Entwicklung und Evolution dienstorientierter Anwendungen im Web Engineering

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Martin Nußbaumer

aus Ravensburg

Tag der mündlichen Prüfung:	19. Juli 2007
Erster Gutachter:	Prof. Dr. Wilfried Juling
Zweiter Gutachter:	Prof. Dr. Hartmut Schmeck

Für meine Familie

Vorwort

Die Ergebnisse dieser Arbeit wurden während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Telematik (ITM) an der Universität Karlsruhe (TH) erzielt.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Wilfried Juling, der mir viel freien Raum für meine Forschung gelassen hat. Seine ständige Bereitschaft, auf meine jeweiligen Probleme einzugehen, war für mich eine große Bereicherung. Ebenso bedanke ich mich bei Prof. Dr. Hartmut Schmeck für die Übernahme des Korreferates sowie bei Prof. Dr. Andreas Oberweis und Prof. Dr. Ralf Reussner für die Begleitung meiner mündlichen Prüfung.

Allen meinen Kollegen danke ich für ihre Hilfsbereitschaft und gute Zusammenarbeit in sehr angenehmer Atmosphäre. Dr. Martin Gaedke möchte ich besonders danken, dass er stets Zeit hatte, sich meine Ideen anzuhören und diese kritisch zu hinterfragen. Er stand mir mit seinen zahlreichen Hinweisen stets als wertvoller Mentor während meiner gesamten Zeit am Institut zur Seite. Ein besonderer Dank gilt Florian Allerding, Jan Buck, Patrick Freudenstein, Frederic Majer und Johannes Meinecke sowie meiner Frau Katharina Nußbaumer für das zügige und gründliche Korrekturreferat sowie deren konstruktive Verbesserungsvorschläge und anregende Diskussionen im Vorfeld meiner Arbeit.

Für ihre wertvollen Beiträge als Studienarbeiter und Diplomanden möchte ich auch allen beteiligten Studierenden meinen Dank aussprechen, die mir im Rahmen der Arbeit geholfen haben. Mein Dank geht an Florian Allerding, Pascal Bihler, Marko Böttger, Lei Dai, Jan Fischer, Alex Günter, Nico Hiller, André Janus, Maxim Jochim, Matthias Keller, Sebastian Kneissl, Andreas Kunz, Uwe Mayer, Johannes Meinecke, Johann Costin Mihutoni, Simone Ofer, Thomas Prommer, Yuliya Razumna, Thomas Michael Rudin, Christian Schwall, Andreas Stöckicht, Christopher Thiele, Christian Trefz, Christian Wilhelm und Zhenxiao Ying.

Ich danke auch allen meinen Koautoren für die stets fruchtbare Zusammenarbeit: Vicente Luque Centeno, Markus Dieckmann, Patrick Freudenstein, Martin Gaedke, Oliver Jung, Carlos Delgado Kloos, Frederic Majer, Johannes Meinecke und Emma Tonkin.

Meinen Eltern danke ich für ihre uneingeschränkte Unterstützung, meine persönlichen Ziele jederzeit verfolgen zu können. Nicht zuletzt gilt mein ganz besonderer Dank meiner Frau Katharina, die mir in schwierigen Situationen stets mit Rat und Tat zur Seite stand.

Karlsruhe, September 2007

Martin Nußbaumer

Inhaltsverzeichnis

Vorwort i

Inhaltsverzeichnis i

1 Einleitung 1

1.1 Beitrag 2

1.2 Aufbau der Arbeit 4

2 Problemstellung und Anforderungen 5

2.1 Web Engineering 6

2.2 Separation of Concerns für Web-Anwendungen 10

2.3 Problemcharakteristiken 12

3 Stand der Entwicklung 21

3.1 Web Services 22

3.2 Wissenschaftliche Ansätze im Web Engineering 43

3.3 Zusammenfassung und Bewertung 53

4 Informationsraum-Entwurf 59

4.1 Ausgangspunkt 59

4.2 Konzeption eines Globalen Typ Systems (GTS) 61

4.3 Systematik zum Informationsraum Design 76

4.4 Zusammenfassung 89

5 Das WSLS-Rahmenwerk 93

5.1 Ausgangspunkt 93

5.2 Konfigurieren anstatt Programmieren 98

5.3 Die WSLS-Referenzarchitektur 104

5.4 Anwendungsspezifische Basisdienste 108

5.5 Komponentenevolution 114

5.6 Beispiel eines Informationsraum-Dienst-Elements 124

5.7 Zusammenfassung ... 128
6 Modellierung der Benutzungsschnittstelle ... 129
6.1 Ausgangspunkt ... 129
6.2 Modellierung der Anwendung ... 132
6.3 Anwendungssicht Präsentation ... 148
6.4 Anwendungssicht Navigation ... 157
6.5 Anwendungssicht Dialog ... 164
6.6 Zusammenfassung ... 169
7 Zusammenfassung und Ausblick ... 173
7.1 Ergebnisse der Arbeit ... 173
7.2 Weiterführende Arbeiten ... 178
Anhang ... 181
A Schema der Aspect Markup Language (AML) ... 181
B OWL-Ontologie zum virtuellen Museum ... 183
C Schema der UIRML ... 186
Index ... 190
Literaturverzeichnis ... 199

1 Einleitung

Drei zentrale Faktoren treiben die moderne webbasierte Systementwicklung. Einmal die ständig wachsende Beschleunigung der Entwicklungszyklen, um (weiter-)entwickelte Produkte noch schneller am Markt platzieren zu können (Jarzabek und Pettersson, 2006). Hinzu tritt die zunehmende Komplexität der zu entwickelnden Lösungen, sowohl im Hinblick auf den Umfang, als auch die zu berücksichtigende, aber nicht aktiv zu beeinflussende, umgebende Systemlandschaft. Standen in der Vergangenheit singuläre, hoch spezialisierte Lösungen klar umrissener Teilprobleme im Vordergrund, so hat sich der Blickpunkt mittlerweile stark zu Gunsten offener, interoperabler – *dienstorientierter* – Anwendungen verschoben. Laut einer Gartner-Studie hat es einen geschätzten Anstieg der Lizenzerträge aus Technologien für Anwendungsintegration sowie für Portalsoftware im Jahr 2005 um 7,1 Prozent auf mehr als 8,5 Milliarden US-Dollar gegeben. Bis 2010 wird ein Wachstum des Marktes auf 10,3 Milliarden US-Dollar mit einer kumulativen jährlichen Wachstumsrate von 4,1 Prozent über einen Zeitraum von fünf Jahren prognostiziert (Biscotti und Correia, 2006). Diesem starken Wachstum stehen 80% enttäuschte CIOs von insgesamt 145 befragten Unternehmen gegenüber, die mit der Umsetzung und Integration ihrer Applikationslandschaft unzufrieden sind (Kinikin, Bartels und Harrington, 2004). Die Gründe hierfür liegen zumeist in unflexiblen Informationssystemen, die eine Integration erschweren (Veryard, 2004). Die Standish Group spricht von 95% der Integrationsprojekte, die nicht in der Lage sind ihren Zeit- oder Budgetplan einzuhalten (The Standish Group International, 1994-2005). Der dritte Faktor stellt die Qualität dar. Teilziele wie Robustheit und Flexibilität sind hier als Grundvoraussetzung der Adaptierbarkeit und Weiterentwicklung von besonderer Bedeutung. Durch die zunehmende Verlagerung dieser Anwendungen auf Web-Technologien, gewinnen auch ästhetische Gesichtspunkte beim Entwurf von Benutzungsschnittstellen (einheitliches Look&Feel) zunehmend an Bedeutung.

Durch ihre ständige Weiterentwicklung, bedingt durch den nach wie vor ungebremsten technologischen Fortschritt des Webs (Nussbaumer und Gaedke, 2006), fehlen vor allem *methodische Standards* und *Hilfestellungen* für eine *effiziente* und *effektive* Entwicklung und Evolution dieser Web-Anwendungen.

Effizienz wird hier im Sinne der Bevorzugung von bereits entwickelten Lösungsbausteinen über der Neuentwicklung verstanden, wodurch Produktionszeiten verkürzt und damit Zeit und Kosten gesenkt werden können.

Effektivität beschreibt die Korrelation der konstruierten Lösung mit den Anforderungen eines Kunden oder Auftraggebers. Durch die Wiederverwendung von Lösungsbausteinen, die

bestimmte Muster im Sinne von »best practices« bereitstellen, lassen sich Web-Anwendungen durch das Zusammenschalten dieser Bausteine auch *effektiv* konstruieren.

Die Entwicklung von Web-Anwendungen hinsichtlich der *Effizienz* und *Effektivität* wird verschärft durch die dem Web inhärent gegebenen Eigenschaften:

- *Nichtlinearität von Informationen:* Das Hypertext Paradigma, das dem Web zugrunde liegt, forciert die flexible Vermaschung von Informationen. Einzelne Informationseinheiten sind beliebig miteinander assoziierbar und können daher in beliebigen Strukturen auftauchen. Dies führt zur Problematik des Auffindens von geeigneten Informationen sowie deren gezielten Wiederverwendung. Die Evolution, als die kontinuierliche Änderung von gezielten Einflussgrößen auf eine Web-Anwendung über die Zeit erfordert einen flexiblen Umgang zwischen Informationsanbietern und -konsumenten.
- *Ubiquitärer Zugriff*: Durch den universalen Zugriff auf Web-Anwendungen durch unterschiedlich befähigte Menschen mit unterschiedlichen Geräten, kommt der Benutzungsschnittstelle im Vergleich zu traditionellen Software-Anwendungen eine enorme Bedeutung zu. Ein Beispiel dazu bildet die Barrierefreiheit von Web-Anwendungen. Durch jüngste gesetzliche Regelungen in Industrienationen wie Deutschland, Japan oder die USA werden die bisher eher nachlässig beachteten Richtlinien zur Barrierefreiheit zukünftig für Öffentliche Einrichtungen wie Universitäten rechtlich bindend sein.
- *Dokument- und Endnutzerzentriertheit*: Die ursprüngliche Intention des Webs als dezentrales Informationsmedium, in dem Informationen leicht durch die Endnutzer hinzugefügt werden sollen, hat sich durch seine stetige Weiterentwicklung zu einer Plattform für verteilte Anwendungen gewandelt. Durch die Standardisierung der Web Services hat das altbekannte Architekturmuster der Service-orientierten Architektur neuen Auftrieb bekommen. Trotz des sich vollziehenden Paradigmenwechsels hin zu verteilten, *dienstorientierten* Web-Anwendungen, fußen die grundlegenden Technologien zur Konstruktion von Web-Anwendungen auf grobgranularen Dokumenten, die schlecht wart- und wiederverwendbar sind.

1.1 Beitrag

Die vorliegende Arbeit untersucht eine methodische und systematische Herangehensweise an die Entwicklung von *dienstorientierten Web-Anwendungen* im Web Engineering. Sie stellt die Architektur in den Vordergrund, deren Grundlage *autonome*, *wiederverwendbare* Komponenten in Form von Web Services (W3C-SOA, 2004) darstellen.

Es werden dedizierte Modelle, Methoden und Werkzeuge entlang dieser auf Web-Technologien basierenden Architektur entwickelt. Dabei werden zwei Ziele stringent verfolgt: die Förderung der Wiederverwendung von Komponenten und Diensten, sowie die Unterstützung der Evolution einer Web-Anwendung als der Evolution dieser Komponenten und Dienste.

