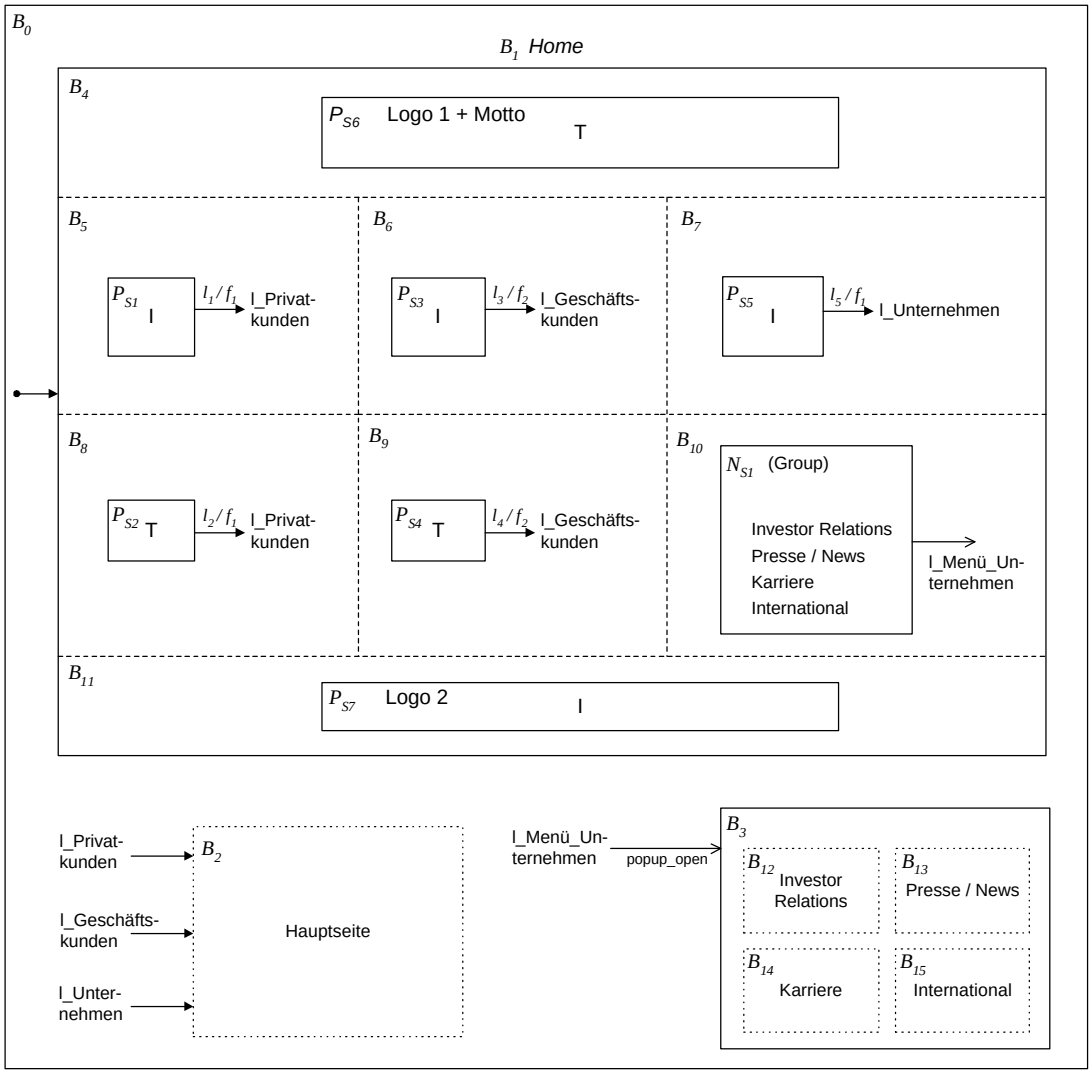


Abbildung 9-5: Komponentenmodell der Hauptseite

Im oberen Bereich der Hauptseite befindet sich ein durch die navigationale Klasse N_{D1} realisiertes Menü vom Typ ‘Tabs’, das die Navigationsauswahl für das darunter liegende horizontale Menü N_{D2} beeinflusst. Der übrige Teil der Hauptseite ist in einen links angeordneten Navigationsbereich und einen Inhaltsbereich partitioniert. Insgesamt enthält die Hauptseite drei Navigationsmenüs, die z. T. voneinander abhängig sind. Das oberste Menü in Form der navigationalen Klasse N_{D1} legt die Auswahl der Menüeinträge des Navigationsmenüs der Klasse N_{D2} entsprechend des konzeptuellen Modells in Abbildung 9-1 fest. Aufgrund der Angabe ‘Tiefe 2’ des Untermenüs werden zu den Hauptmenüeinträgen zusätzlich die unmittelbaren Untermenüeinträge dargestellt. Letztere werden zudem im linken Navigationsbereich durch die Modellklasse N_{D3} bereitgestellt und durch eine zusätzliche Hierarchiestufe erweitert. Interaktionen mit der Komponente N_{D2} führen aufgrund des Broadcasting-Mechanismus zu einer Anpassung des Navigationsmenüs innerhalb der navigationalen Klasse N_{D3}.

Inhalte, die über die zur Verfügung stehenden Navigationsmenüs aufgerufen werden, sollen bei gleicher Hierarchiestufe einen identischen Aufbau besitzen, d. h. sie werden unter Verwendung eines Templates einheitlich strukturiert dargestellt. Die Bezeichnung Template bezieht sich in diesem Zusammenhang auf das Kompositionsmodell einer einzelnen Webseite, dessen Inhalte auf der Basis des konzeptuellen Modells und der Instanzenzuordnung dynamisch nach entworfenem Schema in beliebiger Komplexität bereitgestellt werden.

Für die insgesamt drei vorhandenen Hierarchiestufen werden ebenso viele Ebenen im Zielbereich der navigationalen Klassen für die einzelnen Navigationsmenüs modelliert und jeder Hierarchiestufe einer navigationalen Klasse die Komponente der entsprechenden Ebene zugeordnet. Die Templates lassen sich auch für weitere Inhalte benutzen. So verweisen die den Komponenten ‘Kontakt’ und ‘Neuer Benutzer’ zugehörigen Navigationsrelationen ebenfalls auf die Komponente ‘Ebene 1’.

Anhand dieser Komponente soll die Verwendung von Templates durch navigationale Klassen verdeutlicht werden. Abbildung 9-6 zeigt das Komponentenmodell für ‘Ebene 1’, das sich neben einer Bild- und einer Textkomponente aus den zwei navigationalen Klassen N_{D4} und N_{D5} zusammensetzt. Diese enthalten wiederum maximal vier bzw. drei navigationale Klassen, die nach dem Muster der Modellklassen N_{D6} bzw. N_{D7} aufgebaut sind.

Das Template soll genutzt werden, um die Inhalte der verschiedenen Kategorien ‘Fahrzeug’, ‘Gesundheit’, ‘Haus’ etc., die über die Navigationspunkte des Menüs N_{D2} aufgerufen werden können, einheitlich darzustellen. Die ausgewählte Kategorie wird als Kontextbeschreibung der Hauptkomponente des Templates übergeben, so dass anhand des Kontextes auf Grundlage des konzeptuellen Modells für diesen Ausschnitt (vgl. Abbildung 9-2) die zugeordneten Rubriken ermittelt werden können.

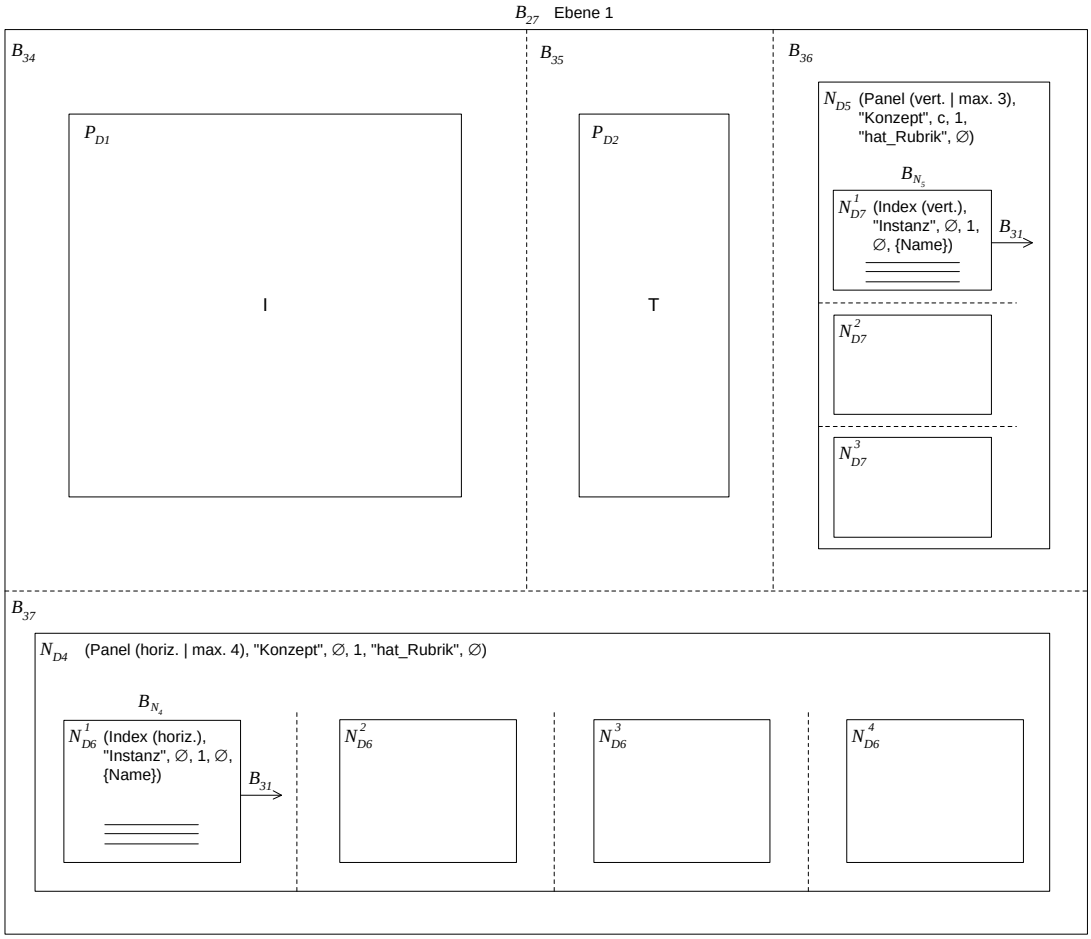


Abbildung 9-6: Komponentenmodell als Template für die Ebene 1

Die einzelnen Rubriken sollen jedoch auf zwei Bereiche aufgeteilt werden: maximal vier Rubriken im unteren und höchstens drei Rubriken im rechten Bereich. Um zu verhindern, dass hierdurch Inhalte doppelt dargestellt werden, wird zur Modellierungszeit die Sortierungsfunktion genutzt, die anhand einer der navigationalen Klasse zugewiesenen Sortierungstabelle eine statische Positionierung der Inhalte ermöglicht. Die dynamische Kontextzuweisung für die navigationalen Klassen N_{D4} und N_{D5} erfordert die Erstellung einer Sortierungstabelle unter Berücksichtigung der verschiedenen Kontexte. Tabelle 9-2 zeigt mögliche Sortierungstabellen, auszugsweise für die beiden Kontexte ‘Fahrzeug’ und ‘Schaden und Service’.

Das Ergebnis ist eine Zuordnung der Rubriken auf die insgesamt sieben untergeordneten navigationalen Klassen in den beiden Rubrikbereichen, so dass anhand der Kontextübergabe die entsprechenden Instanzen zu den einzelnen Rubriken als Liste in den navigationalen Indexklassen aufgeführt werden können.

Konzept	Sortierung $V_{N_{D4}}$ für N_{D4}
Fahrzeug	(Aktuelles, Spezial, Infoservice, Schadenservice)
Schaden und Service	(Aktuelles, Spezial, $\varnothing$, Schadenservice)

Konzept	Sortierung $V_{N_{D5}}$ für N_{D5}
Fahrzeug	(Berechnung, Berechnung und Antrag, Bestellservice)
Schaden und Service	(Berechnung, $\varnothing$, Bestellservice)

Tabelle 9-2: Sortierungstabellen für die navigationalen Klassen N_{D4} und N_{D5}

9.5 Berücksichtigung von Kontextfaktoren

Dem Besucher der Website werden auf der Einstiegsseite die möglichen Rollen ‘Privatkunde’ und ‘Geschäftskunde’ angeboten, für die ein identischer Aufbau der Hauptseite vorgesehen ist. Lediglich die Inhalte sowie die Eigenschaften der Darstellung wie z. B. Hintergrundfarbe etc. sollen variieren. Die verschieden Farben im Erscheinungsbild der Webseite sollen den Unterschied der Besucherrolle widerspiegeln.

Hierzu werden gemäß Abbildung 9-7 für die Hauptseite zwei unterschiedliche Präsentationsklassen definiert, die das grafische Erscheinungsbild entsprechend der gewählten Rolle festlegen.

Abbildung 9-7: Präsentationsklassen zur Unterscheidung der grafischen Präsentation

Der Gültigkeitsbereich der Präsentationsklassen umfasst die Hauptseite und alle in ihr enthaltenen Komponenten, die keine eigene Präsentationsklasse besitzen.

9.6 Generierung und Funktionsweise der Website auf Basis des Modellentwurfs

Das Anwendungsbeispiel weist einen starken Zusammenhang zwischen dem konzeptuellen Modell und der Strukturierung der Inhalte innerhalb der Website anhand des Navigationsmodells auf. Die Verbindung zwischen den beiden Modellen kann genutzt werden, um einerseits den Entwurf des Navigationsmodells durch Bereitstellung entsprechender Modellierungswerkzeuge systemtechnisch zu unterstützen und dadurch zu vereinfachen und andererseits die Generierung der Website automatisiert durchzuführen. Im Folgenden soll dieses Anwendungsbeispiel genutzt werden, um die prinzipielle Funktionsweise des automatisierten Erstellungsprozesses der Website einschließlich des Informationszugriffs zu verdeutlichen.