Damit wird die Basis einer Realisierung von modernen, verteilten und skalierbaren Web-Anwendungen geschaffen. Web-Anwendungen werden nicht mehr von Grund auf neu entwickelt. Sie lassen sich aus bereits vorhandenen Komponenten beliebigen Ursprungs und Ausführungsorts zusammenstellen und, entsprechend den individuellen Anforderungen hinsichtlich unterschiedlicher Aspekte einer Web-Anwendung wie Präsentation oder Navigation, konfigurieren. Die damit verbundenen Vorteile liegen auf der Hand:

- Enorme Verkürzung der Entwicklungszeiten.
- Realisierung qualitativ hochwertiger Komponenten.
- Hohe Flexibilität und Skalierbarkeit der Anwendungen durch Dienste.
- Vereinfachte Wartbarkeit und Evolution.

Kern dieser Herangehensweise stellt die Definition einer Referenzarchitektur dar, die relevante, dem Web inhärent gegebene, Einflussgrößen kapselt und diese als Gegenstand der Wiederverwendung in Form lose gekoppelter Einheiten ansieht. Sie umfasst dabei die folgenden Ziele:

- Einen innovativen Ansatz, der nicht wie im Web Engineering üblich, die Anwendung in den Vordergrund stellt, sondern vielmehr die Architektur. Modelle und Methoden bilden die Anwendung als *Instanz* der Architektur ab und erlauben in Folge dessen eine ideale Unterstützung für die Evolution dieser Anwendungen.
- Methodische Hilfestellungen für die Entwicklung von Modellen und Vorgehen zur Interaktion mit dem Informationsraum von Web-Anwendungen. Hierbei steht der dienstorientierte Zugriff mit Hilfe von *Informationsraum-Diensten* auf das Informationsmodell einer dienstorientierten Web-Anwendung wesentlich im Fokus. Dabei spielt die Verlässlichkeit der Dienstzugangspunkte, die unter dem stetigen Einfluss der Evolution stehen, eine wichtige Rolle.
- Ein Rahmenwerk zur Entwicklung und Verwendung Domänenspezifischer Sprachen (DSL), das den Kommunikationsschwierigkeiten interdisziplinärer Projektteams bei der Umsetzung dienstorientierter Web-Anwendungen begegnet. Hierbei werden die relevanten Aspekte Navigation, Darstellung und Benutzerinteraktion mit Hilfe eines Rahmenwerks zur Verwendung Domänenspezifischer Sprachen definiert.
- Die Definition eines evolutionsorientierten Ansatzes, der das Modell einer Web-Anwendung in einer holistischen Sichtweise beschreibt. Es wird eine Methodik entwickelt, die eine nichtfunktionale Trennung wesentlicher Bestandteile einer Web-Anwendung mit einer fachlichen Verknüpfung durch Komponenten beschreibt.
- Die Erarbeitung dedizierter Lösungsansätze für die effektive Entwicklung und Evolution der grafischen Benutzungsschnittstelle einer Web-Anwendung in Hinblick auf Wiederverwendung existierender Artefakte sowie der Steigerung ihrer Qualität. Dabei wird explizit aufgezeigt, wie Qualitätsbetrachtungen hinsichtlich der Barrierefreiheit gezielt mit der vorgestellten Lösungsarchitektur umgesetzt werden können.

1.2 Aufbau der Arbeit

Im folgenden Kapitel werden die Besonderheiten anhand einer konkreten Problemstellung basierend auf den besonderen Charakteristika für Web-Anwendungen im Allgemeinen und der Dienstorientierung im Besonderen beschrieben. Dabei wird zunächst das Forschungsgebiet Web Engineering, das sich mit der grundlegenden Problematik der Entwicklung von Web-Anwendungen beschäftigt, vorgestellt. Aus den identifizierten Problemstellungen werden schließlich sukzessive die Charakteristika mit den Anforderungen und Kriterien abgeleitet, die an eine Systematik mit Methoden und Modellen zur Lösung dieser Probleme gestellt werden müssen. Das anschließende Kapitel 3 stellt technische Grundlagen sowie dedizierte Ansätze aus dem Umfeld des Web Engineering vor und bewertet sie anhand der dargestellten Anforderungen und Fragestellungen auf ihre Eignung.

Aufbauend auf den technischen Grundlagen wird in Kapitel 4 der Entwurf des Informationsraums auf Basis dedizierter Dienste beschrieben. Um die Evolution dieser Dienste besser unterstützen zu können, wird das Konzept der kanonischen Schnittstellen erarbeitet. Dadurch wird eine explizite Unterstützung der Evolution in sowohl horizontaler (kanonische Erweiterung), als auch vertikaler Richtung (Anwendungsspezifität) erreicht.

Kapitel 5 stellt das WSLS-Rahmenwerk vor, das sich in einer nicht-technischen und einer technischen Facette darstellt. Zum einen wird ein methodisches Vorgehen auf Basis von Domänenspezifischen Sprachen entwickelt, das die Reduktion von Kommunikationsschwierigkeiten durch intensive Einbeziehung ihrer Projektmitglieder anstrebt. Zum anderen wird ein Komponentenmodell basierend auf einer Trennung von fachlichen und konzeptuellen Komponenten eingeführt, um deren Wiederverwendbarkeit und Fähigkeit zur Evolution zu erhöhen. Um eine einwandfreie Orthogonalität unterschiedlicher Aspekte (Separation-Of-Concerns) zu gewährleisten, werden diese mit einem Lebenszyklus ausgestattet, der die Anwendung Aspekt-orientierter Entwurfsmethoden erlaubt. Dazu wird das neue Konzept der *wohldefinierten Verknüpfungspunkte* entwickelt.

Kapitel 6 stellt die Entwicklung der Benutzungsschnittstelle anhand des zuvor eingeführten Komponentenmodells vor. Dabei wird ein innovatives Anwendungsmodell entwickelt, das eine Anwendung als *Instanz* einer Architektur abbildet, die in ständiger Harmonie mit ihren Entwurfsartefakten steht. Dazu wird zunächst der Anwendungskontext eingeführt, der eine Graph-basierte Formalisierung auf Basis der konzeptuellen Komponenten darstellt. Das darauf aufbauende Konzept der Anwendungssicht ermöglicht die Übereinstimmung zwischen Anwendungsinstanz und Entwurfssicht, indem für bestimmte, durchzuführende Aufgaben eines Entwurfs, dedizierte Sichten auf Anwendungskontexte definiert werden. Die Tragfähigkeit des Ansatzes wird durch die Vorstellung dedizierter Modelle und Methodiken für die Aspekte Navigation, Präsentation und Dialog sowie einer Unterstützung für ein proaktives Barrierefreies Design nachgewiesen.

Das Kapitel 7 schließt die Arbeit mit einer Zusammenfassung der Ergebnisse sowie einem Ausblick ab. Mehrere Anhänge ergänzen abschließend die Arbeit mit Zusatzinformationen und Details über die vorgestellten Lösungen.

2 Problemstellung und Anforderungen

Durch Standardisierungsbemühungen im Bereich Web Services haben in jüngster Zeit Architektur-Konzepte wie die Service-orientierte Architektur (SOA) erheblich an Bedeutung gewonnen (Kossmann, Leymann und Taubner, 2004). Die der vorliegenden Arbeit zugrundegelegte Verwendung des Begriffes der Service-orientierten Architektur betont besonders die lose Kopplung zwischen Diensten in einer SOA, die über wohldefinierte Schnittstellen miteinandner kommunizieren können.

> *Definition 2.1* – **Service-orientierte Architektur:** Eine Service-orientierte Architektur (SOA) stellt lose gekoppelte Dienste (services) über eine wohldefinierte, standardisierte Schnittstelle bereit.

Ein wesentliches Ziel einer SOA stellt eine schnelle Reaktion auf Änderungen dar. Dabei stehen flexible interaktionsreiche Web-Anwendungen im Vordergrund, die aus wieder verwendbaren lose gekoppelten Einheiten (Komponenten und Diensten) zusammengebaut werden können. Zusätzlich existieren Forderungen zum flexiblen Austausch von Inhalten (*Syndikation*), wodurch diese Web-Anwendungen gleichzeitig als *Diensterbringer* und *Dienstnehmer* fungieren. In diesem Zusammenhang finden sich aktuelle Klassifikationsmerkmale für solche Web-Anwendungen beispielsweise in Form der so genannten »5G-Portale« (Phifer, 2006a). Diese Portale stellen ihre Funktion, ausgeprägt als Web-Anwendung, aus einer Menge von Diensten zur Verfügung. Die daraus erwachsenden Probleme an die Entwicklung dieser Web-Anwendungen umfassen Problemfelder wie:

- Anwendungsstrukturierung durch Komponenten,
- Austauschbarkeit von Diensten,
- Komponierbarkeit von Diensten,
- Wiederverwendbarkeit von Diensten.

2.1 Web Engineering

Eine Web-Anwendung stellt ein komplexes Softwaresystem dar. Es bedarf zu ihrer Entwicklung einer methodischen Herangehensweise. Ein solches Vorgehen, das auf den Grundlagen des Software Engineering fußt, umfasst die »Anwendung systematischer, disziplinierter und quantifizierbarer Ansätze für die kosteneffizienten Entwicklung und Evolution von qualitativ hochwertigen Anwendungen im World Wide Web« (Gaedke und Graef, 2000). Es wird aufgrund seines direkten Bezugs zu den grundlegend eingesetzten Web-Technologien und Standardisierungsbemühungen des World Wide Web auch als »Web Engineering« bezeichnet (Deshpande, Murugesan, Ginige, Hansen et al., 2002). Mit dem Web Engineering setzt man einen planvollen und wieder verwendbaren Entwicklungsprozess ein, um die kontinuierliche Weiterentwicklung von Web-Anwendungen zu unterstützen. Neben der Kostenreduktion und Risikominderung bei der Neuentwicklung von Web-Anwendungen sowie deren Wartung verspricht man sich darüber hinaus Vorteile in Bezug auf die Qualität der entwickelten Anwendung (G. Kappel, B. Pröll, S. Reich und Retschitzegger, 2003).

Basierend auf der eingangs gestellten Definition der SOA wird nun eine Präzisierung des Begriffs der *dienstorientierten* Web-Anwendung in Form der nachstehenden Definitionen gegeben. Sie fasst die unterschiedlichen Aspekte eines Dienstes hinsichtlich seiner Nutzer zusammen und schließt dabei die im Falle von Web-Anwendungen exponierte Betrachtung der Darstellung explizit mit ein.

> *Definition 2.2* – **Dienstorientierte Anwendung im Web Engineering:** Eine dienstorientierte Anwendung im Web Engineering stellt die Bündelung von Funktionen in Form von Diensten, für einen Dienstnehmer basierend auf Web-Technologien dar. Sie sorgt für eine *adäquate Schnittstelle* zwischen dem Diensterbringer und dem Dienstnehmer.

Dabei kommt der *Schnittstelle* eine exponierte Stellung zu. Dies liegt einerseits an ihrer enormen Wichtigkeit im Web Engineering; hier tritt ein menschlicher Benutzer mit einem Dienst der Web-Anwendung über eine grafische Schnittstelle in Kontakt. Andererseits findet aus Sicht der Anwendung die Kommunikation mit diesen Diensten mittels standardisierter Web-Protokolle über eine wohldefinierte Schnittstelle statt.

> *Definition 2.3* – **Adäquate Schnittstelle**: Eine adäquate Schnittstelle berücksichtigt die inhärenten Besonderheiten eines *technischen* oder *menschlichen* Dienstnehmers. Dazu gehört bei technischen Dienstnehmern beispielsweise die Verwendung von webbasierten Nachrichten- und Transportprotokollen basierend auf dem Client-Server Modell. Menschliche Dienstnehmer hingegen kommunizieren über eine webbasierte Benutzungsschnittstelle, die ihren Bedürfnissen und Fähigkeiten entsprechend angepasst ist.