9.6.1 Umsetzung der Einstiegsseite

Ausgehend vom Navigationsmodell aus Abbildung 9-4 und dem bereits bestehenden konzeptuellen Modell gemäß Abbildung 9-1 beginnt der Aufbau der Website aufgrund des zugewiesenen Startzustandes mit der Einstiegsseite ‘Home’, d. h. mit dem Container B_1. Die Aufteilung der Komponente in drei vertikal angeordnete Bereiche lässt sich in der Darstellung der Webseite mit Hilfe von Frames realisieren, deren Verwendung beispielsweise gegenüber Tabellen den Vorteil hat, dass einzelne Bereiche unabhängig von der gesamten Darstellung einer Webseite aktualisiert werden können. Während der obere und der untere Bereich mit dem Firmenlogo in Form eines Bildes bzw. als Text gefüllt werden, vollzieht sich im mittleren Bereich eine weitere Unterteilung in drei Partitionen. Die darin enthaltenen Container bzw. ihre Teilbereiche stellen ebenfalls statische Informationen mittels Bildern und Texten bereit, die jedoch jeweils mit einem Link unterlegt sind. Dieser ermöglicht die Navigation zum gewünschten Zielbereich unter Mitführung des Kontextes in Form des Zielkonzeptes. Bei der Generierung der Einstiegsseite zur Laufzeit gilt für alle Container die Initialisierung

$$B_i = (\varnothing, \varnothing, \varnothing) \quad \forall\, 0 \le i \le 11 .$$

Den in den Containern enthaltenen primitiven statischen Klassen werden folgende Werte zugeordnet:

P_{S1} = (Image, ‘Privatkunde.gif’), P_{S2} = (Text, ‘Privatkunde’),
P_{S3} = (Image, ‘Geschäftskunde.gif’), P_{S4} = (Text, ‘Geschäftskunde’),
P_{S5} = (Image, ‘Unternehmen.gif’),
P_{S6} = (Image, ‘Logo.gif’), P_{S7} = (Text, ‘XY Versicherung’).

Mit Ausnahme von P_{S6} und P_{S7} wird allen primitiven statischen Klassen jeweils eine Navigationsrelation zugewiesen:

$l_1 = (\varnothing$, Privatkunde, $\varnothing$, P_{S1}, B_2, ‘click’, $\varnothing)$,
$l_2 = (\varnothing$, Privatkunde, $\varnothing$, P_{S2}, B_2, ‘click’, $\varnothing)$,
$l_3 = (\varnothing$, Geschäftskunde, $\varnothing$, P_{S3}, B_2, ‘click’, $\varnothing)$,
$l_4 = (\varnothing$, Geschäftskunde, $\varnothing$, P_{S4}, B_2, ‘click’, $\varnothing)$,
$l_5 = (\varnothing$, Unternehmen, $\varnothing$, P_{S5}, B_2, ‘click’, $\varnothing)$.

Die Navigationsrelationen l_1 bis l_4 lösen bei ihrer Aktivierung aufgrund des Broadcast-Mechanismus ein weiteres Ereignis aus, das für die Festlegung der Besucherrolle in der Zustandstabelle bestimmt ist. Tabelle 9-1 zeigt die entsprechende Zuordnung.

Ereignis	Bezeichner	Wert
f_1	ROLE	Privatkunde
f_2	ROLE	Geschäftskunde

Tabelle 9-3: Definition der Besucherrolle

Zuletzt bleibt die Betrachtung der navigationalen statischen Klasse N_{S1}:

$N_{S1} =$ (Group, $L^{N_{S1}}$, $V_{N_{S1}}$).

Die der navigationalen statischen Klasse zugeordneten Navigationsrelationen sind wie folgt festgelegt:

$l_1^{N_{S1}} = (\varnothing, \varnothing, \varnothing, N_{D1}, B_{12}$, ‘click, popup_open’, ‘Investor Relations’),
$l_2^{N_{S1}} = (\varnothing, \varnothing, \varnothing, N_{D1}, B_{13}$, ‘click, popup_open ‘, ‘Presse / News’),
$l_3^{N_{S1}} = (\varnothing, \varnothing, \varnothing, N_{D1}, B_{14}$, ‘click, popup_open’, ‘Karriere’),
$l_4^{N_{S1}} = (\varnothing, \varnothing, \varnothing, N_{D1}, B_{15}$, ‘click, popup_open’, ‘International’),

so dass für $L^{N_{S1}}$ gilt:

$$L^{N_{S1}} = \left\{ l_i^{N_{S1}} \mid 1 \le i \le 4 \right\}.$$

9.6.2 Umsetzung der Hauptseite

Von der Einstiegsseite aus kann über die Navigationsrelationen l_1 bis l_5 die Hauptseite der Anwendung (vgl. Abbildung 9-5) erreicht werden. Die Inhalte, die auf dieser Seite dargestellt werden sollen, sind vom Kontext der Übergangsrelation der Einstiegsseite zur Hauptseite abhängig. Falls der Besucher die Navigationsrelation

l_1 der Einstiegsseite wählt, gilt für die Container und Partitionen der Hauptseite zunächst folgende Initialisierung:

$B_2 = (P_{S1}$, Privatkunden, $\varnothing)$,
$B_i = (P_{S1}$, Privatkunden, $\varnothing)$ $\quad \forall 16 \le i \le 26$.

Die drei Partitionen des innerhalb von B_{14} liegenden Containers enthalten die Komponenten

$P_{S8} =$ (Image, 'Logo.gif'),
$P_{Z1} =$ (ROLE),
$N_{D1} =$ (Tabs, 'Konzept', Privatkunden, Dienstleistung, 1, {'Dienstleistung_für'}, $\varnothing$, $\{V_{N_{D1}}\}, \{B_{17}\})$.

Mit Hilfe der primitiven Zustandsklasse P_{Z1} wird die vom Besucher auf der Einstiegsseite getroffene Entscheidung bzgl. seiner Rolle auf der Hauptseite dargestellt. Alternativ ließe sich diese für den Fall, dass die Rolle des Besuchers nicht explizit als Zustand gespeichert wird, anhand des Kontextes des übergeordneten Containers ermitteln:

$P_{Z1} = (B_2)$.

Zu beachten ist jedoch, dass der Kontext eines Containers nicht unbedingt eindeutig vorhersehbar ist. Falls z. B. neben der Einstiegsseite eine weitere Seite mit einer Navigationsrelation zur Hauptseite existiert, ist die Übergabe eines anderen Kontextes denkbar.

Die navigationale Klasse N_{D1} stellt Karteireiter für die Konzepte, die über die Eigenschaft 'Dienstleistung_für' mit dem Konzept 'Dienstleistung' assoziiert sind, zur Verfügung. Da der übergeordnete Container keine Angabe zur Ursprungsinstanz enthält, wird stattdessen das Ursprungskonzept 'Privatkunden' des mitgelieferten Kontextes berücksichtigt.

Jede Interaktion mit dieser Modellklasse führt zu einer Änderung der navigationalen Klasse N_{D2}, die im Zielbereich B_{17} liegt und wie folgt initialisiert wird:

$N_{D2} =$ (Tree (horiz.), 'Konzept', Privatkunden, c, 2, {'betrifft', 'beinhaltet'}, $\varnothing$, $\{V_{N_{D2}}\}, \{B_{27}, B_{28}\})$ mit $c \in \boldsymbol{C}$.

Da im Kompositionsmodell der Hauptseite der navigationalen Klasse N_{D2} kein Konzept spezifiziert wurde, erfolgt dessen Bestimmung in Abhängigkeit des von der Modellklasse N_{D1} übergebenen Kontextes. Die Initialisierung von N_{D2} ist dementsprechend von dem vom Besucher ausgewählten Konzept c der navigationalen Klasse N_{D1} abhängig. Zur Laufzeit werden von der navigationalen Klasse zwei

Hierarchieebenen erzeugt. Auf der ersten Ebene befinden sich alle Konzepte $c_1,...,c_n \in \boldsymbol{C}$ die mit c über die Eigenschaft ‘betrifft’ assoziiert sind, während die zweite Ebene alle weiteren Konzepte auflistet, die mit mindestens einem der Konzepte $c_1,...,c_n$ über die Eigenschaft ‘beinhaltet’ in Verbindung stehen. Je nachdem, welcher Ebene ein zur Laufzeit vom Besucher ausgewähltes Konzept zugeordnet ist, entspricht der Zielbereich entweder dem Container B_{27} oder B_{28}. Gleichzeitig wird das gewählte Konzept über den Broadcast-Mechanismus mit der Bezeichnung ‘MenüKontext’ in die Zustandstabelle (s. Tabelle 9-4) geschrieben und kann auf diese Weise von der navigationalen Klasse N_{D3} ausgelesen werden.

Ereignis	Bezeichner	Wert
f_1	ROLE	Privatkunde
f_2	ROLE	Geschäftskunde
f_3	MenüKontext	$\varnothing$

Tabelle 9-4: Erweiterte Zustandstabelle

Die navigationale Klasse N_{D3} ermittelt die Menge der darzustellenden Elemente anhand des dem Bezeichner ‘MenüKontext’ zugewiesenen Wertes innerhalb der Zustandstabelle, der von der Modellklasse N_{D2} abhängig ist. Sie setzt sich demnach folgendermaßen zusammen:

N_{D3} = (Index, ‘Konzept’, $\varnothing$, MenüKontext, 2, {‘beinhaltet’, ‘subclass’}, $\varnothing$, $\{V_{N_{D3}}\}$, $\{B_{28}, B_{29}\}$).

Die zweite Hierarchiestufe listet alle Konzepte auf, die eine Unterklasse zu den Konzepten der ersten Hierarchiestufe bilden. Auch hier existieren für die beiden unterschiedlichen Ebenen zwei voneinander verschiedene Zielbereiche.

Auf der generierten Webseite stellt sich der Zusammenhang zwischen den durch navigationale Klassen realisierten Navigationsmenüs wie folgt dar: Je nachdem, welche Kategorie in der Komponente N_{D1} gewählt wird, aktualisiert sich ausschließlich der Inhalt der navigationalen Klasse N_{D2}. Eine Auswahl des Besuchers von Konzepten der ersten Hierarchieebene führt zum einen zu einer erneuten Darstellung der untergeordneten Konzepte innerhalb der Komponente N_{D3} und zum anderen zu einer Aktualisierung des Hauptbereiches entsprechend des in B_{27} definierten Kompositionsmodells. Werden innerhalb der Modellklasse N_{D2} Konzepte der zweiten Hierarchiestufe gewählt, ändert sich die Darstellung im Hauptbereich gemäß der Vorgaben aus B_{28}. Prinzipiell ist es möglich, auch in diesem Fall die Komponente N_{D3} entsprechend neu aufzubauen, was sich im Navigationsmodell

durch einen Broadcast für Navigationsrelationen der Tiefe zwei ausdrücken ließe. Allerdings würde die navigationale Klasse N_{D3} lediglich benachbarte Elemente des gewählten Konzeptes darstellen, d. h. eine andere Hierarchiestufe betrachten als bei der Auswahl von Konzepten der ersten Ebene. Um dies zu umgehen, wäre eine spezielle Implementierung der Komponente N_{D3} notwendig.

Zuletzt bleibt noch die Frage, wie die Unterscheidbarkeit zwischen den Besucherrollen ‘Privatkunden’ und ‘Geschäftskunden’ beim Aufbau der Menüs anhand der entsprechenden navigationalen Klassen erreicht werden kann. Nach dem zugrunde liegenden konzeptuellen Modell werden beispielsweise die Konzepte ‘Fahrzeug’, ‘Person’, ‘Ertragsausfall’ etc. dem Konzept ‘Versicherung’ ohne eine Rollenbetrachtung zugeordnet, d. h. die Menüinhalte wären für beide Rollen identisch. Zur Lösung dieser Problemstellung sind drei Ansätze denkbar:

1. Es werden zusätzliche Relationen zwischen den Konzepten in das konzeptuelle Modell eingefügt, beispielsweise zwischen ‘Geschäftskunden’ und ‘Ertragsausfall’ bzw. ‘Industrielle Risiken’. In der Regel werden derartige Zusammenhänge bereits bei der Erstellung des Modells berücksichtigt und dementsprechend modelliert. Wird bei der Generierung der Webseite keine Ursprungsinstanz übergeben, d. h. die Navigation erfolgt ausschließlich auf Konzeptebene, kann dem mitgelieferten Kontext das Ausgangskonzept entnommen und zur Einschränkung der zu ermittelnden Instanz- bzw. Konzeptmenge verwendet werden. Der Nachteil dieser Vorgehensweise liegt vor allem darin, dass zusätzlich zum Zielkonzept nur genau ein Ausgangskonzept berücksichtigt werden kann, so dass die Zielmenge lediglich anhand zweier Eigenschaften bestimmt wird. Außerdem ist aufgrund der möglichen Komplexität von Websites, beispielsweise hervorgerufen durch eine Vielzahl von Querverlinkungen, der übergebene Kontext nicht zwingend vorhersehbar, so dass dieser Ansatz nur für einzelne Webseiten mit begrenzter Komplexität verwendbar ist.
2. Innerhalb des konzeptuellen Modells wird eine getrennte Modellierung der Themenstrukturen für Geschäftskunden bzw. Privatkunden vorgenommen. Dieser Ansatz widerspricht jedoch dem Ziel einer Ontologie und soll nur der Vollständigkeit halber erwähnt werden.
3. Um komplexere Beziehung und Abhängigkeiten darzustellen, werden Zugriffsklassen verwendet, die jeder beliebigen Modellklasse zugeordnet werden können. Die Bestimmung der Instanz- bzw. Konzeptmenge erfolgt in Abhängigkeit der in der Zugriffsklasse festgeschriebenen Bedingung.

9.6.3 Templatebasierte Informationsdarstellung

Die durch die navigationalen Klassen realisierten Menüs ermöglichen dem Besucher eine Auswahl von Themen, die sich auf drei Hierarchieebenen verteilen. Die

Inhalte zu diesen Themen sollen im Hauptbereich des Hauptfensters dargestellt werden, wobei Inhalte gleicher Ebene sich in einer einheitlichen Struktur und Darstellungsform präsentieren sollen. Am Beispiel der obersten Ebene (Ebene 1) soll unter Verwendung des zugehörigen Templates (s. Abbildung 9-6) aus Kapitel 9.4 die dynamische Informationsbereitstellung nach einer einheitlichen Strukturierungsvorgabe verdeutlich werden.

Der durch die Menüklasse N_{D2} an den Container B_{27} übergebene Kontext bezieht sich auf die Hauptkonzepte ‘Fahrzeug’, ‘Haus’, etc. des konzeptuellen Modells aus Abbildung 9-2. Die in B_{27} enthaltenen Subcontainer B_{34} bis B_{37} werden mit diesem Kontext initialisiert. Falls der Besucher beispielsweise den Menüpunkt ‘Fahrzeug’ gewählt hat, sieht die Initialisierung der Container wie folgt aus:

$B_{27} = B_{34} = B_{35} = B_{36} = B_{37} = (N_{D2}$, Fahrzeug, $\varnothing)$.

Aufgabe der navigationalen Klassen N_{D4} und N_{D5} ist die parallele Darstellung der dem übergebenen Konzept zugeordneten Rubriken entsprechend der Festlegung im konzeptuellen Modell. Nach obigem Beispiel ergibt sich für die beiden Modellklassen die Initialisierung

$N_{D4} =$ (Panel (horiz. | max. 4), ‘Konzept’, $\varnothing$, Fahrzeug, 1, ‘hat_Rubrik’, $\varnothing$, $\{V_{N_{D4}}\}$, $\varnothing$),
$N_{D5} =$ (Panel (vert. | max. 3), ‘Konzept’, $\varnothing$, Fahrzeug, 1, ‘hat_Rubrik’, $\varnothing$, $\{V_{N_{D5}}\}$, $\varnothing$),

die aufgrund der Sortierungstabelle (vgl. Tabelle 9-2) zu einer Übergabe der Konzepte ‘Aktuelles’, ‘Spezial’, ‘Infoservice’ und ‘Schadenservice’ an die in N_{D4} liegenden navigationalen Indexklassen sowie der Konzepte ‘Berechnung’, ‘Berechnung und Antrag’ und ‘Bestellservice’ an die in N_{D5} enthaltenen navigationalen Indexklassen führt.

Anhand der Instanzenzuordnung gemäß Abbildung 9-3 werden in den parallel angeordneten navigationalen Indexklassen die Instanzen zu den einzelnen Konzepten als Referenzen dargestellt. Die innerhalb der Panelklasse N_{D5} an dritter Stelle positionierte navigationale Indexklasse ist in diesem Beispiel für die Auflistung der zum Konzept ‘Bestellservice’ zugeordneten Instanzen zuständig. Allerdings soll nicht die gesamte Instanzmenge dieses Konzeptes aufgelistet werden, sondern ausschließlich die Instanzen, die im Zusammenhang mit dem Konzept ‘Fahrzeug’ stehen. Da dem übergeordneten Container B_{37} keine Ursprungsinstanz übergeben wurde, wird stattdessen das mitgelieferte Ursprungskonzept zur Auswahl der Instanzmenge herangezogen. Für die entsprechende navigationale Indexklasse N^3_{D7} innerhalb von N_{D5} ergeben sich somit folgende Parameter:

N^3_{D7} = (Index (horiz.), ‘Instanz’, Fahrzeug, Schaden und Service, 1, $\varnothing$, {Titel}, $\{V_{N^3_{D7}}\}$, B_{31}).

Sollen weitere Bedingungen für die Bestimmung der Instanzmenge berücksichtigt werden (z. B. Erstellungsdatum der Inhalte etc.), kann dies über die Definition von Zugriffsklassen erfolgen.

Zuletzt bleibt die Betrachtung der beiden primitiven dynamischen Klassen P_{D1} und P_{D2}, die zu jedem Konzept, das von der Menüklasse N_{D2} übergeben wird, das zugehörige Bild bzw. den Einleitungstext darstellen. Entsprechend des obigen Beispiels gilt bei Auswahl des Konzeptes ‘Fahrzeug’:

P_{D1} = (Image, Fahrzeug, Foto, ‘has’),
P_{D2} = (String, Fahrzeug, Einleitungstext, ‘has’).

10 Integration der Vorgehensweise und der Modelle in eine Web-Engineering-Umgebung

Die in dieser Arbeit vorgestellte Methodologie dient dem strukturierten und schrittweisen Entwurf einer Web-Anwendung unter Zuhilfenahme verschiedener Modelle für die unterschiedlichen Entwurfsbereiche. Die grafischen Modelle sind einerseits für die Unterstützung des Website-Entwicklers in der Entwurfsphase geeignet und dienen darüber hinaus zur Kommunikation mit anderen, am Entwicklungsprozess beteiligten Mitarbeitern, bzw. weiteren Personen, wie beispielsweise dem Auftraggeber. Durch den ihnen zugrunde liegenden Formalismus lassen sie sich andererseits auch für die systemtechnische Unterstützung sowohl des Entwurfsprozesses als auch für die Bereitstellung der Web-Anwendung nutzen. Die formale Beschreibung der einzelnen Modelle ermöglicht den schrittweisen Aufbau der Web-Anwendung. Ausgehend vom konzeptuellen Modell können die darin enthaltenen navigierbaren Beziehungen zwischen den Konzepten für den Aufbau des Navigationsmodells vorgeschlagen werden und als Grundgerüst für den Entwurf des Navigationsmodells dienen. Anhand der weiteren Modelle, also dem Sichtenmodell und dem Präsentationsmodell, wird das erstellte Navigationsmodell weiter spezifiziert. Dabei werden die einzelnen, in den Kompositionsmodellen des Navigationsmodells enthaltenen Modellklassen, um Zugriffsbedingungen und Darstellungseigenschaften erweitert. Diese Faktoren stützen sich jeweils auf die bereits in den Kompositionsmodellen enthaltenen Zugriffsdefinitionen. Die Erstellung der diesen Faktoren zugeordneten Modelle kann daher systemunterstützt durch die Weiterbearbeitung des Navigationsmodells auf der Ebene der Sichten- bzw. Präsentationsmodellierung erfolgen.

Entscheidend für den Einsatz grafischer Modelle zur Website-Entwicklung ist die Unterstützung des zeichnerischen Modellierens durch geeignete Werkzeuge, die möglichst alle Entwurfsbereiche in einem integrierten Ansatz abdecken. Dabei gilt es, die Anwendbarkeit auch für nicht-technische Zielgruppen, die in der Regel maßgeblich am Design der Website beteiligt sind, sicherzustellen. Um den Modellierungsprozess zu vereinfachen, basiert der Entwurf des Navigationsmodells auf der Entwicklung bzw. Bereitstellung eines konzeptuellen Modells, das wesentliche Begriffe und einfache thematische Zusammenhänge kennzeichnet. Die Nutzung einer Ontologie als Ausgangspunkt für die Entwicklung einer Web-Anwendung ist nicht zwingend erforderlich, da die grafische Modellierung den rein zeichnerischen Entwurf und die Spezifikation der Web-Anwendung erlaubt. Die Erstellung der weiteren Modelle sowie die Umsetzung und Implementierung der gesamten Anwendung sind diesem Fall jedoch nur nahezu ausschließlich manuell möglich. Nicht nur aus diesem Grund stellt sich die Frage nach dem Formalisierungsgrad beim Entwurf des konzeptuellen Modells. Da ohne die Kategorisierung und Strukturierung

der für die Web-Präsentation relevanten Themen eine erfolgreiche Website-Entwicklung nicht durchführbar ist, ist eine Abwägung zwischen dem Aufwand der Ontologieentwicklung und dem Nutzungspotential hilfreich.

Vor diesem Hintergrund ist ein wesentliches Ziel dieser Arbeit die Konzeption und Gestaltung der Metamodelle im Hinblick auf eine systemtechnische Weiterverwendbarkeit der vom Benutzer entworfenen Modelle, um deren Nutzungsmöglichkeit innerhalb des gesamten Web-Engineering-Prozesses weitestgehend auszuschöpfen. Dies gilt vor allem in Bezug auf die Überführung der entwickelten Modelle in eine Laufzeitumgebung, da hier ein hoher Automatisierungsgrad erreicht werden kann. Von Bedeutung ist in diesem Zusammenhang die Generierung einer vom Besucher der Website angefragten einzelnen Webseite anhand der formalisierten Modelle unter Berücksichtigung der verschiedenen Kontextfaktoren.

Das systemtechnische Unterstützungspotential der Vorgehensweise sowohl zur Modellierungszeit als auch zur Laufzeit soll in diesem Kapitel thematisiert werden. Hierzu werden die Grundzüge einer Referenzarchitektur aufgezeigt, die den Anforderungen einer weitestgehend automatisierten Generierung der Website auf der Grundlage der in den vorangegangenen Kapiteln vorgestellten Metamodelle Rechnung trägt. Die Architektur umfasst die Unterstützung der Modellentwicklung in allen Entwurfsbereichen und den Zugriff auf die erstellten Modelle für die Generierung der Web-Anwendung zur Laufzeit. Die Möglichkeiten der Systemunterstützung und Automatisierung sollen dazu zunächst im Einzelnen für die entsprechenden Entwurfsebenen betrachtet werden.

10.1 Systemunterstützung bei der Entwicklung und Bereitstellung des konzeptuellen Modells

Die Erstellung des konzeptuellen Modells vollzieht sich in zwei Phasen. Im ersten Schritt wird die Themenstruktur der Ontologie entworfen, die lediglich Konzepte und Relationen enthält, während der zweite Schritt die Klassifizierung der Informationsbestände anhand der Themenstruktur umfasst. Ein bereits existierendes konzeptuelles Modell kann nachträglich sowohl durch weitere Konzepte und Relationen als auch durch Zuordnungen von Informationsressourcen erweitert werden.

Für den Aufbau der Themenstruktur existieren verschiedene Vorgehensweisen, die mit unterschiedlichen systemtechnischen Unterstützungsmöglichkeiten ausgestattet sind. Mit Hilfe von Werkzeugen zur Textanalyse können existierende Dokumente analysiert und ein erstes Begriffssystem aufgebaut werden. Dieses Begriffssystem, bestehend aus Kernthemen, die aus der Analyse hervorgegangen sind und Beziehungen zwischen diesen, lässt sich jedoch auch manuell erzeugen. Auch hierzu existieren umfangreiche Werkzeuge, die den Aufbau der Ontologie unterstützen. Eine dritte Möglichkeit liegt darin, bereits existierende Ontologien, sowie über das

Web bereitgestellte domänenspezifische Ontologien zumindest teilweise zu verwenden und ggf. den eigenen Bedürfnissen anzupassen.

Die anschließende Klassifikation und entsprechende Zuordnung der Dokumentinhalte zu den Konzepten der Themenstruktur lässt sich ebenfalls manuell oder unter Zuhilfenahme entsprechender Werkzeuge durchführen. Bei diesem Schritt werden zum Aufbau der Ressourcenontologie für die analysierten Dokumentinhalte Individuen bzw. Instanzen erstellt und den Konzepten der Themenstruktur zugeordnet. Von hoher Bedeutung ist in dieser Hinsicht die getrennte Betrachtung der eigentlichen Dokumentinhalte, ihrer Struktur und ihrer Darstellung. Dieses Prinzip, das wesentliches Merkmal heutiger Content-Management-Systeme ist, ermöglicht eine geräte- und medienunabhängige Präsentation der Dokumentinhalte, wie sie zur webbasierten Darstellung benötigt wird.

Über die Analyse von Dokumentbeständen hinaus, lassen sich Aufgaben und Arbeitsabläufe in einem Unternehmen identifizieren, um daraus Grundzüge für eine webbasierte Abbildung und Bearbeitung von Benutzeraufgaben zu erzielen. Die Untersuchungen resultieren in einer semantischen Beschreibung der aufgabenbezogenen Prozesse mit entsprechenden Prozessmodellen. Die Ressourcenontologie wird entsprechend Abbildung 10-1 durch eine Prozessontologie ergänzt, dessen Prozessbeschreibung aus der Modellierung der Arbeitsabläufe hervorgeht und sich auf Konzepte der Ressourcenontologie bezieht.

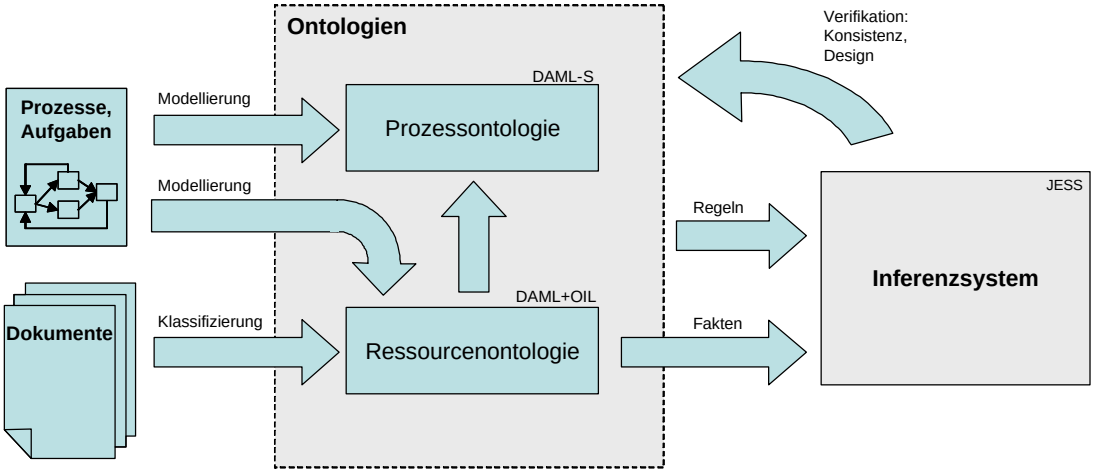


Abbildung 10-1: Systemunterstützung beim Aufbau des konzeptuellen Modells

Mit diesen Vorgaben werden die notwendigen Voraussetzungen erfüllt, die für die Bereitstellung einer aufgabenbezogenen Präsentation und Navigation innerhalb der

Web-Anwendung erforderlich sind. Anhand des Prozessmodells können Arbeitsschritte im Einzelnen betrachtet und thematische Zusammenhänge mit Hilfe der Ressourcenontologie identifiziert werden. Dies ermöglicht die automatisierte Generierung von Webseiten, die sich auf die einzelnen Arbeitsschritte beziehen und eine dem Prozessmodell angepasste Navigation zur Verfügung stellen. Vorschläge zur Modellierung und Überführung von Prozessen in ontologische Beschreibungen stellen u. a. BPEL und DAML-S dar.

Sowohl die Ressourcen- als auch die Prozessontologie erlauben die Adressierung der eigentlichen Inhalte anhand einer thematischen Strukturierung. Durch die Anbindung eines Inferenzsystems kann diese Strukturierung hinsichtlich der Konsistenz des konzeptuellen Modells und des Designs überprüft werden (s. Abbildung 10-1). Hierzu ist die Überführung der Konzepte und Relationen der Ressourcenontologie in Fakten und Regeln des Inferenzsystems erforderlich. Vorgehensweisen hierzu wurden bereits in Abschnitt 5.3.2 erwähnt. Neben der Verifikation des konzeptuellen Modells können zusätzlich Objekte anhand ihrer Eigenschaften automatisch klassifiziert werden. Um dies zu erreichen, werden die Eigenschaften des Objektes mit den definierten Eigenschaften existierender Objekte verglichen, auf dessen Grundlage das Objekt dem entsprechenden Konzept zugeordnet werden kann.