Eine dienstorientierte Web-Anwendung zeichnet sich durch eine hohe Dynamik hinsichtlich der Interaktion und Darstellung bezüglich ihrer Inhalte und dem zugrunde liegenden *Informationsraum* aus.

> *Definition 2.4* – **Informationsraum**: Der Informationsraum einer Web-Anwendung stellt die Fülle an Informationen und Inhalten dar, die einer Web-Anwendung zur Darstellung, Bereitstellung, Berechnung und Modifikation zur Verfügung stehen.

Bedingt durch den rasanten technologischen Fortschritt sowie sich ändernden Anforderungen ist der Betrieb einer Web-Anwendung und ihres angeschlossenen Informationsraums erheblich durch Anpassungen und Neuentwicklungen geprägt. So sind Web-Anwendungen beispielsweise aufgrund ihrer speziellen Anforderungen an die Benutzungsschnittstelle in besonderem Maße *Trends* unterworfen, was häufig teure Anpassungen nach sich zieht (G. Kappel, B. Pröll, S. Reich und Retschitzegger, 2003). In diesem Zusammenhang spricht man auch vom Phänomen der *Evolution* von Web-Anwendungen.

> *Definition 2.5* – **Evolution von Web-Anwendungen**: Die Evolution einer Web-Anwendung beschreibt die hohe Änderungsdynamik, denen sie nach ihrer initialen Implementierung ausgesetzt ist. Diese Dynamik wirkt sich sowohl auf den Informationsraum, als auch auf die Struktur der Anwendung aus.

2.1.1 Abgrenzung zum Software Engineering

Obwohl viele Gemeinsamkeiten zwischen dem Software Engineering und dem Web Engineering existieren, finden sich doch markante Unterschiede (Deshpande et al., 2002), die durch die speziellen Charakteristika von Web-Anwendungen hervorgerufen werden. Web-Anwendungen können abhängig von ihrem Verhalten in unterschiedliche Kategorien eingestuft werden. So unterscheidet Powell (Powell, Jones und Cutts, 1998) unterschiedliche Arten der Komplexität von Anwendungen beginnend bei dokumentzentrierten Ansätzen bis hin zu dynamisch generierten Anwendungen. Für diese unterschiedlichen Komplexitätsstufen finden sich entsprechend unterschiedliche Charakteristika für Web-Anwendungen. Deren Spektrum umfasst dabei nach (G. Kappel, B. Pröll, S. Reich und Retschitzegger, 2003) Einflussgrößen bezogen auf *Inhalt*, *Präsentation* und *Navigation*. In interaktionsreicheren Web-Anwendungen finden sich darüber hinaus Fragestellungen zum *Dialog* zwischen einem Nutzer und der Anwendung mittels Formularen (Gaedke und Nussbaumer, 2002). Betrachtet man stärker dynamische Anwendungen, finden sich Einflussgrößen bezogen auf die Verarbeitung von Informationen in Form von *Verarbeitungsprozessen*. Die in den letzten Jahren aufkeimende Dienstorientierung führt zunehmend zu Web-Anwendungen, die verstärkt Aspekte wie *Kommunikation* und *Geschäftsprozesse* z.B. in Form von Portalen zusammenführen.

Die dedizierte Betrachtungsweise der aufgeführten Aspekte führte zur kontinuierlichen Anpassung und Weiterentwicklung bestehender Modellierungsansätze und Methodiken aus dem Software Engineering. Während traditionelle Softwareanwendungen stärker zur Tren-

nung in die Benutzungsschnittstelle und zugehöriger Anwendungslogik tendieren, findet sich im Web Engineering, bedingt durch die Vielzahl an Einflussgrößen, eine ausgeprägtere Tendenz zur dedizierten Auftrennung in unterschiedliche Aspekte. Das dahinterliegende Prinzip wird *Separation of Concerns* genannt und bildet ein für das Web Engineering äußerst wichtiges Merkmal (C. Kerer und Kirda, 2004).

Die Modellierung von Anwendungen im Software Engineering kann, wie in Abbildung 2-1 angelehnt an (Schwinger und Koch, 2003) dargestellt, entlang der drei orthogonalen Dimensionen *Ebenen*, *Aspekte* und *Phasen* vorgenommen werden. Die Dimension *Aspekte* trennt dabei die Innen- bzw. Außensicht einer Anwendung, also das »Was« und das »Wie«. Im Software Engineering finden sich hier üblicherweise die Anwendungslogik und die Benutzungsschnittstelle. Die Dimension *Ebene* umfasst die Informationsstruktur (Objekte, Attribute, Beziehungen) und das funktionale Verhalten (Funktionen, Methoden) einer Anwendung. Die dritte Dimension *Phasen* umfasst die einzelnen Schritte eines zugrunde gelegten Entwicklungsprozesses. Nach (Balzert, 1996) finden sich hier üblicherweise Phasen des Kernprozesses: Planung, Analyse, Entwurf, Programmierung und Validierung.

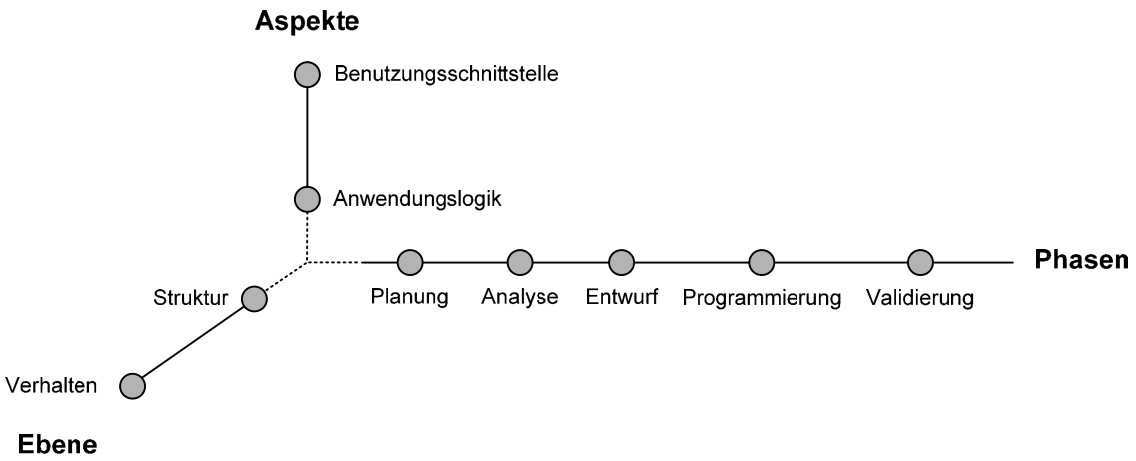


Abbildung 2-1: Anforderungen an die Modellierung von Softwareanwendungen

Ziele des Software Engineering, die gleichermaßen auch für das Web Engineering gelten, sind unter anderem die detaillierte Spezifikation, die reduzierte Komplexität, die Dokumentation der Entwurfsentscheidungen, die vereinfachte und lesbare Darstellung sowie die Visualisierung der relevanten Aspekte einer Anwendung. Ein wesentlich zu unterstützendes Merkmal bei der Entwicklung von Web-Anwendungen bildet die Berücksichtigung der Evolution.

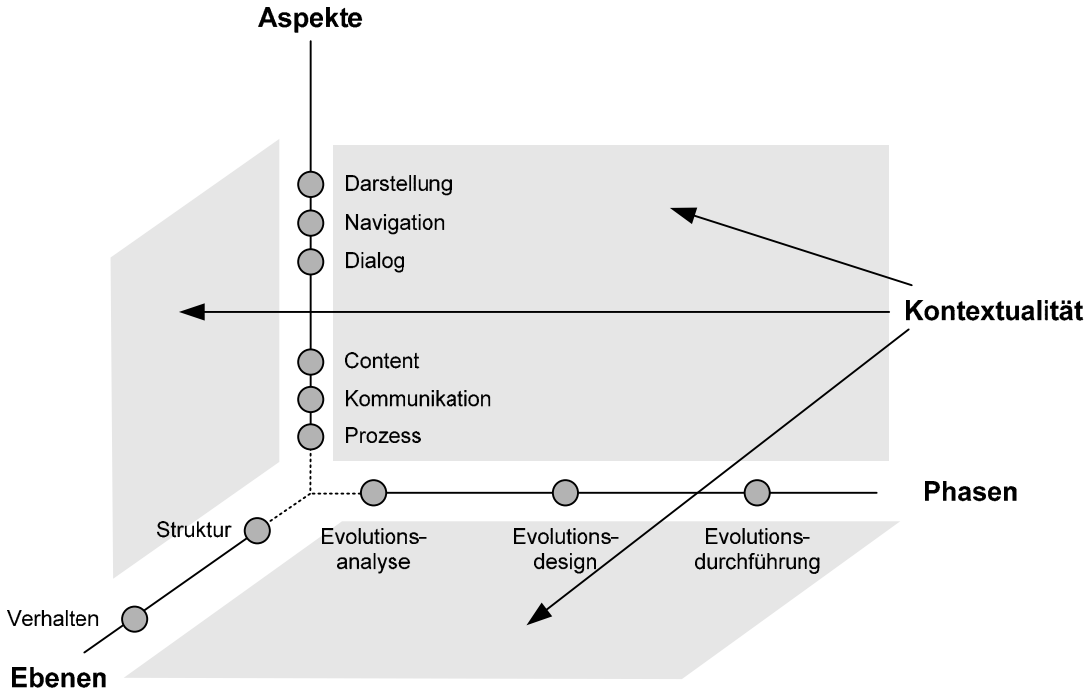


Abbildung 2-2: Anforderungen an die Modellierung von Web-Anwendungen basierend auf dem WebComposition Prozessmodell

Abbildung 2-2 stellt eine Einordnung der unterschiedlichen relevanten Einflussgrößen für das Web Engineering vor. Die Phasen der Entwicklung von Web-Anwendungen werden auf das evolutionsorientierte WebComposition Prozess Modell abgebildet (Gaedke, 2000). Dieses beschreibt eine systematische Entwicklung von Web-Anwendungen auf Basis eines offenen dreistufigen Vorgehensmodells mittels der Phasen: *Evolutionsanalyse*, *Evolutionsdesign* und *Evolutionsdurchführung*. Von besonderer Bedeutung ist hierbei die Berücksichtigung der Evolution in Hinblick auf Wiederverwendung. Die zentrale Bedeutung der Evolution bildet einen wesentlichen Unterschied zum Software Engineering (Gellersen und Gaedke, 1998). Ein Beispiel hierfür stellt die *Wikipedia* (Jimmy Wales, 2004-2006) Web-Anwendung dar. Sie wird durch ihre Nutzer kontinuierlich weiterentwickelt und ist daher nie abgeschlossen; sie befindet sich in einem Zustand der kontinuierlichen Weiterentwicklung – also der permanenten Evolution.

Eine weitere Besonderheit im Vergleich zum Software Engineering stellt eine Erweiterung der Dimension Aspekte dar. Aspekte von zentraler Bedeutung im Web Engineering bilden die Benutzungsschnittstelle-relevanten Einflussgrößen. Nach (Gaedke, 2000) sind dies: Präsentation, Navigation und Dialog, sowie die den Informationsraum beschreibenden Aspekte *Inhalt*, *Prozess und Kommunikation*. Letztere haben Einfluss auf die Informationsstruktur und die Anwendungslogik einer Web-Anwendung. Gerade hinsichtlich der Dienstorientierung bestimmen diese Aspekte hauptsächlich Aufbau und Inhalt einer Web-Anwendung – sie haben auf einen Benutzer eine eher mittelbare Auswirkung. Eine unmittelbare Wirkung auf einen Benutzer hingegen wohnt den Aspekten der Benutzungsschnittstelle inne. Die Navigation beschreibt die Verknüpfung einzelner Knoten des Inhalt mittels Verweisen, während sich die Präsentation mit der Organisation und Darstellung der Information befasst. Der Aspekt Dialog findet sich in Web-Anwendungen, die durch direkte Interaktion eines Benutzers mit der Web-Anwendung das Ziel der Manipulation des Informationsraums verfolgen.

Das Prinzip dieser Trennung ermöglicht die Modellevolution, die Definition aufeinander aufbauender Modelle sowie unterschiedliche Modellierungszielsetzungen innerhalb der unterschiedlichen Aspekte. Es trägt somit auch der Multidisziplinarität Rechnung, die Web-Anwendungen innewohnen (Powell, Jones und Cutts, 1998). Demnach kann es unterschiedliche Navigationsstrukturen auf Basis desselben Inhalts oder auch unterschiedliche Präsentationsmodelle auf Basis desselben Navigationsmodells geben. Umgekehrt sind Hypertext Muster (Hypermedia Design Pattern) auf unterschiedlichem Inhalt anwendbar. In vielen Szenarien spielt die Interaktion mit dem Inhalt einer Web-Anwendung durch Dialoge eine wichtige Rolle. Inhalt kann wiederum für unterschiedliche Medien, Geräte und Nutzer oder allgemeiner in unterschiedlichen *Anwendungskontexten* präsentiert werden.

2.2 Separation of Concerns für Web-Anwendungen

Systematisiert man diesen Vorgang weiter, so schlägt sich das im grundlegenden Prinzip des »Separation of Concern« nieder (Dijkstra, 1982b). Hierbei wird eine weitestgehende Trennung in unterschiedliche, voneinander unabhängige Einflussgrößen angestrebt, wodurch grundsätzliche Eigenschaften bezogen auf Entwicklung, Wiederverwendung, Wartbarkeit oder Testbarkeit durch diese Komplexitätsreduktion besser unterstützt werden können. Diese Trennung gewinnt gerade in Hinblick auf die besondere Rolle spezifischer Problemstellungen wie Ästhetik und Berücksichtigung kognitiver Überlastung an Bedeutung für das Web Engineering (Nanard und Nanard, 1995).

Der Anteil von Struktur und Verhalten, der in entsprechenden Einflussgrößen berücksichtigt werden muss, ist abhängig von der Art und Weise der zu entwickelnden Web-Anwendung (Powell, Jones und Cutts, 1998). Statische Web-Anwendungen erfordern tendenziell weniger Aufwand bei der Verhaltensmodellierung. Im Gegensatz muss bei der Konzeption von hochinteraktiven Web-Anwendungen verstärkt das Augenmerk auf die Verhaltensmodellierung gelegt werden.

Eine zusätzliche Modellierungsdimension, die bei der Entwicklung von Web-Anwendungen Berücksichtigung finden muss, ist die *Kontextualität*. Sie bezeichnet die Anpassung einer Web-Anwendung an einen aktuellen *Nutzungskontext*. Dabei kann ein Nutzungskontext diverse Aspekte einer Web-Anwendung umspannen.

Der Nutzungskontext setzt die Dimension Ebene und Aspekte, aber auch die Entwicklungsphasen einer Web-Anwendung in Beziehung zu *laufzeitbedingten Größen.* Solche Größen können beispielsweise den aktuellen Nutzer einer Web-Anwendung, die Zeit, die Lokation oder unterschiedliche Browser umfassen. Dabei ist die Aufgabe einen Kontext zu einer Anwendung zuzuordnen oder wie im Fall einer dienstorientierten Web-Anwendung, die Zuordnung zu einem bestimmten durch die Web-Anwendung erbrachten Dienst, nicht trivial.

Das zuvor eingeführte Forschungsgebiet Web Engineering hat sich als eigenständige Disziplin etabliert und beschäftigt sich mit der systematischen Lösung eben dargestellter Problemstellungen.

Um der Evolution zu begegnen, bedarf es geeigneter Verfahren und Modelle, die sowohl die technologisch-funktionale Dimension (z.B. Web Services und Komponenten) als auch konzeptionell-abstrakte Aspekte berücksichtigen. Die gerade hinsichtlich der Interaktion stark ausgeprägten Web-Anwendungen umfassen hierbei Problemstellungen zu:

- der strukturierten *Navigation* über den angeschlossenen Informationsraum,
- einer adäquaten *Präsentation* von Informationen
- sowie dem *Dialog* als der interaktiven Manipulation des Informationsraums durch einen Menschen.

Dabei muss die Tatsache Berücksichtigung finden, dass Web-Anwendungen in zunehmendem Maße von Menschen mit nicht technischem Hintergrund entwickelt werden.

Eine weitere wichtige Anforderung bei der Entwicklung von Web-Anwendungen bildet die Wiederverwendung von Web Services. Das Zusammenführen von Web Services zu *dienstorientierten Web-Anwendungen* nach dem Baukastenprinzip wird dabei neben technischen und kommerziellen Belangen auch vermehrt durch soziologische Aspekte geprägt. So existieren neben Forderungen nach Sicherheit und Robustheit dienstorientierter Web-Anwendungen auch Einflussfaktoren wie das Vermeiden kognitiver Überlast oder der barrierefreie Zugang für Menschen mit Behinderungen.

Darüber hinaus setzen aktuelle Trends – zusammengefasst unter dem Schlagwort Web 2.0 – kontinuierlich neue Maßstäbe an die Gestaltung der Benutzungsschnittstelle und damit Herausforderungen an die Gesamtarchitektur. Während die zugrunde liegende Web-Technologie durch den kontinuierlichen Fortschritt einem rasanten Wandel unterworfen ist, fehlt es an methodischen Standards, welche eine systematische Entwicklung solch dienstorientierter Web-Anwendungen ermöglichen. Dedizierte Modelle und Methodiken für vereinzelte Aspekte werden in der Literatur zwar diskutiert (siehe Abschnitt 3.2), sind aber in aller Regel wenig oder gar nicht auf eine komponenten- und dienstorientierte Sichtweise vorbereitet. Dadurch lassen sich solch hochgradig interaktive und dynamische Web-Anwendungen nur mit hohem Aufwand realisieren.

Im nächsten Abschnitt werden konkrete Anforderungen abgeleitet, welche die Brauchbarkeit aktueller Methodiken und Entwicklungsmodelle im Web Engineering für Entwicklung und Evolution einer dienstorientierten Web-Anwendung im Sinne der Definition 2.2 bilden. Dabei werden grundlegende Problemstellungen erörtert, die durch den Problembereich der *Dienstorientierung* im Sinne einer Service-orientierten Architektur (SOA) und ihrer Umsetzung in Form von dienstorientierter Web-Anwendungen entstehen. Ein wichtiger Fokus liegt dabei auf dem Aspekt der Kommunikation. Dieser wird in zwei Facetten betrachtet. Einerseits in Form einer nutzerzentrierten Kommunikation mittels der Benutzungsschnittstelle einer Web-Anwendung und den zugehörigen Einflussgrößen *Inhalt*, *Präsentation* und *Navigation*. Andererseits durch die Fähigkeit zur Anwendungskomposition durch lose gekoppelte Bausteine, die mit Hilfe standardisierter Protokolle miteinander kommunizieren.

2.3 Problemcharakteristiken

Im Folgenden werden die Anforderungen ermittelt, die nötig sind, um eine methodische Entwicklung und systematische Evolution einer dienstorientierten Web-Anwendung zu gewährleisten. Dabei werden insgesamt drei zentrale Bereiche identifiziert, die die einzelnen Problembereiche einer dienstorientierten Web-Anwendung beschreiben:

- Informationsraum
- Referenzarchitektur
- Benutzungsschnittstelle

Im Folgenden werden diese Problembereiche spezifiziert und detailliert ausgeführt, welche Anforderungen zur Lösung dieser Probleme an eine dienstorientierte Web-Anwendung gestellt werden müssen. Dazu werden basierend in der entsprechenden Problemcharakteristik Haupt- und Unterkriterien beschrieben, die sich als Quintessenz von Problemen und Anforderungen in der einschlägigen Literatur herauskristallisiert haben.

2.3.1 Problemcharakteristik: Informationsraum

Die Wichtigkeit von Informationen und deren Verfügbarkeit in unterschiedlichen Anwendungen geht Hand in Hand mit dem aufkeimenden Wunsch nach Service-orientierten Architekturen. Gerade Forderungen nach Aktualität von Informationen, ihrer Austauschbarkeit (Syndikation), der Integration von Informationen (EII, *Enterprise Information Integration*) oder gar Teilen einer Anwendung (EAI, *Enterprise Application Integration*) werden durch das Anbieten einer Dienstschnittstelle unterstützt. Durch den Paradigmenwechsel hin zu einer Struktur von lose gekoppelten Diensten lässt sich auch die Ableitung qualitativer Parameter zur Beschreibung, des Monitoring und der Verwaltung von ganzen Dienstlandschaften effektiv unterstützen (Gaedke, Meinecke und Nussbaumer, 2005). Eine dedizierte Anbindung des Informationsraums einer Web-Anwendung durch Dienste bildet somit eine zentrale Anforderung zur Realisierung dienstorientierter Web-Anwendungen.

Anforderungsprofil Dienstorientierung

Die Dienstorientierung verspricht, die inner- und zwischenbetriebliche Integration von heterogenen Anwendungssystemen zu vereinfachen. So prognostiziert die Aberdeen Group den 2000 führenden Unternehmen über die nächsten fünf Jahre Kosteneinsparungen von bis zu 53 Mrd. USD bei gleichzeitig höherer Leistungsfähigkeit und Flexibilität (Mougayar, 2005). Die Gartner Group rechnet damit, dass bis 2009 über 80% der neu entwickelten Anwendungen auf Diensten basieren (Hayward, 2005).

Lose Kopplung

Änderungen in eng gekoppelten Systemen, beispielsweise aufgrund von Fehlern oder Anforderungsänderungen, ziehen häufig auch Änderungen in allen angekoppelten Komponenten und Diensten nach sich. Das ist insbesondere in verteilten Systemen wie Web-Anwendungen ein zeit- und kostenintensiver Vorgang. Eine Web-Anwendung aus lose gekoppelten, logisch von einander unabhängigen Architekturkomponenten (Fritz, 2004), verspricht daher eine flexible Anwendungskonstruktion, bei gleichzeitiger Senkung der Entwicklungskosten (Richter, Haller und Schrey, 2005).

Informationsföderation

Häufige Probleme (bei der Integration) stellen unter anderem Plattformabhängigkeit, unterschiedliche Datenmodelle oder auch Prozessabhängigkeit dar (Barkmeyer, Barnard Feeney, Denno, Flater et al., 2003; Kossmann, Leymann und Taubner, 2004). Der homogene Zugriff auf den Informationsraum einer Web-Anwendung durch Web Services stellt somit eine wichtige Anforderung für eine dienstorientierte Web-Anwendung dar. Dadurch werden Konzepte der Informationsföderation (Content Syndication), wie sie nach (Gootzit und Phifer, 2003) für Portale der 4. Generation gefordert werden, ermöglicht.

2.3.2 Problemcharakteristik: Wiederverwendung und Evolution

Die Konstruktion von Web-Anwendungen aus Komponenten offeriert gerade in Hinsicht auf die Realisierung in Form einer Service-orientierten Architektur viele Vorteile und bildet eine grundsätzliche Anforderung (Phifer, 2006b). Durch die Verwendung von Komponenten – in Form abgeschlossener Bausteine – kann das Prinzip der losen Kopplung – vom Dienst bis in die Anwendung – ideal unterstützt werden. Die Wiederverwendung von Anwendungsteilen wird durch Komponenten gefördert und hilft die Kosten für die Entwicklung von Web-Anwendungen zu senken. Gleichzeitig gestattet ein Einsatz von Komponenten auch die Verwendung wiederverwendungsorientierter Entwicklungsmodelle und begünstigt so die Verkürzung von Entwicklungszyklen. Eine wichtige Anforderung beruht dabei auf einer Referenzarchitektur, die diese wichtigen Eigenschaften in einem geeigneten Modell abstrahiert und eine effiziente Umsetzung erlaubt.