10.2 Systemunterstützung bei der Entwicklung des Navigations-, Sichten- und Präsentationsmodells

Für den Entwurf des Navigationsmodells ermöglicht das konzeptuelle Modell dem Website-Entwickler eine erste Orientierung. Bei entsprechender Gestaltung der Ontologie können die darin enthaltenen Konzepte und Relationen als grundlegende Strukturierung der Informationsdarstellung für den Aufbau der einzelnen Webseiten herangezogen werden. Die formale Beschreibung der Modellklassen des Navigations- bzw. Kompositionsmodells erlaubt während der Modellierung die systemunterstützte Generierung einer Auswahl möglicher Konzepte und Eigenschaften als Parameter für jede Modellklasse. Der Benutzer kann somit den bereits strukturierten Informationsbestand für den weiteren Entwurf der Web-Anwendung nutzen.

Auf dieser Grundlage können durch Softwareunterstützung anhand des grafischen Aufbaus des Navigationsmodells und den zugeordneten Sichten- bzw. Präsentationsklassen, Zugriffsmechanismen auf den Informationsbestand automatisiert in den formalen Repräsentationen der entsprechenden Modelle festgelegt werden. Der somit bereits zur Modellierungszeit bestehende Zusammenhang zwischen Modellentwicklung und der Adressierung der im Modell festgelegten Inhalte, stellt die automatisierte Weiterverarbeitung der Modelle in den Vordergrund. Die mit den Zugriffsinformationen angereicherten Modelle erlauben zur Laufzeit den automati-

schen Zugriff auf den Informationsbestand und ermöglichen die dynamische Generierung der entsprechenden Webseite.

Ausgangspunkt für die automatische Weiterverarbeitung der zeichnerisch erstellten Modelle mit ihrer zugehörigen formalen Beschreibung ist die Generierung einer maschinenlesbaren Ausgabe der parametrisierten Komponenten des Navigationsmodells auf der Basis des Austauschformates XML. Den in diesem Format spezifizierten Modellklassen werden die ebenfalls durch XML beschriebenen Sichten- bzw. Präsentationsklassen zugeordnet. Die Transformation in das Austauschformat lässt sich anhand der im Anhang C aufgezeigten Dokument-Typ-Definition[16] (DTD) durchführen. Die Aufgabe der softwaretechnischen Unterstützung liegt hier vor allem in der Übersetzung der grafischen Modelle anhand ihrer formalen Beschreibung in eine XML-basierte Ausgabe. Diese definiert Struktur, Inhalt und Darstellung der gesamten Web-Anwendung sowie den Zugriff auf die entsprechenden Inhalte und stellt die Grundlage für die Generierung der Website zur Laufzeit dar.

Eine Evaluation der erstellten Modelle ist hier ebenfalls möglich. Die Modellierungsumgebung kann bereits bei der Erstellung der Komponentenmodelle die Erreichbarkeit der in den Modellklassen definierten Inhalte verifizieren und die Übergabe des Kontextes für jede Komponente analysieren. Im Hinblick auf die Einbeziehung der Kontextrückverfolgung (vgl. Kapitel 6.1.4) lassen sich zur Modellierungszeit konzeptionelle Fehler feststellen.

10.3 Referenzarchitektur

Die in den vorangegangen Abschnitten aufgezeigten systemtechnischen Unterstützungsmöglichkeiten beziehen sich auf den Modellierungsprozess, der vom Benutzer durchgeführt wird und dessen Ergebnis eine XML-basierte Spezifikation des strukturellen Aufbaus sowie der rollenbezogenen bzw. personalisierten Darstellung ist. Dies schließt die Definition der Zugriffe auf den Informationsbestand mit ein. Die Adressierung der Inhalte erfolgt entweder unmittelbar durch direkten Zugriff auf die Ontologie oder über eine Anfrage an das Inferenzsystem. Dieses kann vor allem zur Beantwortung komplexerer Fragestellungen, die in den Zugriffsklassen modelliert werden können und zudem aus der Betrachtung verschiedener Kontextfaktoren resultieren, herangezogen werden. Weiterhin ermöglicht es im Zusammenhang mit einer aufgabenbezogenen Navigation die Vorschlagsgenerierung bzgl. des als nächstes auszuführenden Arbeitsschrittes. Voraussetzung ist jedoch, dass alle in Frage kommenden Prozessschritte als Regeln im Inferenzsystem vorliegen und zusätzlich Start- und Zielzustand als Fakten definiert wurden. Je nachdem, ob ein Inferenzsystem für den Zugriff auf die in der Ressourcenontologie referenzierten Inhalte in die Gesamtarchitektur einbezogen werden soll, müssen die Zugriffsmus-

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

ter der Zugriffsklasse entsprechend definiert werden (vgl. Abschnitt 7.2.1). Die Art der Darstellung der Inhalte wird schließlich durch das Präsentationsmodell bestimmt.

Mit Hilfe einer modellgesteuerten Laufzeitumgebung wird anhand des XML-basierten Navigations- bzw. Kompositionsmodells, die Grundstruktur einer Webseite generiert. Die darin festgelegten Inhalte werden mit Hilfe der ebenfalls XML-basierten Sichtenbeschreibungen und den zugehörigen Zugriffsdefinitionen ermittelt und unter Berücksichtigung der von der Laufzeitumgebung erfassten Kontextfaktoren entsprechend der Präsentationsbeschreibung visualisiert. Zu beachten ist hierbei, dass die Berücksichtigung der Kontextfaktoren deren Abbildung und Aktualisierung auf Fakten des Inferenzsystems voraussetzt. Wird auf ein Inferenzsystem verzichtet, muss die grundlegende Logik zur Überprüfung der darzustellenden Sichten von der Laufzeitumgebung selbst getragen werden.

Aufgabe der Laufzeitumgebung ist des Weiteren die Bereitstellung einer Navigationsmöglichkeit entsprechend der navigationalen Klassen. Die Implementierungen zu den verschiedenen Darstellungsformen können in ihrer Funktionalität und Darstellung erweitert werden, um anhand der Grundformen spezielle Navigationstypen zu realisieren. Durch die Unbeschränktheit der Darstellungsarten navigationaler Klassen und der einheitlichen formalen Beschreibung können beliebige neue Klassen konzipiert werden, beispielsweise zur freien ontologiebasierten Navigation über die Konzepte und Relationen des konzeptuellen Modells oder zur automatischen Auswahl der passenden Darstellungsform in Abhängigkeit der Kardinalität der Instanzmenge und der verfügbaren Darstellungsfläche.

Falls zusätzlich zur informationsbasierten Navigation durch entsprechende Modellierung eine aufgabenbezogene Darstellung und Navigation bereitgestellt wird, ist die Erfassung des aktuellen Prozesszustandes notwendig. Dies kann mit Hilfe der Anbindung eines Workflow-Management-Systems erfolgen oder, falls die einzelnen Prozessmodelle in Form von Fakten und Regeln auf das Inferenzsystem abgebildet wurden, durch Abfrage des Status im Inferenzsystem.

Bei der Ausgabe der modellgesteuerten Laufzeitumgebung handelt es sich um einen dynamisch generierten HTML-Code, der für jeden Zugriff auf eine Seite durch den Besucher der Website neu generiert wird. Die Teilbereiche der Webseite, die aufgrund der Benutzerinteraktion neu erstellt werden müssen, können vom System anhand des Navigationsmodells entnommen werden, dessen zugrunde liegender Statechart-Formalismus den Zustand der einzelnen Teilbereiche berücksichtigt. Prinzipiell existieren ebenso wie bei gängigen Content-Management-Systemen verschiedene Verfahren zur Optimierung des Seitenzugriffs, auf die jedoch nicht weiter eingegangen werden soll.

Abbildung 10-2 zeigt die Architektur des Gesamtsystems und das Zusammenwirken der einzelnen Komponenten sowohl zur Modellierungszeit als auch zur Laufzeit. Die Architektur enthält die wesentlichen Komponenten und kennzeichnet den Informationsfluss sowie die Abhängigkeiten zwischen den einzelnen Komponenten.

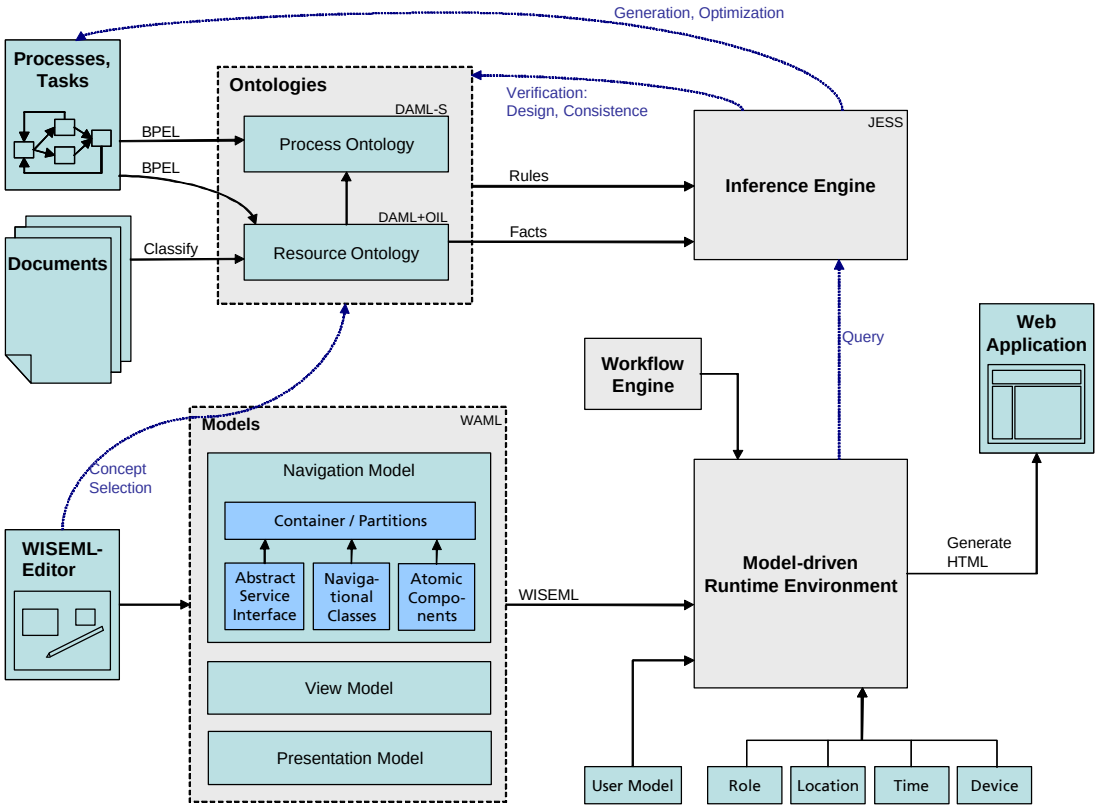


Abbildung 10-2: Referenzarchitektur des Gesamtsystems

11 Bewertung der Vorgehensweise

Die Bewertung des in dieser Arbeit vorgestellten Methodenverbundes bezieht sich sowohl auf den Modellierungsprozess webbasierter Applikationen als auch auf die softwaretechnische Unterstützung beim Entwurf der Anwendung bis hin zur automatischen Generierung eines zugehörigen Prototyps. Im Hinblick auf die in Kapitel 2.3 aufgestellten Zielsetzungen sollen im Folgenden die Konzeption des Methodenverbundes, die Anwendbarkeit der Entwurfsmodelle für den Benutzer sowie die Realisierbarkeit entsprechender Unterstützungswerkzeuge untersucht werden.

11.1 Untersuchung des Methodenverbundes

Der Methodenverbund stellt Metamodelle für das konzeptuelle Design, das Navigationsdesign, das Zugriffs- und Sichtendesign und für das Präsentationsdesign einer Web-Anwendung zur Verfügung. Sowohl das konzeptuelle Modell als auch das Navigationsmodell können zum einen als eigenständige Modelle betrachtet werden, d. h. der Entwurf des Navigationsmodells auf der Grundlage des konzeptionellen Designs ist nicht zwingend erforderlich. Hierdurch wird beim Entwurf einer Web-Anwendung eine Evaluation beider Modelle unabhängig voneinander ermöglicht, die sich ausschließlich auf deren syntaktische Korrektheit bezieht. Eine prototypische Umsetzung des Navigationsmodells ist in diesem Fall nicht möglich. Zum anderen lässt sich das Navigationsmodell auf der Grundlage des konzeptuellen Modells erstellen, so dass sich neben der syntaktischen Verifikation der Modelle auch die Möglichkeit bietet, zur Evaluation einen Prototyp der Anwendung durch entsprechende softwaretechnische Werkzeuge zu generieren. Präsentations- und Sichtenmodell stellen hingegen eine Erweiterung des Navigationsmodells dar und sind von diesem abhängig. Evaluationsmöglichkeiten beziehen sich hier auf entsprechende Ergänzungen des Prototyps.

Da die einzelnen Bearbeitungsschritte bei der Entwicklung der Web-Applikation aufeinander aufbauen, ist die Überführbarkeit der Modelle in die jeweils nachfolgende Entwurfsphase sichergestellt. Zudem können sowohl das Sichtenmodell als auch das Präsentationsmodell unmittelbar in den Entwurf des Navigationsmodells integriert werden, so dass die jeweiligen Entwurfsphasen nahtlos ineinander übergehen.

Ein wichtiges Kriterium stellt die Wiederverwendbarkeit von Teilen des Entwurfs dar, die es dem Entwickler erlaubt, einmal modellierte Navigations- und Darstellungsmuster im Entwurf wiederholt einzusetzen. Durch die Berücksichtigung dynamisch generierter Inhalte und Kontextstrukturen zur Laufzeit erlaubt der Methodenverbund die Definition derartiger Muster auf Basis von Templates (vgl. Kapitel 9.4).

11.2 Bewertung der Anwendbarkeit des Methodenverbundes

Zur Bewertung der Anwendbarkeit des Methodenverbundes wurden Untersuchungen vorgenommen mit dem Ziel, das Vorgehen beim Entwurf von Websites mit dem Modellierungsansatz des Methodenverbundes zu vergleichen. Folgende weiteren Untersuchungsziele wurden mitberücksichtigt:

- Intuition bzgl. des Vorgehens bei der Modellierung und Vergleich zum Methodenverbund
- Erlernbarkeit der Entwurfsmodelle
- Lernbereitschaft und Lerneffizienz hinsichtlich der Anwendung der Modelle
- Verständlichkeit der Entwurfsmodelle

Durchgeführt wurden die Untersuchungen im Rahmen von Befragungen mittels der Evaluationsmethode *Interview*. Die Aufgabenstellung umfasste zunächst den Entwurf einer schematischen Darstellung zu einer konkreten Website, der schrittweise ergänzt wurde. Gegenstand der anschließenden Befragung war die Anwendung der einzelnen Entwurfsmodelle des Methodenverbundes.