Anforderungsprofil Wiederverwendung

Wiederverwendbarkeit stellt eine der zentralen Anforderungen für die systematische Entwicklung von Web-Anwendungen dar. So ist die Konstruktion von Web-Anwendungen auch heutzutage noch zumeist geprägt durch ad hoc Entwicklungsmodelle, die eine planvolle Wiederwendung erschweren. Wenn Wiederverwendung betrieben wird, dann zumeist auf dem grobgranularen, dokumentenzentrischen Web-Entwicklungsmodell durch eine »Copy&Paste«-Strategie von Dokumenten oder Teilen eines Dokumentes. Es finden sich Kopierraten zwischen 17% und 67% bei sowohl neu entwickelten als auch in der Wartung befindlichen Web-Anwendungen (Rajapakse und Jarzabek, 2005). Diese Art der Wiederverwendung ist jedoch mit extremem Aufwand in Bezug auf Wartung und Weiterentwicklung verbunden, da sich Änderungen nicht an *einer Stelle* vornehmen lassen, sondern in *allen* kopierten Deri-

vaten vorgenommen werden müssen. Dieser Aufwand wirkt sich negativ auf den zeitlichen Verlauf von Entwicklungs- oder Wartungsarbeiten aus, wodurch zusätzliche Kosten entstehen.

Eine effiziente Wiederverwendungsstrategie ist daher zwingend erforderlich, um solche Nachteile zu eliminieren und Zeit und Kosten zu sparen.

Komponentenorientierung

Die Verwendung von Komponenten fördert das effiziente Bauen von Web-Anwendungen nach dem Baukastenprinzip. Zusätzlich erlaubt ein komponentenbasierter Ansatz die selektive Anwendung unterschiedlicher Methodiken wie bspw. Die Durchführung von Unit-Tests oder die Auftrennung in funktionale Aspekte (funktionale Dekomposition); deren Abgeschlossenheit und Verteilbarkeit (C. Szyperski, 1997) fördert die Wiederverwendung. Hinsichtlich des »Separation of Concerns«-Prinzips ermöglicht die Verwendung von Komponenten, etwa in für das Web Engineering wichtige Einflussgrößen wie Inhalt, Darstellung, Dialog und Navigation eine sowohl effiziente wie effektive Möglichkeit zur Entwicklung und Evolution von Web-Anwendungen.

Konfigurierbare Konvertibilität

Austauschbare Komponenten erhöhen die *Anpassbarkeit* einer Anwendung auf einen – sich möglicherweise über die Zeit ändernden – Problembereich. Dies gilt vor allem dann, wenn die Problemdomäne, wie bei Web-Anwendungen üblich, häufigen Änderungen unterworfen ist. In engem Zusammenhang mit der Konvertibilität[1] – dem Austauschen oder Substituieren einer Komponente – steht die Fähigkeit zur *Konfiguration* (Dart, 2001). Diese erhöht die Wiederverwendbarkeit von Komponenten, da Einzelteile besser an gegebene Situationen anpassbar sind (Barkmeyer et al., 2003). Das Konfigurationsmanagement identifiziert die »Kontrolle der Evolution komplexer Systeme« als einen wesentlich zu unterstützenden Aspekt (Estublier, 2000).

Die Unterstützung der Evolution durch *Konfigurierbare Konvertibilität* stellt daher eine Anforderung an ein Komponentenunterstützungsmodell dar.

Anforderungsprofil Evolutionsfähigkeit

Mit der zunehmenden Dienstorientierung werden Eigenschaften wie Aktualität und der direkte Zugriff auf Funktionen zur Interaktion (Modifikation) von Daten durch Dienste stetig wichtiger. Getrieben durch einerseits den technologischen Fortschritt und andererseits durch Änderungen von Anforderungen, sei es durch äußere Einflüsse wie Moden und Trends oder durch unvollständige Anforderungsanalysen, stellt die Fähigkeit zur Evolution eine zentrale Anforderung an dienstorientierte Web-Anwendungen dar.

[1] Der Begriff der Konvertibilität kommt ursprünglich aus dem Finanzwesen und bezeichnet die Souveränität von Staaten ihre Währungen beliebig gegen andere auszutauschen.

Der zentrale Aspekt der Evolution liegt hierbei in der Fähigkeit eine Web-Anwendung an Änderungen (inhaltlicher, fachlicher oder struktureller Art) in einer Organisation oder einem Unternehmen anzupassen.

Dazu bedarf es geeigneter Methoden zur dynamischen Konstruktion und Evolution dieser Web-Anwendungen. *Computer Aided Web Engineering* (CAWE) wird in (Semia Sonia Selmi, Naoufel Kraiem und Ghezala, 2005) als Voraussetzung für eine bestmögliche Abdeckung des Entwicklungsprozesses für Web-Anwendungen herausgestellt. Dies ist gerade in Hinblick auf die Evolution von Web-Anwendungen wichtig, da Änderungen zumeist alle Phasen des Entwicklungsprozesses betreffen. Um einen medialen Bruch zwischen Konzeption, Entwurf, Konstruktion und Evolution einer Web-Anwendung zu vermeiden, erfordert es daher konsequenterweise geeigneter Modelle und Methoden, die eine Unterstützung durch Werkzeuge in den unterschiedlichen Phasen des Entwicklungsprozesses unterstützen.

Referenzarchitektur

In einer Umfrage von mehr als 4000 Führungskräften weltweit kommt eine Studie der Economist Intelligence Unit (Borzo, 2005) zum Schluss, dass ein wesentlicher Erfolgsfaktor für Unternehmen in den nächsten Jahren die Innovation ihrer Geschäftsmodelle darstellt. Dabei stellt sich die Fähigkeit, diese Innovationen *schnell* innerhalb der eigenen betrieblichen Informationssysteme umzusetzen, als erfolgsentscheidend dar (Kagermann und Österle, 2006). Aufgrund stagnierender finanzieller Mittel und gleichzeitig zunemehnend kürzerer Innovations- und Anpassungszyklen werden harte Anforderungen an die Adaptionsfähigkeit der zugrunde liegenden Informationssysteme gestellt (Zarnekow, Brenner und Pilgram, 2005). Um eine effiziente Umsetzung zu erreichen, wird eine agilen Plattform erfordert, die eine flexible Anpassung im Sinne eines »composition frameworks« (Phifer, 2006a) zur Laufzeit unterstützt.

Für eine erfolgreiche Realisierung dienstorientierter Web-Anwendungen ist daher die Existenz einer Referenzarchitektur unabdingbar und stellt eine zentrale Anforderung dar.

Produktionszyklen

Die Aktualität einer Web-Anwendung und ihrer Benutzungsschnittstelle im Besonderen bedarf einer kontinuierlichen Überprüfung und Abstimmung hinsichtlich der Effektivität bei der Umsetzung von Anforderungen. Gerade Web-Anwendungen sind einer hohen Änderungsdynamik ausgesetzt und benötigen daher geeignete Unterstützung, um ihrer eigenen Evolution gerecht zu werden. (Ginige und Murugesan, 2001) sprechen in diesem Zusammenhang auch von einer *Web Krise*. So kann gerade im Bereich des E-Commerce die Aktualität der Benutzungsschnittstelle den erfolgskritischen Faktor darstellen (Remus, 2006). Das liegt in der Art und Weise begründet, wie Web-Anwendungen von ihren Nutzern wahrgenommen werden. Eine Festlegung und Beschreibung von Anforderungen sollte daher idealerweise durch Prototypen, lauffähige Programme oder Reviews ergänzt werden (Constantine und Lockwood, 2001). Eine wiederverwendungsorientierte Konstruktion von Web-Anwendungen nach dem Baukastenprinzip ermöglicht ein solches Vorgehen aufgrund kurzer Produktionszyklen (McDonald und Welland, 2004) und erlaubt schnelle Reaktionszyklen durch Kunden- und oder Nutzerrückmeldungen. Um die effiziente Produktion solcher Prototypen zu gestatten, bedarf es geeigneter Werkzeuge, Methodiken und einer Referenzarchitektur, die eine Konstruktion solcher Anwendungen in Sinne eines *Joint Application Development* (Livesey und Guinane, 1996) fördern.

Der Konstruktionsrahmen für eine dienstorientierte Web-Anwendung muss kurze Produktionszyklen fördern.

Entwicklungsprozess

Ein Entwicklungsprozess muss sich mit Problemfeldern wie z.B. dem Umgang mit unklaren bzw. sich ständig ändernden Anforderungen oder der Forderung nach Verkürzung der Produktionszyklen auseinander setzen. Während konventionelle Prozessmodelle, wie das Wasserfallmodell (Royce, 1970) stark dokumentengetrieben und in einem linearen Prozess ablaufen, legen beispielsweise agile Vorgehensmodelle großen Wert auf eine möglichst frühe und häufige Auslieferung lauffähiger Software. Ein flexibler Entwicklungsprozess verfolgt nicht eine starre, vorhersagbare Kette von Aktivitäten, sondern agiert flexibel und situationsbedingt (Selmi, Kraiem und Ghézala, 2006). Das bedeutet, dass der Entwicklungsprozess möglichst parallelisierbar gehalten werden soll und eine flexible Zuordnung zu dedizierten Rollen erlaubt. Im Zusammenhang mit einer Unterstützungsplattform gestattet er somit ein reversibles[2] Verhalten, also die verlustfreie Abbildung von Entwurfsartefakten auf das Implementierungsmodell. Das bedeutet, die Web-Anwendung und ihre korrespondierenden Entwurfsmodelle streben nach einem harmonischen Gleichgewicht.

Der Konstruktionsrahmen für eine dienstorientierte Web-Anwendung muss einen flexiblen Entwicklungsprozess fördern.

2.3.3 Problemcharakteristik: Benutzungsschnittstelle

Die Benutzungsschnittstelle stellt einen Dienstzugangspunkt zwischen menschlichen Benutzern und der Web-Anwendung mit ihren zugrunde liegenden Diensten dar. Sie stellt somit in gewisser Hinsicht eine Integrationsschicht dar, nämlich bezogen auf die Verwendbarkeit von Aufgaben oder Prozessschritten, die durch einen Benutzer – in Form von Diensten – erbracht werden kann. Hierbei müssen Fragestellungen bezogen auf die relevanten Aspekte der Benutzungsschnittstelle betrachtet werden. Im Einzelnen sind dies die *Navigation* über einen dienstorientierten Informationsraum, die adäquate *Darstellung* von Informationen und *Nutzerinteraktion* mit dem dienstorientierten Informationsraum. Die Separierung dieser Aspekte in möglichst orthogonal zu einander stehende Entwurfsentitäten spielt dabei eine zentrale Rolle, da sie Wiederverwendung und Anpassbarkeit einzelner Entwurfsentitäten begünstigt. Der Wirkungsbereich dieser Trennung umfasst modellbehaftete Eigenschaften sowohl bezogen auf technisch funktionale Details einer Implementierungsplattform, als auch nichtfunktionale Eigenschaften wie regionale und kulturelle Unterschiede (Becker und Mottay, 2001)[3].

---

[2] Der Begriff *Reversibilität* geht auf den Entwicklungspsychologen Jean Piaget zurück und beschreibt die Fähigkeit von Kindern, Operationen geistig vollständig durchzuführen und auch wieder umzukehren.