Die Gruppe der dreizehn befragten Personen bestand etwa zur Hälfte aus Usability-Experten, deren Arbeitsschwerpunkte vorrangig im Design und der Evaluation von Benutzungsschnittstellen, insbesondere von Web-Interfaces liegen. Der übrige Teil der Gruppe setzte sich aus Personen mit unterschiedlichen Kenntnissen zusammen, die nicht notwendigerweise Erfahrungen in der Gestaltung von Websites haben mussten.

Entsprechend der Untersuchungsziele orientiert sich die Bewertung der Vorgehensweise sowohl an den Hauptkriterien Verständlichkeit, Erlernbarkeit und Anwendbarkeit der Entwurfsmodelle als auch an dem Nutzen der Vorgehensweise insgesamt und dessen softwaretechnischen Unterstützungsmöglichkeiten.

11.2.1 Intuitives Vorgehen bei der Modellierung

Die von den Probanden erstellten schematischen Modelle der Website zeigten untereinander eine hohe Übereinstimmung bzgl. der Komponenten, die gleichzeitig auf einer Webseite dargestellt werden. Unterschiedliche Ansätze wurden jedoch bei der Modellierung von Teilbereichen einer Seite, deren Inhalte sich verändern (z. B. aufgrund eines Navigationsmenüs), verfolgt. Die der Aufgabe zugrunde liegende Problemstellung bezog sich vor allem auf die Frage, nach welchem Prinzip bei der Modellierung zwischen konjunktiven und disjunktiven Darstellungsbereichen unterschieden wird. Tabelle 11-1 zeigt die grafischen Lösungsmöglichkeiten, die hierzu vorgeschlagen wurden.

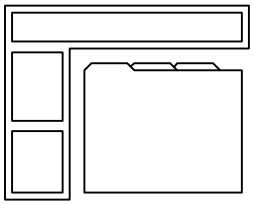
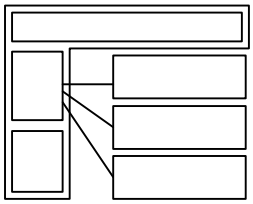
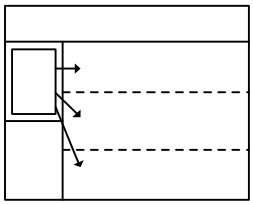
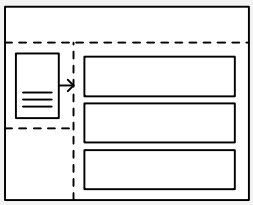
Ansatz	Darstellung	Erläuterung	Anteil (%)
1		Konjunktiv: umfassender Rahmen Disjunktiv: Verwendung von Karteireitern	31
2		Konjunktiv: umfassender Rahmen Disjunktiv: kein umfassender Rahmen	46
3		Konjunktiv: durchgezogene Linie Disjunktiv: gestrichelte Linie	23
4		Konjunktiv: gestrichelte Linie Disjunktiv: keine Linie	-

Tabelle 11-1: Unterschiedliche Darstellungsmöglichkeiten konjunktiver und disjunktiver Bereiche; Ansatz 4 zeigt die in dieser Arbeit konzipierte Notation

Ansatz 1 beinhaltet eine Technik, die es erlaubt, einzelne Bereiche im Modell hervorzuheben. Im Hintergrund befindliche Bereiche werden an anderer Stelle innerhalb einer neuen schematischen Gesamtdarstellung präzisiert. Dieses Vorgehen zeigt aufgrund der Einzelscreen-Darstellung starke Bezüge zum *Storyboarding*[17] und wurde ausschließlich von Personen gewählt, die diesbezüglich bereits Erfahrungen gesammelt hatten.

Die zweite Lösung beruht auf der Kennzeichnung konjunktiver Teilbereiche durch eine Umrandung. Disjunktive Komponenten werden innerhalb des Modells parallel

[17] Unter dem Begriff *Storyboarding* wird der konzeptionell-planerische Einsatz spezieller Skizzen verstanden, die eine formalisierte Darstellung der gesamten Anwendung beinhalten.

dargestellt. Bei dieser Technik wird von der Präsentation der einzelnen Webseiten der Anwendung abstrahiert und eine Übersicht innerhalb des Modells erstellt.

Eine ähnliche Vorgehensweise findet sich im dritten Ansatz, bei dem parallel darzustellende Bereiche durch eine durchgezogene Linie und disjunktive Bereiche durch eine gestrichelte Linie voneinander getrennt werden. Dieser Lösungsansatz entspricht im prinzipiellen Vorgehen dem Modellierungsansatz dieser Arbeit (Ansatz 4), jedoch ist die grafische Notation bzgl. der Linienart gegensätzlich.

Auch wenn sowohl die Effizienz als auch die Konsistenz der verschiedenen Lösungsansätze unterschiedlich zu bewerten ist, weisen alle Modelle eine starke Orientierung an der tatsächlichen Darstellung einer Webseite auf. Der überwiegende Teil der Probanden favorisierte einen Ansatz, der hinsichtlich disjunktiver Teilbereiche von der hohen Ausrichtung auf die Präsentation der Seite abstrahierte. Hier zeigten sich Unterschiede in den einzelnen Modellen in erster Linie bei der grafischen Notation der disjunktiven Darstellung.

Bei der Erstellung der schematischen Darstellungen bestand die übliche Vorgehensweise darin, konjunktive Bereiche nicht zusätzlich, z. B. mittels einer gestrichelten Linie, voneinander zu trennen. Die Bereiche wurden stattdessen als umrandete Komponenten modelliert oder mit Hilfe einer durchgezogenen Linie unterteilt. Auch wenn diese Methodik weitgehend einem intuitiven Vorgehen entspricht, beurteilte der überwiegende Teil bei einer direkten Gegenüberstellung die auf Grundlage der Notation dieser Arbeit erstellten Modelle im ersten Eindruck positiver. Die Bewertung bezog sich dabei auf die Kriterien Strukturiertheit, Verständlichkeit und Ähnlichkeit zu einer Webseite. Nachdem den Testpersonen beide Notationen erläutert wurden, fiel das Ergebnis gegenüber der ursprünglichen Bewertung deutlicher aus. Abbildung 11-1 fasst das Resultat zusammen.

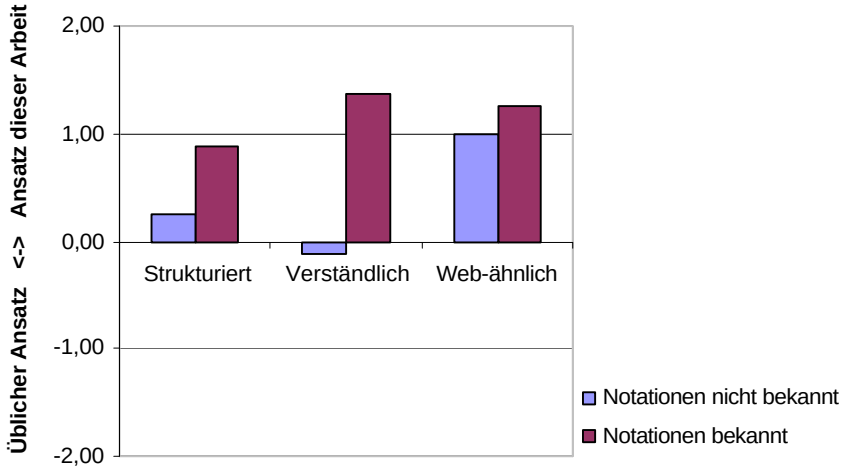


Abbildung 11-1: Resultat der Gegenüberstellung beider Modelle

11.2.2 Lernbereitschaft und Lerneffizienz

Ohne eine genaue Kenntnis über die in dieser Arbeit konzipierte Notation zu besitzen, wirkten die unter Verwendung dieser Notation erstellten Modelle im Vergleich zum intuitiven Ansatz der Testpersonen nur geringfügig strukturierter jedoch auch etwas weniger verständlich. Die geringe Abweichung lässt darauf schließen, dass die Entwurfsmodelle dieser Arbeit den Benutzer auf den ersten Blick nicht überfordern. Unter Kenntnis der beiden Notationen wirken sie sogar deutlich strukturierter und verständlicher gegenüber der intuitiven Vorgehensweise.

Dieser Eindruck wird durch die Ergebnisse der Anwendung der Entwurfsmodelle unterstrichen. Zu einem vorgegebenen Modell (s. Abbildung 6-9) sollten nach einer maximal vierminütigen Erläuterung die entsprechenden Webseiten hinsichtlich Aufbau, Struktur und Navigation skizziert werden. Die Aufgabe wurde von allen Probanden fehlerfrei gelöst. Dementsprechend empfanden fast alle Probanden die Notation als eher unkompliziert und relativ leicht erlernbar.

11.3 Bewertung der softwaretechnischen Unterstützung

Auf der Grundlage der in dieser Arbeit entwickelten Konzeption der Entwurfsmodelle und der aufgezeigten technischen Umsetzungsmöglichkeiten (vgl. Kap. 10) wurde ein Unterstützungswerkzeug implementiert, das es erlaubt, anhand der vom Anwender erstellten Modelle einen Prototypen der Website zu generieren (Häusser et al., 2003; Wissen & Ziegler, 2003a, 2003b, 2003c). Das System unterstützt den Anwender zunächst bei der Konzeption und dem Entwurf eines konzeptionellen Modells, dessen Konzepten konkrete Objekte des Informationsbestands als Instanzen zugewiesen werden können. Darauf aufbauend ermöglicht das Werkzeug den Aufbau der Kompositions- und Navigationsmodelle. Hier werden entsprechend der verwendeten Modellklassen Konzepte bzw. Instanzen des konzeptuellen Modells den einzelnen Komponenten zugeordnet. Bevor anhand der Modelle automatisch ein Prototyp der Anwendung generiert wird, können unterschiedliche Sichten sowie verschiedene Layoutinformationen anhand definierter Stylesheets erzeugt werden.

Bzgl. der Anwendung des Systems existieren aufgrund des prototypischen Charakters bislang keine Evaluationsergebnisse hinsichtlich Benutzerakzeptanz und Verbesserungen des Entwicklungsprozesses webbasierter Anwendungen. Das System zeigt jedoch die grundsätzliche Verwendbarkeit der Metamodelle zur automatisierten Anwendungsgenerierung.

12 Zusammenfassung und Ausblick

12.1 Zusammenfassung

In der vorliegenden Arbeit wurde eine Vorgehensweise für den komponentenorientierten Entwurf webbasierter Anwendungen vorgestellt, die den schrittweisen Entwurf sowohl content- aus auch applikationsorientierter Web-Anwendungen innerhalb eines Web-Engineering-Prozesses unterstützt. Die Methodik bezieht sich auf Modellierungsaspekte der Anwendungsentwicklung und ist auf eine enge Kopplung zwischen der Erstellung bzw. Aktualisierung des Modells und der Bereitstellung der Web-Anwendung zur Laufzeit durch generative und konfigurative Verfahren ausgerichtet. Zu den einzelnen Prozessschritten wurden Metamodelle konzipiert, die ausgehend von der Modellierung der Informations- und Metadatenstruktur den Entwurf des Navigations-, Sichten- und Präsentationsmodells einer Anwendung erlauben.

Die Entwurfsmethodik sieht die Modellierung der Anwendung auf der Grundlage einer grafischen Notation der verschiedenen Modellkomponenten vor, deren formale Beschreibung eine Überführung der Entwurfsmodelle in einen Prototyp der Website ermöglicht. Im Unterschied zu bislang entwickelten Ansätzen orientiert sich der Modellentwurf am strukturellen Aufbau und der tatsächlichen Darstellung der zu generierenden Website. Obwohl nicht nur die Funktionalität der Methodik, sondern auch deren Anwendbarkeit entscheidende Kriterien für den erfolgreichen Einsatz sind, wurde in bisherigen Arbeiten bei der Konzeption der verschiedenen Vorgehensweisen auf eine zielgruppengerechte und benutzungsorientierte Ausrichtung verzichtet.

Anhand eines Anwendungsbeispiels wurde aufbauend auf einer modellierten Themenstruktur und entsprechender Ressourcenzuordnung der Entwurf der Seiten- und Navigationsstruktur demonstriert. Neben der grafischen Modellentwicklung lag zur Verdeutlichung der Funktionsweise der Entwurfsmodelle ein Schwerpunkt in der Anwendung der formalisierten Komponenten.

Im Hinblick auf die technische Unterstützung des Automatisierungsprozesses anhand der formalen Beschreibung der Modelle wurden Integrationsmöglichkeiten der Vorgehensweise und des zugehörigen Methodenverbundes in den systemgestützten Entwicklungsprozess webbasierter Anwendungen aufgezeigt. Hieraus wurde eine Referenzarchitektur abgeleitet, die den Anforderungen einer weitestgehend automatisierten Generierung der Website auf der Grundlage der Metamodelle gerecht wird.

Die Verwendung der in dieser Arbeit konzipierten Entwurfsmodelle wurde im Rahmen von Anwendungstests untersucht. Dabei hat sich gezeigt, dass die Ausrich-

tung der Modelle an Aufbau und Struktur von Webseiten die intuitive Vorgehensweise der Benutzer bei der Modellierung unterstützen und die Methodik insgesamt in ihrer Verständlichkeit und Erlernbarkeit eine benutzergerechte Anwendung zulässt.

12.2 Ausblick

Sowohl die Systemunterstützung bei der Modellentwicklung als auch die automatisierte Verarbeitung der Entwurfsmodelle zur Generierung eines Prototyps der Web-Anwendung basieren auf Technologien zum Verwalten und Editieren von Content. Demzufolge greift die in dieser Arbeit entwickelte Vorgehensweise den Gedanken des Web-Content-Management auf und erweitert diesen entsprechend des Web-Engineering-Prozesses um Aspekte der Anwendungsmodellierung. Weiterentwicklungen der systemtechnischen Unterstützung der Methodik sind daher vor allem in der Integration der werkzeuggestützten Modellierung in Web-Content-Management-Lösungen zu sehen.

Die Konzeption der Entwurfsmodelle basiert auf einem komponentenorientierten Ansatz, der einen feingranularen Aufbau einzelner Webseiten ermöglicht. Um wiederkehrende Muster beispielsweise hinsichtlich der Menüstruktur einer Web-Anwendung nicht wiederholt modellieren zu müssen, läge eine Weiterentwicklung in der Berücksichtigung navigationaler Patterns. Durch das Prinzip der navigationalen Klassen und der Komponentenorientierung ließen sich einmal modellierte komplexere Navigationsmechanismen wieder verwenden.

Schließlich stellt eine mögliche Weiterentwicklung die zusätzliche Unterstützung einer aufgabenbasierten Navigation neben der reinen inhaltsorientierten Informationsaufbereitung dar. Hierzu ist eine Erweiterung der Entwurfsmodelle in Bezug auf die Modellierung von Prozessabläufen notwendig, so dass navigationale Relationen in Abhängigkeit von Prozesszuständen definiert werden können.

13 Summary

The development of web-based information systems is in its complexity tantamount to the design and implementation of conventional software. However, currently software engineering approaches are not applied in development practice, i.e., the development process of web-based applications is not based on a systematic process comprising modeling aspects. This has not only adverse effects on the quality of the application but, above all, results in high maintenance and support expenses due to the high modification dynamics of web-based applications as well as complex navigation relations and content structures.

This thesis introduces an approach for the component-oriented design of web-based applications supporting a step-by-step design of content-oriented as well as application-oriented web applications. The methodology refers to the modeling aspects of application development and is oriented towards a tight interlinking between the creation and updating of the model and providing the web-application at runtime through applying generative and configurative techniques. Proceeding from the design aspects of the information and metadata structure of an application, modeling techniques for the construction of a navigation, view and representation model are developed in a consistent approach. The design methodology provides for the modeling of the application based on a graphic notation of the different model components. Their formal description enables the transfer of the design models to a website prototype.

With the aid of an example of use and based on a modeled theme structure as well as the respective resource classification, the design of the page and navigation structure of a web application is demonstrated. Besides the graphic model development, the focus lies, for clarification of the functionality of the design models including their automatizable processing, on the application of formalized components. With regard to the technical support of the automation process based on the formal model description, integration possibilities of the approach and the respective methodology network in the system-supported development process are presented. From this, a reference architecture is derived which meets the requirements of a highly automated generation of the website based on the models created.

The use of the design models conceived in this paper was studied in application tests. These revealed that the alignment of the models to the composition and structure of websites support the user’s intuitive course of action in modeling and the overall methodology allows a user-friendly application due to its understandability and learnability.

14 Literaturverzeichnis

Ankolenkar, A.; Burstein, M.; Hobbs, J. R.; Lassila, O.; Martin, D. L.; McIlraith, S. A.; Narayanan, S.; Paolucci, M.; Payne, T.; Sycara, K.; Zeng, H. (2001): *DAML-S: A Semantic Markup Language For Web Services.* In: Proceedings of the Semantic Web Working Symposium (SWWS), Stanford, California.

Atzeni, P.; Mecca, G.; Merialdo, P. (1998a): *Design and Maintenance of Data-Intensive Web Sites*. In: Ramos, I.; Schek, H. J.; Saltor, F.; Alanso, G. (editors): Proceedings of the International Conference on Extending Database Technology, EDBT98, Valencia, Spain, page 436-450, March 1998.

Atzeni, P.; Mecca, G.; Merialdo, P.; Masci, A.; Sindoni, G. (1998b): *The Araneus Web-Based Management System*. In: Haas, L. M.; Tiwary, A. (editors): Proceedings of the International Conference Sigmod98, Exhibits Program, Seattle, WA, USA, page 544-546, June 1998.

Baeza-Yates, R. (1999): *Modern Information Retrieval.* Addison-Wesley, New York.

Barnes, H.; Gwyer, M. (1996): *Designer/2000 Web enabling your applications.* Oracle white paper. Technical report (March.), Oracle Corporation.

Barry, C.; Lang, M. (2001): *A Survey of Multimedia and Web Development Techniques and Methodology Usage.* IEEE Mul-timedia. 8(3), 52-60.

Baumeister, H.; Koch, N.; Mandel, L. (1999): *Towards a UML Extension for Hypermedia Design.* In: *Proceedings of the 2nd International Conference on the Unified Modeling Language – Beyond the Standard (UML'99),* Lecture Notes in Computer Science, Vol. 1723, R. France and B. Rumpe, Eds., Springer, Fort Collins, CO, pp. 614–629.

Berners-Lee, T.; Hendler, J.; Lassila, O. (2001): *The Semantic Web.* In: Scientific American.

Bichler, M.; Nusser, S. (1996a): *Developing Structured WWW-Sites with W3DT*. Vienna University of Business Administration and Economics, Wien, http://www.hyperg. bab.cinvestav.mx

Bichler, M.; Nusser, S. (1996b): *Modular Design of Complex Web-Applications with W3DT.* In: IEEE 5th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, California, June 19 – 21.

Biere, M.; Bomsdorf, B.; Szwillus, G. (1999): *The Visual Task Model Builder*. In: J. Vanderdonckt, A. Puerta (Hrsg.): *Computer-Aided Design of User Interfaces II*, Proceedings of the 3rd International Conference on Computer-Aided Design of

User Interfaces CADUI'99 (Louvain-la-Neuve, 21-23 October 1999), Kluwer Academics, Dordrecht, pp. 245-256.

Bongio, A.; Ceri, S.; Fraternali, P.; Maurino, A. (2000): *Modelling Data Entry and Operations in WebML*. Proceedings of WebDB 2000, Lectures Notes in Computer Science, Vol. 1997, Springer Verlag, 201-214.

Booch G.; Rumbaugh J.; Jacobson I. (1999): *The Unified Modeling Language: A User Guide.* Addison Wesley.

Botafogo, R.; Rivlin, E.; Shneiderman, B. (1992): *Structural analysis of hypertexts: Identifying hierarchies and useful metrics.* ACM Transactions on Information Systems, 10(2):142-180, April 1992.

Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E. (2000): *Extensible Markup Language (XML) 1.0 (Second Edition)*, 6. Oktober 2000. http://www.w3.org/TR/2000/REC-xml-20001006

Brickley, D.; Guha, R. V. (2000): *Resource DescriptionFramework (RDF) - Schema Specification 1.0*, 27 March 2000. URL http://www.w3.org/TR/2000/CR-rdf-schema-20000327.

Bullinger, H.-J. (Hrsg.); Schuster, E.; Wilhelm, S. (2000): *Content Management Systeme: Auswahlstrategien, Architektur und Produkte.* Düsseldorf: Verlagsgruppe Handelsblatt, Wirtschaftswoche.

Ceri, S.; Fraternali, P.; Paraboschi, S. (1999): *Design Principles for Data-Intensive Web Sites.* SIGMOD Record, Vol. 24, No. 1, March 1999.

Ceri, S.; Fraternali P.; Bongio, A. (2000*): Web Modelling Language (WebML): a modelling language for designing Web sites.* In: Proceedings WWW9 Conference, Amsterdam, May 2000.

Chen, P. P. S. (1976): *The entity-relationship model - toward a unified view of data.* ACM Transactions on Database Systems, 1(3), March 1976.

Christodoulou, S.; Styliaras, G.; Papatheodourou, T. (1998): *Evaluation of Hypermedia Application Development and Management Systems.* Proc. of the 9th ACM Conference on Hypertext and Hypermedia, Pittsburgh.

Conallen, J. (1999): *Modeling Web Application Architectures with UML.* In: Communications of the ACM (CACM), Vol. 42, No. 10, October 1999.

Costagliola, G.; Ferrucci, F.; Francese, R. (2002): *Web engineering: Models and methodologies for the design of hypermedia applications.* In: Chang, S. K., editor, *Handbook of Software Engineering & Knowledge Engineering*, volume 2, Emerging Technologies, pages 181 – 199. World Scientific.

Cowan, D.; Ierusalimschy, R.; Lucena, C. J. P.; Stepien, T. M. (1993): *Abstract Data Views.* Structured Programming, 14(1):1-13, January 1993.

Curbera, F.; Goland, Y.; Klein, J.; Leymann, F.; Roller, D.; Thatte, S.; Werawarana, S. (2002): *Business Process Execution Language for Web Services (V 1.0)*, Initial Public Draft Release, (http://www.ibm.com/developerworks/library/ws-bpel)

De Troyer, O.; Leune, C. (1997): WSDM: *A user-centered design method for Web sites.* In: Proceedings of the 7th International World Wide Web Conference.

Diaz, A.; Isakowitz, T.; Maiorana, V.; Gilabert, G. (1995): *RMC: A tool to design WWW applications.* In Proceedings of the Fourth International Conference on World Wide Web (Boston, MA), 11–14.

Dolog, P.; Bielikova, M. (2002): *Hypermedia modelling using UML*. In: Hanacek, P. (ed), Proceedings of ISM'2002 – Information System Modelling, pp 79-86.

Fernandez, M.; Florescu, D.; Kang, J.; Levy, A. (1998): *Catching the Boat with Strudel: Experiences with a Web-Site Management System.* In: Proceedings of Sigmod ’98, Seattle, Washington, USA, page 414 425, June 1998.

Fernandez-Lopez, M., Gomez-Perez, A. & Juristo, N. (1997): *METHODOLOGY: From Ontological Art towards Ontological Engineering.* In: AAAI-97 Spring Symposium on Ontological Engineering. Stanford University, March 24-26.

Florescu, D.; Kossmann, D. (2001): *An XML Programming Language for Web Service Specification and Composition.* IEEE Data Engineering Bulletin 24(2): 48-56.

Fowler, M. (1997): *Analysis Patterns: Reusable Object Models.* Object Technology Series. Addison-Wesley, Reading, MA.

Fox, M. S.; Gruninger, M. (1998): *Enterprise Modeling*. AI-Magazine 19:3, 109–121.

Fraternali, P. (1999): *Tools and approaches for data-intensive Web applications: A survey.* ACM Computing Surveys, Vol. 31, No. 3, September 1999.

Fraternali, P.; Paolini, P. (2000): *Model-Driven Development of Web Applications: The Autoweb System.* ACM Transactions on Information Systems, 18(4):323 382.

Friedman-Hill, E. J. (1999): *Jess, The Java Expert System Shell*., http://herzberg.ca.sandia.gov/jess (Stand 21.09.1999).

Fuentes, L.; Troya, J. M.; Vallecillo, A. (2002): *Using UML Profiles for Documenting Web-Based Application Frameworks.*, Annals of Software Engineering 13, pp 249 - 264.

Gaedke M. (1999): *Wiederverwendung von Komponenten in Web-Anwendungen*. Universität Karlsruhe; 1999. In Turowski K. (Hrsg.): Tagungsband des 1. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 1), Magdeburg, S. 31-36.

Garzotto, F.; Paolini, P.; Schwabe, D. (1993a): *HDM: A model-based approach to hypertext application design.* ACM Transactions on Information Systems, 11(1):1-26, January 1993.

Garzotto, F., Paolini, P., and Mainetti, L. (1993b): *Navigation patterns in hypermedia data bases.* In Proceedings of the 26th Hawaii International Conference on System Sciences, IEEE Computer Society Press, 370-379.

Gellersen, H. (1997): *Web Engineering: Softwaretechnik für Anwendungen im World-Wide Web.* In Theorie und Praxis der Wirtschaftsinformatik, Heft 196, Juli 1997.

Gellersen, H.; Gaedke, M. (1999): *An Object-Oriented Model for the Web Application Development Process*. IEEE Internet Computing, January - Feb 1999.

Ginige, A. (1998): *Web Engineering: Methodologies for Developing Large and Maintainable Web Based Information Systems.* Proc IEEE International Conference on Networking the India and the World CNIW'98, Ahmedabad, India, Dec 1998.

Gruber, T. R. (1993): *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* Technical Report KSL 93-04, Knowledge Systems Labroratory, Stanford University, Stanford 1993 (b). In: Guarino, N.; Poli, R. (Hrsg.): *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Amsterdam. (http://ksl-web.stanford.edu/KSL_Abstracts/KSL-93-04.html, 10.06. 1999).

Harel, D. (1987): *Statecharts: A Visual Formalism for Complex Systems*. In: *Science of Computer Programming.* Elsevier Science Publishers. North Holland. S. 231-274.

Harel, D. (1988): *On visual formalisms.* Communications of the ACM, 31(5), pp. 514–530.

Häusser, T.; Mirochnitchenko, M.; Rindermann, M. (2003): *Web Ontology Management Application.* In Szwillus, G.; Ziegler, J. (Hrsg.): Mensch & Computer 2003: Interaktion in Bewegung. Stuttgart: B. G. Teubner, S. 247-256.

Hay, D. C. (1996): *Data Model Patterns: Conventions of Thought.* Dorset House Publishing, New York, NY.

Heflin, J.; Hendler, J.; Luke, S. (1999): *SHOE: A Knowledge Representation Language for Internet Applications.* Technical Report CS-TR-4078, University of Maryland, College Park.

Henderson-Sellers, B. (1995): *Who needs an object-oriented method anyway?* The Journal of Object Oriented Programming, vol. 8, No. 6.

Hendler, J.; McGuinnes, D. L. (2001): *Darpa Agent Markup Language.IEEE Intelligent Systems.*, 15(6):72-73.

Hennicker, R.; Koch, N. (2000): *A UML-based Methodology for Hypermedia Design.* In Evans, A.; Stuart, S.; Selic, B. (editors): UML'2000 - The Unified Modeling Language - Advancing the Standard, volume 1939 of Lecture Notes in Computer Science, York, England, October 2000. Springer Verlag.

Isakowitz, T.; Stohr, E. A.; Balasubramanian, P. (1995): RMM: *A methodology for structured hypermedia design.* Communications of the ACM, 38, pp. 34-44.

Jacobson, I.; Booch, G.; Rumbaugh, J. (1999): *The Unified Process.* IEEE Software Mai./Juni 1999, S. 96-102. IEEE Computer Society, New York.

Jin, Y.; Decker, S.; Wiederhold, G. (2001): *OntoWebber: Model-Driven Ontology-Based Web Site Management.* The 1st International Semantic Web Working Symposium (SWWS'01), Stanford University, Stanford, CA, July 29-Aug 1.

Jin, Y.; Xu, S.; Decker, S. (2002): *OntoWebber: A Novel Approach for Managing Data on the Web.* In: The 8th International Conference on Extending Database Technology (EDBT 2002). Prague, Czech Republic, March 24-28.

Karp, P. D.; Chaudhri, V. K.; Thomere, J. (1999): *XOL: An XML-Based Ontology Exchange Language.* ftp://smi.stanford.edu/pub/bio-ontology/xol.doc

Kent, R. E. (1999): *Conceptual Knowledge Markup Language: The Central Core.* In: Proceedings of the 12 Workshop on Knowledge Acquisition, Modeling and Management, Banff. http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html

Kifer, M.; Lausen, G. (1989): *F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme.* In: ACM SIGMOD Record, 18.2, 134 – 146.