[3] Becker und Mottay stellen fünf Hypothesen auf, die die regionalen und kulturellen Unterschiede für Web-Anwendungen verdeutlichen. Durch Deduktion führen Sie an verschiedenen Beispielen vor, wie sich diese Hypothesen auf die Benutzerschnittstelle von Web-Anwendungen auswirken. Ein Beispiel dafür stellen regionale Unterschiede der Leserichtung (von rechts nach links) für Texte und Dialoge im arabischen Sprachraum dar.

Anforderungsprofil Benutzungsschnittstelle

Die Rolle des menschlichen Kommunikationsnehmers von Web-Anwendungen nimmt in jüngster Zeit deutlich an Wichtigkeit zu. Dies äußert sich durch aufstrebende Trends wie dem Web 2.0 (O'Reilly, 2005) oder durch regulierende Gesetze zur Einhaltung der Barrierefreiheit von Web-Anwendungen (Bundesministerium der Justiz, 2002). So wandelt sich das bisher überwiegend passive genutzte Medium Web zu einer (inter-)aktiven Plattform. Durch den Einsatz von Diensten können Teile der Benutzungsschnittstelle zwischen Web-Anwendungen föderiert werden. Neuartige Anwendungen versetzen den bisherigen Konsumenten von Informationen nun auch in die Lage, eigene Inhalte selbst zu publizieren und erweitern so fortlaufend den Informationsraum des Webs. Auf diese Weise werden hohe Anforderungen an die Interaktion mit der Benutzungsschnittstelle gestellt.

Joe Pine und Jim Gilmore beschreiben in (Pine II und Gilmore, 1999) den zeitlichen Verlauf hin zur Dienstleistung als *erfahrbares Erlebnis*. Dieses Erlebnis (engl. *experience*) wird vornehmlich geprägt von den Vorlieben, Wünschen und Werten der Individuen. Sie prognostizieren die »experience economy«, die durch interaktionsreiche Web-Anwendungen (*Rich Internet Application*, *RIA*) verkörpert wird, in denen die Benutzungsschnittstelle die zentrale Rolle spielen wird.

Um die Entwicklung von Benutzungsschnittstellen in Hinsicht auf Wiederverwendung, Austauschbarkeit oder Anpassbarkeit effizient zu gestalten, bedarf es der dedizierten Unterstützung und separaten Betrachtung der dafür relevanten Aspekte Darstellung, Dialog und Navigation.

Barrierefreies Darstellungsdesign

Eine wichtige Anforderung an die Benutzungsschnittstelle stellt eine *adäquate Darstellung* von Informationen für die Benutzer dar. Dabei umfasst die adäquate Darstellung mehrere unterschiedliche – sich teilweise beeinflussende – Faktoren. Neben Fragestellungen der Ästhetik und Konformität, wie sie bspw. für Unternehmensportale oder öffentliche Einrichtungen wie Universitäten durch sogenannte Corporate Design (CD) Vorgaben existieren, rücken zunehmend Forderungen nach barrierefreier Zugänglichkeit von Informationen in den Vordergrund. Das Spannungsverhältnis zwischen einerseits ästhetisch gelungenen und andererseits *barrierefreien* Web-Anwendungen stellt dabei oft ein großes Problem dar. Der Trend, die Wichtigkeit ästhetischer Merkmale über die Barrierefreiheit zu heben, ist besonders im Bereich kommerzieller Web-Anwendungen ausgeprägt. Dies wird durch Modetrends (Pressman, 2000), denen beispielsweise E-Commerce Anwendungen unterworfen sind, noch verschärft.

Ein solches Vorgehen kann jedoch ernste rechtliche Konsequenzen nach sich ziehen, da mittlerweile viele Industrienationen über entsprechende nationale Regulierungen verfügen (ITAW, 1998; Bundesministerium der Justiz, 2002; Japanese Standards Association, 2004), in denen Sanktionen auf einen Verstoß gegen diese Erlässe explizit mit eingeschlossen werden.

Ist eine Web-Anwendung einer solch hohen Darstellungsdynamik (Präsentations-Evolution) ausgesetzt, bedarf es geeigneter Methoden, um die Zugänglichkeit systematisch zu gewährleisten. Das Missverhältnis zwischen ästhetisch gelungenen Design und barrierefreier Web-Seiten zeigt sich unter anderem in den Ergebnissen einer Studie von Rosson wonach dem Testen der Web-Anwendung für unterschiedliche Browser deutlich mehr Beachtung zukommt (28%), als dem »Accessibility Testing« (5%) (Rosson, Ballin, Rode und Toward, 2005).

Die Schlussfolgerung aus dieser Studie ist zweifältig. Zum einen bedarf es einer guten und möglichst automatisierbaren Unterstützung zum »Accessibility Testing«. Zum anderen darf das Darstellungsdesign die ästhetisch-künstlerische Seite *nicht* vernachlässigen.

Ein unterstützendes Präsentationsdesign muss neben ästhetischen Qualitätsmerkmalen auch den barrierefreien Entwurf von Komponenten unterstützen.

Aspekt Dialog

Die Forderung nach einer einfachen Änderung von Informationen im Web wurde schon früh durch den Erfinder des Web, Tim Berners-Lee, formuliert (Tim Berners-Lee, Cailliau, Luotonen, Nielsen et al., 1994). Das größte Problem bei der Entwicklung von stark interaktiven Web-Anwendungen stellt das Missverhältnis einer breiten Palette an existierenden und grundlegenden Technologien zum Fehlen methodischer Hilfestellungen, Modellen und Werkzeugen dar, wie eines der wohl prominentesten Beispiele, Google Maps, zeigt (Rasmussen, 2005)[4]. Gerade die zunehmende Dienstorientierung benötigt geeignete Maßnahmen, um die Funktionalität von Geschäftsprozessen in Form von Web Services in Web-Anwendungen *effizient* zu integrieren. Dazu bedarf es geeigneter methodischer Hilfestellungen zur effizienten Steigerung der Benutzungsschnittstelle in Hinblick auf Entwurf und Konstruktion von Dialogen, um die entstehende Lücke zwischen dem kontinuierlich technologischen Fortschritt und den Anwendungsanforderungen zu schließen.

Aspekt Navigation

Ein gutes und konsistentes Navigationsdesign stellt eine große Herausforderung an den Entwurf einer Web-Anwendung dar. Navigationsentscheidungen werden dabei maßgeblich vom zugrunde liegenden Informationsraum beeinflusst (Pressman, 2000). In dienstorientierten Web-Anwendungen, deren Informationsraum dem stetigen Wandel unterworfen ist und Navigationsentscheidungen immer dann neu getroffen werden können, wenn neue Dienste hinzugefügt werden, wird das Navigationsdesign dadurch noch zusätzlich erschwert. Eine wichtige Anforderung im Umgang mit Navigation ist die Reduzierung von kognitiver Überlast[5]. Sie entsteht durch komplexe und unübersichtliche Navigationsstrukturen oder dem Fehlen einer zugrunde gelegten konsistenten Navigationslogik. Ein Beispiel dafür ist die unsachgemäße Verwendung des »Rückwärts« Knopfs eines Browsers (Akhilesh und Ramayya, 2002)[6]. Gezielte Linksetzung und eine intelligente Benennung (Jeff, 1987) helfen die kogniti-

[4] In seiner Keynote auf der ICWE 2005 hat Lars Rasmussen (Google Maps, Technical Leader) über die Entwicklung von Google Maps gesprochen und dabei unter anderem auch die Probleme im Umgang mit den neuen Technologien und fehlenden methodischen Standards aufgezeigt.

[5] Die herausragende Stärke von Hypertext Anwendungen – die Nichtlinearität – birgt gleichzeitig auch eine der größten Gefahren bei der Erstellung von Web-Anwendungen: die Desorientierung. Sie bezeichnet die Tendenz eines Nutzers, die Orientierung innerhalb eines Hypertext Dokuments zu verlieren. Unter kognitiver Belastung versteht man den *zusätzlichen* Aufwand und Konzentration, die aufgebracht werden muss, um die Orientierung wieder zu finden.

[6] In statischen Web-Seiten auf Basis einzelner HTML Dokumente stellt »Lost in Hyperspace« in der Regel noch kein technisches Problem, sondern lediglich ein perzeptorisches dar. So speichern gängige User Agents direkt den Navigationsverlauf und unterstützen so eine Rückwärtsnavigation, wenn das Navigationsdesign dieses nicht explizit unterstützt. In dynamischen Web-Anwendungen, kann dies zusätzlich zu erheblichen technischen Problemstellungen führen. Dies wird am Beispiel von ASP.NET deutlich. Dynamische Seiten verwenden das

ve Belastung zu reduzieren und führten über die Zeit zu verschiedenen Mustern, die sich ähnlich den Software Design Pattern (Gamma, Helm, Johnson und Vlissides, 1995) als »best practices« herauskristallisiert haben. Der konsequente Einsatz dieser *Hypermedia Design Pattern* (Lowe, 1999) verspricht eine systematische Navigationsunterstützung in dienstorientierten Web-Anwendungen zu gewährleisten (Akanda und German, 2005).

Der Navigationsaspekt bedarf bei der Entwicklung und Evolution dienstorientierter Web-Anwendungen einer gesonderten, auf Mustern basierenden, Unterstützung.

Konzept des *viewstate (Warren, 2004)*, um die Anwendung wieder aufbauen zu können und auf individuell erzeugte Ereignisse zu reagieren (und zu navigieren). Er wird zwischen einzelnen Serverzyklen mittels der HTTP/POST Methode hin- und hergeschickt. Wird diese Information, wie im Falle von ASP.NET, jedoch zur Navigation verwendet, versagt das von Nutzern gewohnte, auf HTTP/GET basierte Verhalten eines User Agents durch »Rückwärts« Navigation.

3 Stand der Entwicklung

Die Forschung im Bereich der Entwicklung von Web-Anwendungen bietet ein breites Spektrum an Methoden und Modellen, die seit der Einführung des Web diskutiert und kontinuierlich weiterentwickelt wurden. Da die Disziplin Web Engineering noch jung und die Bandbreite an relevanten Aspekten im Web Engineering breit gefächert ist, findet sich auch eine Vielzahl an existierenden Lösungsvorschlägen, Vorgehensmodellen und Methoden. Um einen besseren Überblick über die existierenden Ansätze zu bekommen, werden sie im Folgenden zunächst einer Kategorisierung unterzogen. Anhand dieser Taxonomie werden die jeweils prominentesten Vertreter ihrer Art zusammengefasst und anschließend diskutiert. Die zuvor aufgestellten, allgemeinen Anforderungen an die Entwicklung und Evolution von dienstorientierten Web-Anwendungen werden als Bewertungskriterium der existierenden Methoden herangezogen. Es zeigt sich, dass einzelne Methoden nicht als Lösung für die zugrunde liegende Problematik geeignet sind. Darüber hinaus stellt sich auch heraus, dass eine einfache Kombination eventuell vorhandener Teillösungen ebenfalls nicht zum gewünschten Ziel führen.

Im folgenden Abschnitt 3.1 werden zunächst grundlegende Technologien und Ansätze erläutert, die zur Entwicklung und Evolution von dienstorientierten Anwendungen im Web Engineering als technologische Grundlage dienen. Zum einen sind das Web Services in ihrer Funktion als »Motor« für Service-orientierte Architekturen und dienstorientierte Anwendungen (Alt, Heutschi und Österle, 2003). Web Services bieten einen guten Ausgangspunkt als treibende Technologie für das Anbieten und Konsumieren von Informationen. Zusätzlich bildet das Semantische Web einen grundlegenden Pfeiler hin zur sinnhaften Beschreibung von Daten und damit von *Informationen*. Die Definition eines semantisch angereicherten Informationsmodells bietet darüber hinaus eine Investion in die Zukunft (Behrendt, 2003).

3.1 Web Services

> *»A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.« (Booth, Champion, Ferris, McCabe et al., 2003)*

Der folgende Abschnitt gibt einen Überblick über die Technologien und Standardisierungsbemühungen im Umfeld der Web Services[7]. Dazu gehört zunächst das allgemeine Architekturmuster der Service-orientierten Architektur (SOA), das durch die Web Services einen enormen Aufwind bekommen hat. Anschließend folgt eine Einführung in die für Web Services grundlegenden Technologien SOAP und WSDL. Abschließend wird die für eine effiziente Entwicklung und spätere Evolution von dienstorientierten Web-Anwendungen wichtige Unterstützungstechnologie für Dienstverzeichnisse vorgestellt. Dabei wird konkret UDDI (Universal Description, Discovery and Integration) beleuchtet, das wichtige Aufgaben beim Spezifizieren, Suchen und Finden von Web Services ermöglicht.

3.1.1 Service-orientierte Architektur (SOA)

Eine Service-orientierte Architektur (SOA) beschreibt ein Konzept, das eine Anwendung als Zusammenschluss autonomer Subsysteme – üblicherweise in Form von Diensten – realisiert. Diese Dienste arbeiten zusammen, indem sie sich wechselseitig ihre Schnittstelle zur Verfügung stellen und durch den Austausch von Nachrichten gegenseitig auf ihre Funktionen zugreifen. Die komplette Autonomie und lose Kopplung dieser Subkomponenten bringt einerseits Vorteile bezüglich der Robustheit und Flexibilität der gesamten Anwendung. Andererseits werden explizit die Wiederverwendung und Anbindung heterogener Systemlandschaften gefördert.

In diesem Zusammenhang besteht die einzige Abhängigkeit der Subkomponenten untereinander durch Kenntnis der Schnittstelle. Änderungen an der zugrunde liegenden Implementierung der Subkomponenten können verborgen werden; immer vorausgesetzt die Schnittstelle bleibt stabil. Berücksichtigt man die Geschwindigkeit, mit der sich Technologien ändern, stellt das Muster der Service-orientierte Architektur eine effiziente Möglichkeit dar, der dadurch verursachten technologischen Agilität in modernen Anwendungen zu begegnen.

[7] Zu deutsch: Web Dienst. Die Begriffe Service, Web Service und Dienst werden in dieser Arbeit synonym verwendet.

Die zentralen Dienstkomponenten in einer SOA definieren sich durch zwei Aspekte:

- dem Dienstendpunkt (*service endpoint*), um einen Dienst zu erreichen *sowie*
- der Dienstschnittstelle (*service interface*) als Voraussetzung, um seine Funktionalität zu nutzen und Nachrichten auszutauschen.

Das SOA Paradigma fördert eine klare Trennung zwischen der Schnittstellenbeschreibung und der zugehörigen Implementierung. Die Schnittstellenbeschreibung wird in standardisierter Form veröffentlicht, die eine explizite Beschreibung der Operationen und Parameter in Form von Nachrichten vorsieht. Im Gegensatz dazu werden die Implementierung und nötige Details und Kenntnisse darüber weitestgehend verschattet. Es spielt somit keine Rolle, welche Plattform oder Programmiersprache zugrunde liegt, die Implementierung des Dienstes stellt sich nach außen lediglich als standardkonforme Realisierung der entsprechenden Schnittstelle dar – immer unter der Prämisse, weder Inhalt noch Struktur der Schnittstelle zu beeinflussen. Eine klare Auftrennung findet sich auch in den beteiligten Komponenten und deren Rolle innerhalb einer Service-orientierten Architektur. Es werden insgesamt drei unterschiedliche Rollen und Aufgaben in einer SOA identifiziert: Diensterbringer (*service provider*), Dienstnehmer (*service consumer*) und Verzeichnisdienst *(service broker*). Ihre jeweils dedizierten Aufgaben werden in folgenden Abschnitten kurz umrissen. Der Zusammenhang zwischen ihnen findet sich überblicksweise in Abbildung 3-1.

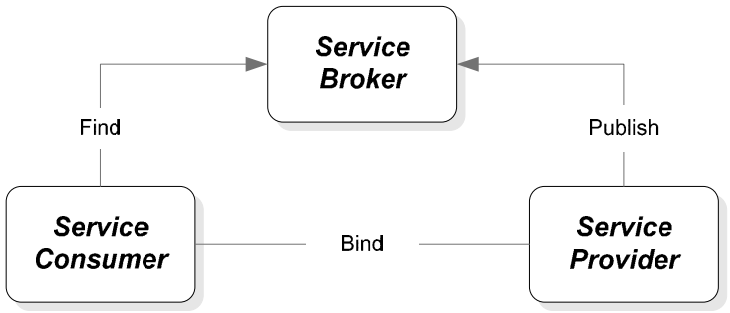


Abbildung 3-1: Die Beziehung zwischen den Komponenten einer Service-orientierten Architektur

3.1.1.1 Diensterbringer

Ein Diensterbringer stellt eine Dienstkomponente zur Verfügung, die eine bestimmte Aufgabe löst und dafür auf eine entsprechende Implementierung zurückgreift. Entscheidend ist hierbei eine eindeutige Definition der Schnittstelle und der damit verbundenen Dienstoperationen, die veröffentlicht werden sollen. Der Diensterbringer hat einerseits die Möglichkeit, seine Dienstleistung bei einem entsprechenden Verzeichnisdienst zu veröffentlichen (*publish*). Je nach Beschaffenheit des Diensterbringers und zusätzlicher Möglichkeiten zur Provisionierung des Vermittlers können hier neben der Veröffentlichung der Schnittstelle auch weitere Eigenschaften wie Taxonomierung und Kategorisierung, semantische Beziehungen oder Lizenzierungsinformationen eingetragen werden. Eine Dienstleistung wird durch eine entsprechende Anfragenachricht erbracht oder angestoßen und kann dann gegebenenfalls eine entsprechende Antwortnachricht hervorbringen.

3.1.1.2 Dienstnehmer

Der Dienstnehmer spielt die Rolle des entsprechenden Gegenparts eines Diensterbringers. Er wird in der Literatur oft auch als der Dienst Requestor bezeichnet. Dabei ist es zunächst notwendig einen passenden Diensterbringer zu finden (*find*) bevor seine Funktionalität in Anspruch genommen werden kann. Dazu kann ein Dienstnehmer eine entsprechende Anfrage an den Dienstvermittler stellen und um einen passenden Dienst – in Form seines Endpunktes und der Schnittstellenbeschreibung – ersuchen. Eine entsprechende Anfrage kann dabei beispielsweise gewünschte Typinformationen, Qualitätsparameter oder Lizenzierungsinformationen beinhalten, abhängig davon, was ein Dienstvermittler anbietet. Auf eine Anfrage können unter Umständen mehrere passende Dienste als Antwort geliefert werden; die Auswahl eines Dienstes aus dieser Liste obliegt dann dem Dienstnehmer. Hat sich der Dienstnehmer für einen Dienst entschieden, kann auf Basis der standardisierten Dienstbeschreibung eine Softwarekomponente als Stellvertreter (*Proxy*) generiert werden, der die Kommunikation mit diesem Dienst realisiert. Dieser Vorgang der Bindung (*bind*) kann entweder statisch oder dynamisch erfolgen. Im Fall der statischen Bindung ist ein späterer Austausch des Endpunkts bei gleich bleibender Schnittstelle nicht mehr möglich. Eine dynamische Bindung gestattet das spätere Austauschen des Endpunktes und erlaubt so einen flexiblen Umgang zwischen Dienstnehmer und Vermittler. Durch den erzeugten Proxy werden schließlich Nachrichten zwischen Dienstnehmer und -erbringer zur Realisierung des Dienstes ausgetauscht.

3.1.1.3 Dienstvermittler

Der Dienstvermittler nimmt seine vermittelnde Position zwischen Dienstkonsumenten und Diensterbringer ein. Er erhält die Veröffentlichungsanfrage des Dienstgebers und speichert die Informationen in einem Dienstverzeichnis. Außerdem verarbeitet er die Suchanfragen von Dienstnehmern, um passende Dienste im Dienstverzeichnis zu finden. Dabei kommt dem Vermittler oder Makler eine lediglich optionale Rolle innerhalb einer Service-orientierten Architektur zu. Er tritt nur dann in Erscheinung, wenn sich Dienstnehmer und Dienstgeber nicht schon im vornherein kennen. In sich ändernden Umgebungen, die einer permanenten Evolution unterworfen sind, nimmt ein Dienstvermittler nachgerade eine Schlüsselrolle ein, da ansonsten das Finden und Verwalten von Diensten nicht mehr effizient durchgeführt werden kann.

Nachdem die strukturellen Komponenten einer Service-orientierten Architektur vorgestellt wurden, geben die nächsten Abschnitte einen Überblick über die grundlegenden Technologien, die eingesetzt werden können, um eine SOA umzusetzen. Obwohl das zu Grunde liegende Architekturmuster prinzipiell unabhängig von bestimmten Technologien ist, wird im Folgenden die aktuell am häufigsten diskutierte Besetzung (Quantz und Wichmann, 2003) von technologischen Vertretern beschrieben – der Web Service Technologie. Die rudimentären Technologien umfassen dabei die Vertreter SOAP, als Spezifikation zur Beschreibung von Nachrichten, WSDL als Beschreibung von Diensten und UDDI als Beschreibung eines Dienstvermittlers. Diese drei Basistechnologien erlauben die Realisierung der Interaktion wie sie in Abbildung 3-1 dargestellt wurden.

3.1.2 SOAP – von der Operation zur Nachricht

SOAP stellt die Kerntechnologie im Umfeld der Web Services dar und dient als Kommunikationsprotokoll für die Web Services-Technologie. SOAP spezifiziert dabei lediglich das Nachrichtenformat, in dem Informationen zwischen zwei Systemen ausgetauscht werden, die über Web Services-Technologie miteinander verbunden sind. SOAP stand ursprünglich für »**S**imple **O**bject **A**ccess **P**rotocol« und wurde in seiner ersten Version als XML-basiertes RPC-Protokoll (Remote Procedure Call) von Userland und Microsoft entwickelt (Winer, 1999). Die Version 1.1 wurde dann gemeinsam von den Firmen Ariba, IBM und Microsoft entwickelt. Es gewann sehr schnell die Unterstützung der wichtigsten Anbieter im Bereich der Web Services und wird seither im Rahmen der »Web Services Activity« vom W3C weiterentwickelt. Aktuell befindet sich SOAP in Version 1.2 und wurde im Juni 2003 vom W3C als »Recommendation« verabschiedet (Mitro, 2002).

Während die an die XML-RPC basierten Nachrichten ursprünglich lediglich über HTTP transportiert werden konnten, findet sich ab Version 1.1 eine flexiblere Bindung an unterschiedliche Transportprotokolle. Gegenüber dem Vorgänger XML-RPC verfügt SOAP über wesentliche Vorteile hinsichtlich einer Flexibilisierung der Kommunikation zwischen zwei Informationssystemen. So können Nachrichten sowohl synchron als auch asynchron verschickt werden. Außerdem gestattet SOAP die Definition von benutzerdefinierten Datentypen. Ein wesentlicher Punkt aber bildet die Möglichkeit zur Spezifikation eines Transportprotokolls unabhängig von der übermittelten Nachricht durch eine spezielle Bindung (Chinnici, Gudgin, Moreau und Weerawarana, 2003). Auf diese Weise lassen sich unterschiedliche Protokolle neben HTTP, beispielsweise auch SMTP (*Simple Mail Transport Protocol)*, FTP (*File Transfer Protocol*) oder auch TCP (*Transmission Control Protocol*) einsetzen, um Nachrichten zu transportieren.