Koch, N. (1998): *Towards a Methodology for Adaptive Hypermedia Systems Development.* In *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Proceedings of the 6th Workkshop, ABIS-98, U. Timm and M. Roessel (Eds.).

Koch, N. (1999): *A Comparative Study of Methods for Hypermedia Development.* Technical Report 9905, Ludwig-Maximilians-Universität München, November 1999.

Koch, N. (2002): *Reference Model, Modeling Techniques and Development Process.* Software Engineering for Adaptive Hypermedia Systems. KI 16(3): 40-41.

Lam, W.; Ruiz, M. E.; Srinivasan, P. (1999): *Automatic text categorization and its applications to text retrieval.* IEEE Transactions on Knowledge and Data Engineering, 11(6):865-879, 1999.

Lange, D. (1994): *An Object-Oriented Design Method for Hypermedia Information Systems.* In: Proceedings of the 28th Hawaii International Conference on System Services, Jan 1994.

Lange, D. (1996): *An object-oriented design approach for developing hypermedia information systems.* Journal of Organizational Computing and Electronic Commerce, 6(3), 269-293.

Lee H.; Lee C.; Yoo C. (1998): *A scenario-based object-oriented methodology for developing hypermedia information systems.* In: Proceedings of 31st Annual Conference on Systems Science, Eds. Sprague R.

Lee, H.; Lee, C.; Yoo, C. (1999): *A Scenario-based Objectoriented Design Methodology.* Information and Management, Vol. 36, pp. 121-138.

Lenat, D. B.; Guha, R. V. (1990): *Building large knowledge-based systems.* Addison-Wesley.

Lowe, D.; Hall, W. (1999): *Hypermedia and the Web: An Engineering Approach.* John Wiley & Sons, New York.

Lowe, D.; Webby, R. (1999): *Utilisation of process modeling in improving the hypermedia development process,* The New Review of Hypermedia and Multimedia, 5, 133-150, Taylor Graham Publishers.

MacGregor, R.; Bates, R. (1987): *The Loom knowledge representation language.* In Proceedings Knowledge-Based Systems Workshop, St.Louis.

Minsky, M. (1975): *A Framework for Representing Knowledge*., The Psychology of Computer Vision, McGraw-Hill.

Mitchell, T. M. (1996): *Machine Learning.* McGraw Hill, NY, US.

Morik, K.; Wrobel, S.; Joachims, T. (2000): *Maschinelles Lernen und Data Mining.* In: G. Görz, C. Rollinger, J. Schneeberger: *Einführung in die Künstliche Intelligenz.* 2000. http://www.cl.uni-heidelberg.de/kurs/ss01/ml/

Motik, B.; Maedche, A.; Volz, R. (2002): *A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications.* Proceedings of the First International Conference on Ontologies, Databases and Application of Semantics (ODBASE-2002), Springer, LNAI, California, USA.

Murugesan, S.; Deshpande, Y.; Hansen, S.; Ginige, A. (2001): *Web Engineering: A New Discipline for Development of Web-Based System*. In: *Web Engineering: Managing Diversity and Complexity of Web Application Development,* San Murugesan and Yogesh Deshpande (editors.), LNCS – Hot Topics, Vol. 2016, Springer Verlag, Heidelberg, Germany, pp 3-14.

Neches, R.; Fikes, R.; Finin, R.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W. R. (1991): *Enabling technology for knowledge sharing*. AI Magazine: 36-56.

Nielson, J. (1993): *Usability Engineering.* San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Olsina, L. (1998): *Building a Web-based information system applying the hypermedia flexible process modeling strategy.* 1st International Workshop on Hypermedia Development, Hypertext´98.

Pagel, B.-U.; Six, H.-W. (1994): *Software Engineering: Die Phasen der Softwareentwicklung.* Addison-Wesley.

Parsons, J.; Wand, Y. (1997): *Using Objects for Systems Analysis.* In Communications of the ACM, Vol 40, No. 12., S. 104-100.

Reinhartz-Berger, I.; Dori, D.; Katz, S. (2002): *OPM/Web – Object-Process Methodology for Developing Web Applications.* Annals of Software Engineering 13, pp 141–161.

Retschitzegger, W.; Schwinger, W. (2000): *Towards Modeling of DataWeb Applications - A Requirements' Perspective.* Proc. of the Americas Conf on Information Systems (AMCIS), California, Vol. I, Aug. 2000.

Rossi, G.; Schwabe, D. (1994): *From domain models to hypermedia applications.* In: Workshop of Methodologies for Designing and Developing Hypermedia Applications, Edinburgh, September 1994.

Rossi, G.; Schwabe, D. (1995): *Building hypermedia applications as navigational views of information models.* In Proceedings of Annual Hawaii International Conference on System Science, number 28, January 1995.

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W. (1991): *Object Oriented Modeling and Design.*, Prentice Hall Inc.

Rumbaugh, J. (1995): *What is a method?* The Journal of Object Oriented Programming, vol. 8, No. 6.

Schmidt, A.; Beigl, M.; Gellerson, H. W. (1999): *There is more to Context than Location.* Computers & Graphics Journal, Elsevier, Volume 23, No.6, December 1999.

Schwabe, D.; Rossi, G. (1998): *Developing hypermedia applications using OOHDM.* In: Proceedings of Workshop on Hypermedia development Process, Methods and Models, Hypertext'98.

Sciore, E. (1989): *Object specialization*. ACM Transactions on Information Systems, 7:2, 103–122.

Sebastiani, F. (2002): *Machine Learning in Automated Text Categorization.* ACM Computer Surveys, Vol. 34, No. 1, pp. 1-47.

Siau, K.; Cau, Q. (2001): *Unified Modeling Language (UML) – A Complexity Analysis.* Journal of Database Management 12, 1, 26–34.

Spies, M. (2002): *Wissensmanagement: Trends, Ansätze, Herausforderungen.* In I. f. S. u. Wirtschaftförderung (Ed.): *Wissensmanagement: Wettbewerbsvorteile für den Mittelstand.* Halle.

Szwillus, G.; Bomsdorf, B. (2002): *Strukturierte Entwicklung von Websites: Welche Modelle sind relevant?* Workshop Mensch und Computer 2002, Hamburg, http://mc.informatik.uni-hamburg.de/konferenzbaende/mc2002/w2-WSSzwillusBomsdorf.doc

Szwillus, G.; Bomsdorf, B. (2002): *Models for Task-Object-Based Web Site Management*, In: Proceedings of DSV-IS 2002, Rostock.

Tanzer, D.; Shasha, D. (1999): *Queryable Acyclic Production Systems.* Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, pp. 284-291.

Uhr, H. (2001): *Die Definition und Ausführung von benutzerspezifischen Webprozessen mit TOMBOLA*, Poster. In: Informatiktage 2001: Fachwissenschaftlicher Informatik-Kongress, Gesellschaft für Informatik, Bad Schussenried, November 2001, S. 322.

Uschold, M.; King, M. (1995): *Towards a methodology for building ontologies.* Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, Montreal, Canada.

Vilain, P.; Schwabe, D.; de Souza, C. S. (2000): *A Diagrammatic Tool for Representing User Interaction in UML*. In: *Proceedings of the 3rd International Conference on the Unified Modeling Language – Advancing the Standard* (UML'2000), York, UK, Lecture Notes in Computer Science, Vol. 1939, A. Evans, S. Kent and B. Selic, Eds., Springer, pp. 133–147.

Wissen, M.; Ziegler, J. (2003a): *Methoden und Werkzeuge für kooperatives Content Engineering.* In Mambrey, P. / Pipek, V. / Rohde, M. (Hrsg.): Wissen und Lernen

in virtuellen Organisationen. Konzepte, Praxisbeispiele, Perspektiven. Heidelberg: Springer.

Wissen, M.; Ziegler, J. (2003b): *A Methodology for the Component-Based Development of Web Applications.* In Stephanidis, C.; Jacko, J. (Eds.): Proceedings of 10th Int. Conf. on Human - Computer Interaction (HCI International 2003), Vol. 1, Crete, Greece, June 2003.

Wissen, M.; Ziegler, J. (2003c): *Ontologiebasierte Vorgehensweise zur Modellierung komponentenorientierter Web-Anwendungen.* In: Szwillus, G.; Ziegler, J. (Hrsg.): Mensch & Computer 2003: Interaktion in Bewegung. Stuttgart: B. G. Teubner, S. 247-256.

Woukeu, A.; Carr, L.; Wills, G.; Hall, W. (2003): *Rethinking Web Design Models: Requirements for Addressing the Content.* Technical Report ECSTR-IAM03 - 002, Department of Electronics and Comnputer Science, University of Southampton.

Yang, Y. (1999): *An evaluation of statistical approaches to text categorization.* Information Retrieval, 1(1-2):69-90.

Zelewski, S. (2002): *Wissensmanagement mit Ontologien - eine einführende Darstellung*, Arbeitsbericht Nr. 15, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen.

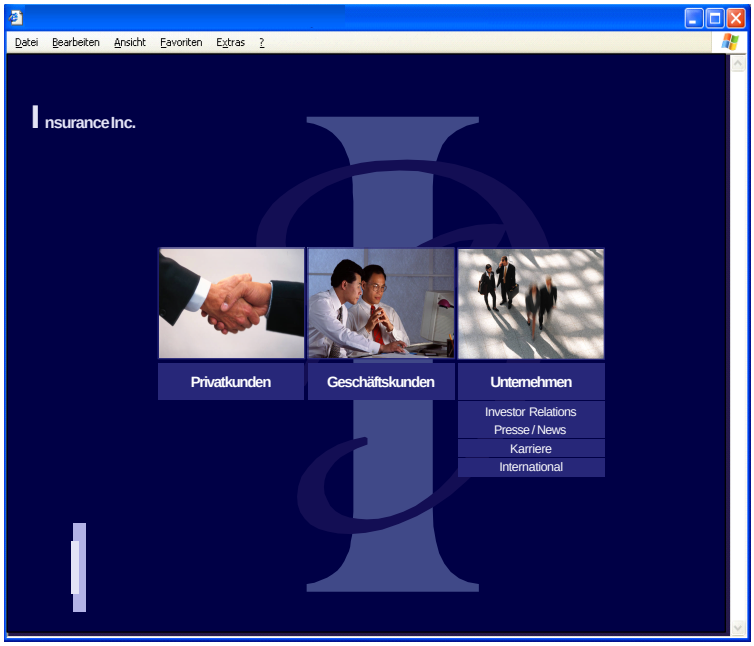
Ziegler, J. E. (1997): *ViewNet - Conceptual design and modelling of navigation*. In S. Howard, J. Hammon & G. Lingaard (Eds.), Human-Computer Interaction: Interact'97, London: Chapman & Hall.

Anhang A: Werkzeuge zur Website-Entwicklung

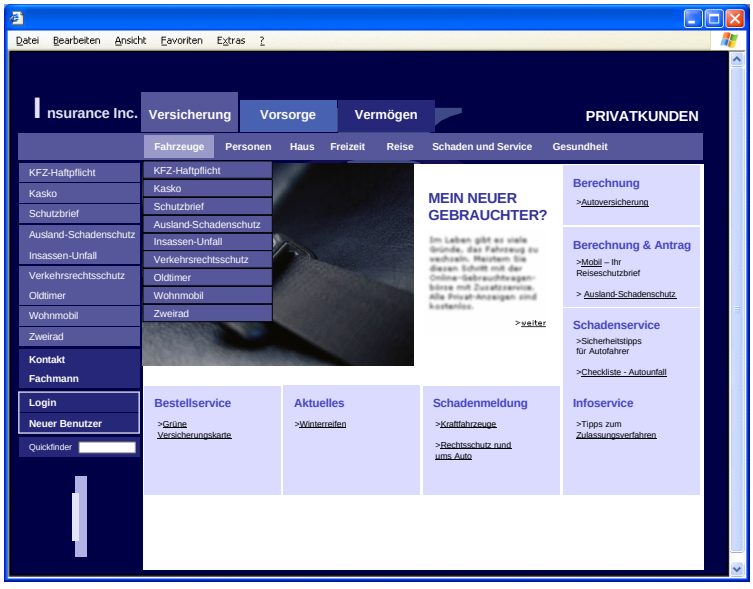
Allaire Cold Fusion Application Wizards	http://www.macromedia.com/
AllenCommunication Quest	http://www.allencomm.com/
Asymetrix Toolbook Assistant / Instructor	http://www.asymetrix.com/
Macromedia Director	http://www.macromedia.com/
Microsoft Access	http://www.microsoft.com/
Microsoft Frontpage	http://www.microsoft.com/
Microsoft Visual Basic	http://www.microsoft.com/
Microsoft Visual InterDev	http://www.microsoft.com/
NetObject Fusion	http://www.netobjects.com/
Oracle Developer 2000	http://www.oracle.com/
Oracle PL/SQL Web Toolkit	http://www.oracle.com/
Oracle Web Development Suite	http://www.oracle.com/
Sybase PowerBuilder Web.PB class Library	http://www.sybase.com/
Vignette StoryServer	http://www.vignette.com/

Anhang B: Darstellungen der modellbezogenen Webseiten[18]

1 Webseite zu Abbildung 9-4 (Container B_1 *Home*)



2 Webseite zu Abbildung 9-5 und Abbildung 9-6 (Container B_2 *Hauptseite*)



[18] Darstellungen nach Allianz (www.allianz.de)

Anhang C: Dokument-Typ-Definition (DTD)

1 Allgemeine Deklarationen

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ============================================================ -->
<!--                                                              -->
<!--  Web Information and Service Engineering (WISE)              -->
<!--  Modeling Language WISEML - Document Type Definition         -->
<!--                                                              -->
<!--                                                              -->
<!--  Version 1.1                                                 -->
<!--                                                              -->
<!-- ============================================================ -->

<!ENTITY % PrimType     "(String | Image | Link | Stream)">
<!ENTITY % NavStatType  "(Index | Group | Panel | Sequence | Tabs)">
<!ENTITY % NavDynType   "(%NavStatType; | Tree)">
<!ENTITY % ObjectType   "(Instance | Concept)">
<!ENTITY % ServiceType  "(Application | WebService)">
<!ENTITY % EventType    "(MouseClick | MouseOver |
                          PopupOpen | PopupClose)">

<!ELEMENT WISE (ConceptualModel, NavigationModel,
                ViewModel?, PresentationModel?)>

<!-- ============================================================ -->
<!--  Import Conceptual Model DTD                                 -->
<!-- ============================================================ -->
<!ENTITY % ConceptualModelDTD SYSTEM "ConceptualModel.dtd">
%ConceptualModelDTD;

<!-- ============================================================ -->
<!--  Import Navigation Model DTD                                 -->
<!-- ============================================================ -->
<!ENTITY % NavigationModelDTD SYSTEM "NavigationModel.dtd">
%NavigationModelDTD;

<!-- ============================================================ -->
<!--  Import View Model DTD                                       -->
<!-- ============================================================ -->
<!ENTITY % ViewModelDTD SYSTEM "ViewModel.dtd">
%ViewModelDTD;

<!-- ============================================================ -->
<!--  Import Presentation Model DTD                               -->
<!-- ============================================================ -->
<!ENTITY % PresentationModelDTD SYSTEM "PresentationModel.dtd">
%PresentationModelDTD;
```

2 Konzeptuelles Modell

```
<!-- ============================================================ -->
<!--                                                              -->
<!--  Conceptual Model                                            -->
<!--                                                              -->
<!-- ============================================================ -->
<!ELEMENT ConceptualModel (Concept*, Property*, Instance*)>

<!-- ============================================================ -->
<!--  Concept                                                     -->
<!-- ============================================================ -->
<!ELEMENT Concept (Parent?)>
<!ATTLIST Concept
          id                    ID                    #REQUIRED
          label                 CDATA                 #REQUIRED
>

<!ELEMENT Parent EMPTY>
<!ATTLIST Parent
          id                    IDREF                 #REQUIRED
>

<!-- ============================================================ -->
<!--  Property                                                    -->
<!-- ============================================================ -->
<!ELEMENT Property (Domain+, Range+)>
<!ATTLIST Property
          id                    ID                    #REQUIRED
          label                 CDATA                 #REQUIRED
>

<!ELEMENT Domain EMPTY>
<!ATTLIST Domain
          concept               IDREF                 #REQUIRED
>

<!ELEMENT Range EMPTY>
<!ATTLIST Range
          concept               IDREF                 #REQUIRED
>

<!-- ============================================================ -->
<!--  Instance                                                    -->
<!-- ============================================================ -->
<!ELEMENT Instance (PropertyRef*)>
<!ATTLIST Instance
          id                    ID                    #REQUIRED
          label                 CDATA                 #REQUIRED
          instanceof            IDREF                 #REQUIRED
>
```

```
<!ELEMENT PropertyRef EMPTY>
<!ATTLIST PropertyRef
          property             IDREF                #REQUIRED
          value                CDATA                #REQUIRED>
```

3 Navigationsmodell

```
<!-- ========================================================== -->
<!--                                                            -->
<!--  Navigation Model                                          -->
<!--                                                            -->
<!-- ========================================================== -->
<!ELEMENT NavigationModel (Container+, Navigationsrelation*, Status-
tabelle?)>

<!-- ========================================================== -->
<!--  Container                                                 -->
<!-- ========================================================== -->
<!ELEMENT Container (Container*, ModelClass*)>
<!ATTLIST Container
          id                   ID                   #REQUIRED
          label                CDATA                #IMPLIED
          partitioned          (no | horz | vert)   "no"
          startcontainer       IDREF                #IMPLIED
>

<!-- ==========================================================-->
<!--  Model Class                                               -->
<!-- ========================================================== -->
<!ELEMENT ModelClass (PrimStaticClass   | PrimDynamicClass |
                      NavStaticClass    | NavDynamicClass  |
                      AbstractServClass | PrimStatusClass)>
<!ATTLIST ModelClass
          id                   ID                   #REQUIRED
>

<!-- ========================================================== -->
<!--  Primitive Static Class                                    -->
<!-- ========================================================== -->
<!ELEMENT PrimStaticClass EMPTY>
<!ATTLIST PrimStaticClass
          type                 %PrimType;           #REQUIRED
          instance             IDREF                #REQUIRED
>

<!-- ========================================================== -->
<!--  Primitive Dynamic Class                                   -->
<!-- ========================================================== -->
<!ELEMENT PrimDynamicClass EMPTY>
<!ATTLIST PrimDynamicClass
          type                 %PrimType;           #REQUIRED
          sourceelement        IDREF                #REQUIRED
          targetconcept        IDREF                #REQUIRED
          relation             IDREF                #REQUIRED
>
```

```
<!-- ========================================================== -->
<!--  Navigational Dynamic Class                                -->
<!-- ========================================================== -->
<!ELEMENT NavDynamicClass EMPTY>
<!ATTLIST NavDynamicClass
          navtype              %NavDynType;         #REQUIRED
          objtype              %ObjectType;         #REQUIRED
          sourceelement        IDREF                #REQUIRED
          element              IDREF                #REQUIRED
          depth                NMTOKEN              #REQUIRED
          properties           IDREFS               #REQUIRED
          attributes           IDREFS               #IMPLIED
          targetcontainers     IDREFS               #REQUIRED
          broadcast            IDREFS               #IMPLIED
>

<!-- ========================================================== -->
<!--  Navigational Static Class                                 -->
<!-- ========================================================== -->
<!ELEMENT NavStaticClass EMPTY>
<!ATTLIST NavStaticClass
          type                 %NavStatType;        #REQUIRED
          navrelations         IDREFS               #REQUIRED
>

<!-- ========================================================== -->
<!--  Abstract Service Class                                    -->
<!-- ========================================================== -->
<!ELEMENT AbstractServClass (Parameter)*>
<!ATTLIST AbstractServClass
          type                 %ServiceType;        #REQUIRED
          method               CDATA                #REQUIRED
>

<!ELEMENT Parameter EMPTY>
<!ATTLIST Parameter
          name                 CDATA                #REQUIRED
          source               CDATA                #REQUIRED
          type                 CDATA                #IMPLIED
          value                CDATA                #IMPLIED
>

<!-- ========================================================== -->
<!--  Primitive Status Class                                    -->
<!-- ========================================================== -->
<!ELEMENT PrimStatusClass EMPTY>
<!ATTLIST PrimStatusClass
          element              IDREF                #REQUIRED
>
```

```
<!-- ========================================================== -->
<!--  Status Table                                              -->
<!-- ========================================================== -->
<!ELEMENT Statustable (StatusIdentifier)*>
<!ELEMENT StatusIdentifier EMPTY>
<!ATTLIST StatusIdentifier
          id                    ID                  #REQUIRED
          name                  CDATA               #REQUIRED
>

<!-- ========================================================== -->
<!--  Navigation Relation                                       -->
<!-- ========================================================== -->
<!ELEMENT NavRelation EMPTY>
<!ATTLIST NavRelation
          id                    IDREF               #REQUIRED
          sourceinstance        IDREF               #REQUIRED
          targetconcept         IDREF               #REQUIRED
          property              IDREF               #REQUIRED
          sourcecomponent       IDREF               #REQUIRED
          targetcomponent       IDREF               #REQUIRED
          event                 %EventType;         #REQUIRED
          broadcast             IDREFS              #IMPLIED
>
```

4 Sichtenmodell

```
<!-- ========================================================== -->
<!--                                                            -->
<!--  View Model                                                -->
<!--                                                            -->
<!-- ========================================================== -->
<!ELEMENT ViewModel (AccessClass)*>

<!ELEMENT AccessClass EMPTY>
<!ATTLIST AccessClass
          modelclass            IDREF               #REQUIRED
          role                  CDATA               #REQUIRED
          condition             CDATA               #REQUIRED
>
```

5 Präsentationsmodell

```
<!-- ============================================================ -->
<!--                                                              -->
<!--  Presentation Model                                          -->
<!--                                                              -->
<!-- ============================================================ -->
<!ELEMENT PresentationModel (PresentationClass)*>

<!ELEMENT PresentationClass EMPTY>
<!ATTLIST PresentationClass
          modelclass           IDREF               #REQUIRED
          role                 CDATA               #REQUIRED
          type                 CDATA               #REQUIRED
          value                CDATA               #REQUIRED
>
```

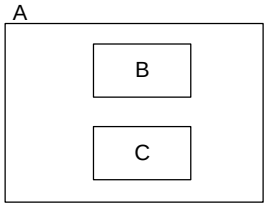
Anhang D: Verwendung der grafischen Notation

Die folgende Zusammenstellung zeigt die Verwendung der zur grafischen Navigationsmodellierung vorgesehenen Komponenten.

3 Container

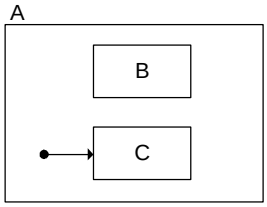
3.1 Subcontainer

Innerhalb von A wird entweder B oder C dargestellt.



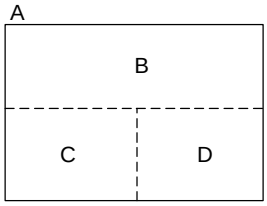
3.2 Startzustand

Innerhalb von A wird zunächst C dargestellt.



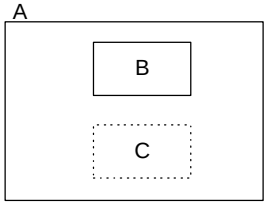
3.3 Partitionierung

Innerhalb von A werden B, C und D gleichzeitig dargestellt.



3.4 Platzhalter

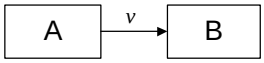
Der Bereich C wird an anderer Stelle modelliert.



4 Navigationsrelationen

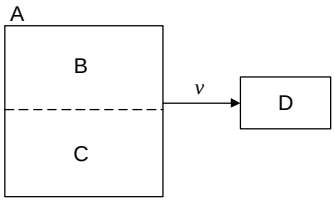
4.1 Modellklasse als Ursprung

Alle Vorkommen der Beschriftung v in A werden mit einem Link versehen. Ist $v = \varnothing$, wird der gesamte Inhalt mit einem Link versehen.



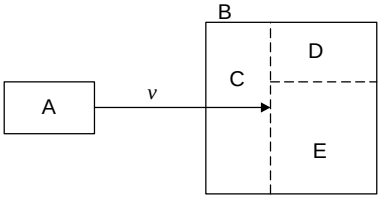
4.2 Container als Ursprung

Die Beschriftung v wird als Link innerhalb von A eingefügt. Ist der Ursprungscontainer partitioniert, wird der Link ebenfalls genau einmal eingefügt.



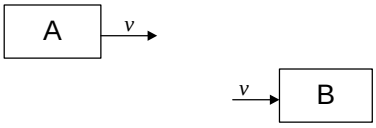
4.3 Partition als Zielbereich

Trifft die Pfeilspitze der Navigationsrelation auf eine Partitionslinie, gelten als Zielbereich alle dahinter liegenden Partitionen (im Beispiel D und E). Soll der Zielbereich nur eine Partition umfassen, muss die Pfeilspitze innerhalb der entsprechenden Partition enden.



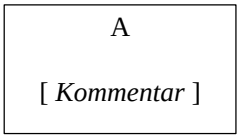
4.4 Verweise

Die grafische Darstellung von Navigationsrelationen kann unterbrochen und an anderer Stelle weitergeführt werden.



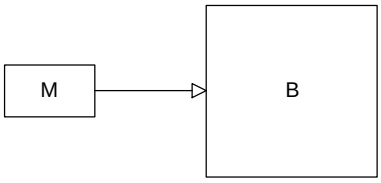
4.5 Kommentare

Zur Angabe von Kommentaren werden eckigen Klammern verwendet. Der Kommentartext ist kursiv.



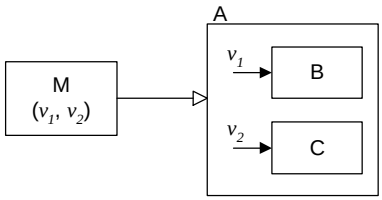
4.6 Linkkollektionen

4.6.1 Zielcontainer enthält keine Subcontainer



Die einzelnen Relationen werden dem Zielcontainer zugeordnet.

4.6.2 Zielcontainer enthält Subcontainer und ist nicht partitioniert

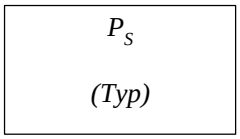


Die einzelnen Relationen können den Subcontainern zugeordnet werden.

5 Modellklassen

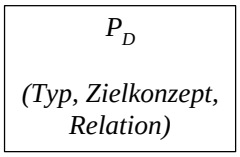
5.1 Primitive Klassen

5.1.1 Primitive statische Klasse



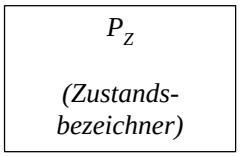
Angabe: Typ

5.1.2 Primitive dynamische Klasse



Angabe: Typ, Zielkonzept, Relation

5.1.3 Primitive Zustandsklasse

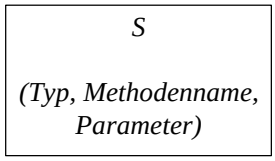


Angabe: Zustandsbezeichner

5.2 Navigationale Klassen

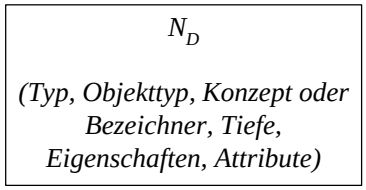
5.2.1 Navigationale statische Klasse

Angabe: Typ, Navigationsrelationen

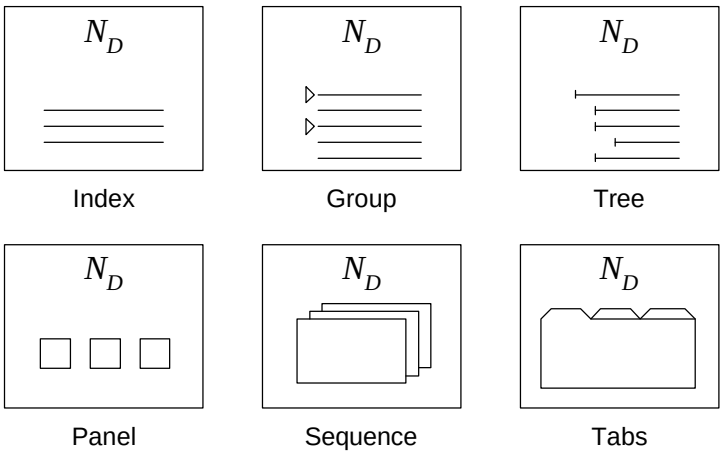


5.2.2 Navigationale dynamische Klasse

Angabe: Typ, Objekttyp, Konzept oder Bezeichner, Tiefe, Eigenschaften, Attribute



5.2.3 Darstellungsformen navigationaler Klassen



5.3 Weitere Klassen

5.3.1 Abstrakte Serviceklasse

Angabe: Typ, Methodenname, Parameter