Wie eingangs erwähnt, wurde SOAP zum Austausch von Nachrichten zwischen verschiedenen Kommunikationspartnern konzipiert. Eine SOAP Nachricht wiederum basiert auf XML und besteht aus drei Elementen: dem *SOAP Envelope*, dem *SOAP Header* und dem *SOAP Body*[8]. Abbildung 3-2 stellt den Zusammenhang zwischen den substanziellen Teilen einer SOAP Nachricht dar; ein Beispiel für eine SOAP Nachricht in XML findet sich in Listing 3-1.

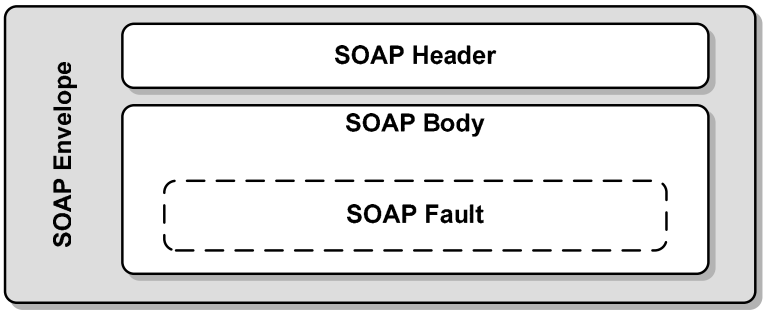


Abbildung 3-2:Schematischer Aufbau und Struktur einer SOAP Nachricht.

[8] Theoretisch existiert noch ein viertes Element, der SOAP Fault. Er wird allerdings als Unterelement des SOAP Body betrachtet und deswegen nicht als eigenständig angesehen.

```
01.  <?xml version="1.0" encoding="utf-8"?>
02.  <soap:Envelope
03.    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
04.    xmlns:ws="http://mwrg.de/services/"
05.   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
06.    <soap:Header>
07.      <ws:ID mustUnderstand="true">4603-4-15955f3bf96c</ws:ID>
08.    </soap:Header>
09.    <soap:Body>
10.     <Read xmlns="http://wsls.net/2004/04/WSLS/Service">
11.      <readContext>
12.       <identifier>11b005d46893fa3a:6043beca:10537eb9</identifier>
13.      </readContext>
14.     </Read>
15.    </soap:Body>
16.  </soap:Envelope>
```

Listing 3-1: Beispiel einer SOAP Nachricht.

3.1.2.1 Das Element SOAP Envelope

Der SOAP Envelope bildet das Wurzelelement einer SOAP Nachricht (Zeile 2). Seine Aufgabe ist die Strukturierung des Header und Body Elements und darf – wie in XML Dokumenten für Wurzelelemente üblich – nur einmal definiert werden. Während der SOAP Header optional sein kann, ist das SOAP Body Element grundsätzlich obligatorisch. Die verwendete SOAP Version wird mittels dafür vorgesehener *namespaces* definiert, im Beispiel wird auf SOAP Version 1.2 (http://www.w3.org/2003/05/soap-envelope, Zeile 3) verwiesen. Dadurch können Versionskonflikte frühzeitig entdeckt und entsprechende Maßnahmen zur Behebung oder zu eventuell einzuleitenden Transformationen ergriffen werden. Zusätzliche anwendungsspezifische Namespace-Erweiterungen (Zeile 4) sind möglich und üblicherweise ebenfalls im SOAP Envelope untergebracht genauso wie die Richtlinien zur Kodierung der Nachrichten (Zeile 5). In SOAP können unterschiedliche Kodierungen zum Einsatz kommen, wodurch unter anderem auch die Interoperabilität zwischen Sender und Empfänger von Nachrichten negativ beeinflusst werden kann. Um Problemen im Umgang mit der Interoperabilität begegnen zu können wurde das *Web Services Interoperability Forum* (WS-I, www.ws-i.org) ins Leben gerufen. Hier werden entsprechende Richtlinien verabschiedet, wie Web Services unterschiedlicher Plattformen mit unterschiedlichen Kodierungen interoperabel zu einander werden können. Diese Richtlinien finden sich in den so genannten *Basic Profiles* und umfassen neben Regeln zur Kodierung auch Erweiterungen hinsichtlich der Interoperabilität mit sicherheitsrelevanten SOAP-Erweiterungen. Die Auswirkungen und Konsequenzen der unterschiedlichen Kodierungen auf die Interoperabilität von Web Services werden im späteren Abschnitt WSDL (siehe 3.1.3) detaillierter besprochen.

3.1.2.2 Der SOAP Header

Der SOAP Header in seiner Eigenschaft als optionales Element gestattet die Erweiterbarkeit von SOAP Nachrichten mit jeglicher Art von Kontroll- und Managementinformationen. Syntaktisch, das heißt mit seinem Namespace, gehört das Kopfelement zum umschließenden Envelope Element. Die einzelnen Kindelemente des Kopfes werden als *SOAP Header Block* bezeichnet und enthalten die jeweiligen Informationen und Parameter zu den Erweiterungen. Dabei können mehrere solcher Erweiterungsblöcke pro SOAP Nachricht definiert werden. Jede dieser Erweiterungen verfügt dabei über ihren eigenen, wohldefinierten Namespace. Eine Erweiterung kann auch spezielle Eigenschaften definieren, die das weitere Verarbeiten einer SOAP Nachricht beeinflussen. Neben Kodierungsrichtlinien finden sich hier Konkretisierungen zur Steuerung der Nachrichtenverarbeitung von eventuellen Erweiterungen in Form der Steuerattribute *role*, *mustUnderstand* und *relay*, die im Folgenden näher beleuchtet werden.

Das Steuerattribut *role*, das in der Version 1.1 noch *actor* hieß, ermöglicht die Abbildung einer einzelnen SOAP Header Informationseinheit auf einen bestimmten Nachrichtenknoten innerhalb des Nachrichtenflusses. Damit werden Szenarien möglich, in denen eine SOAP Nachricht nicht direkt, d.h. in einem Schritt, zwischen Sender und Empfänger verschickt wird. Vielmehr können dabei beliebig viele Zwischenschritte durchlaufen werden, in denen die Nachricht auf verschiedene Weise in Abhängigkeit ihres Inhalts verarbeitet werden kann. Syntaktisch wird das *role* Attribut durch den XML Schema Typ xs:anyURI beschrieben. Dabei gibt es vordefinierte Verarbeitungsrollenwerte, die eine effiziente und standardisierte Bearbeitung von Nachrichten gestatten. So kann beispielsweise bestimmt werden, dass eine SOAP Header Informationseinheit lediglich vom letzten Empfänger bearbeitet werden muss oder alle Zwischensysteme (*SOAP intermediary*) zur Verarbeitung herangezogen werden müssen. Auf diese Weise lassen sich SOAP Nachrichten durch verschiedene Zwischensysteme routen, wobei gleichzeitig eine Ende-zu-Ende Sicherheit durch beispielsweise WS-Security (Nadalin, Kaler, Hallam-Baker und Monzillo, 2003) gewährleistet werden kann. Werden keine Angaben bezüglich der Verarbeitungsrolle gemacht, gilt, dass der letzte Empfänger die Verarbeitung des Nachrichtenknotens übernimmt.

Aufgrund der Erweiterbarkeit von SOAP durch Hinzufügen weiterer Verarbeitungsrollen kann sich aufgrund unbekannter oder inkompatibler SOAP Erweiterungen ein Problem hinsichtlich der Stabilität von SOAP Anwendungen ergeben. Oftmals sind auch Zwischensysteme beteiligt, die bestimmte, anwendungsspezifische Erweiterungen, die nur für den eigentlichen Empfänger der Nachricht relevant sind, nicht zu kennen brauchen. Um diese anwendungsspezifischen Header Informationseinheiten durch eventuell beteiligte Zwischensysteme hindurchzuschleusen, kommen die Parameter *mustUnderstand* und *relay* zum Einsatz. Einerseits kann damit gesteuert werden, ob (*mustUndertsand=true*) eine Header Informationseinheit durch ein Zwischensystem bearbeiten werden muss. Ist dies der Fall und ein Zwischensystem ist nicht in der Lage die Informationseinheit zu bearbeiten wird eine SOAP Fault generiert, der die entsprechende Fehlerursache, in dem Fall der SOAP Fault Code *MustUnderstand*, enthält. Andererseits lässt sich steuern, ob eine solche Informationseinheit ohne eine weitere Bearbeitung in die ausgehende SOAP Nachricht kopiert werden soll (*relay=true*) und so quasi eine Weiterleitung dieser Informationseinheit darstellt. Aus Effizienzgründen und um die Nachrichten klein zu halten, werden die Verarbeitungssteuerattribute nur im positiven Fall einer Verarbeitung spezifiziert.

3.1.2.3 Das SOAP Body Element

Das SOAP Body Element definiert die eigentliche Nachricht und ist dementsprechend für eine SOAP Nachricht obligatorisch. Innerhalb des Body Elements sind die anwendungsspezifischen Daten, die den Dienst im eigentlichen Sinne erbringen, untergebracht. Hier finden sich Methodenaufrufe und entsprechende Parametrisierungen genauso wie korrespondierende Antworten, serialisiert in XML Syntax, wieder. Die Form der Serialisierung kann auf unterschiedliche Art und Weise geschehen. SOAP sieht hier eine bestimmte Auswahl von Kodierungsstilen (*encodingStyles*) vor. Dadurch lassen sich zwar unterschiedliche Plattformen besser anbinden, bieten allerdings auch ein gefährliches Potenzial für Inkompatibilitäten und Interoperabilitätsprobleme. Die einzelnen Teile eines SOAP Body Elements werden als sogenannte *body entries* bezeichnet und stellen die Grundstruktur der Nachricht dar. Einen Sonderfall stellt hierbei der *SOAP fault* dar, der einen Fehlerfall indiziert. Er stellt eine standardisierte Schnittstelle für die Ausnahmebehandlung (*exception handling*) dar. Dadurch können die Fehlerbehandlungs-Mechanismen, wie sie in modernen Programmierumgebungen üblicherweise zu finden sind, mit Hilfe von SOAP beschrieben werden[9]. Im Ausnahmefall generiert der entsprechende Knoten eine neue SOAP Nachricht, die entsprechende Hinweise auf die Fehlerursachen sowie detailliertere Beschreibungen beinhalten kann. Die generierte *fault* Nachricht wird dann an den ursprünglichen Knoten zurückgeschickt. Listing 3-2 stellt eine solche Fehlernachricht dar, die den Fehlerfall beschreibt, wonach eine bestimmte Operation nicht zugeordnet werden konnte.

```
01.  <soap:Fault>
02.    <soap:Code>
03.      <soap:Value>soap:Server</soap:Value>
04.      <soap:Subcode>
05.        <soap:Value>generalException</soap:Value>
06.      </soap:Subcode>
07.    </soap:Code>
08.    <soap:Reason>
09.      <soap:Text xml:lang="en">No such operation: method Read</soap:Text>
10.    </soap:Reason>
11.  </soap:Fault>
```

Listing 3-2: Eine SOAP fault Nachricht

Das SOAP Protokoll stellt eine Spezifikation dar, die es ermöglicht Nachrichten zwischen Interaktionspartnern zu beschreiben. Im Sinne der Web Service-Technologie stellt SOAP einen essenziellen Baustein zur Kommunikation zwischen diesen Partnern dar. Um jedoch eine Kommunikation effizient zu unterstützen, bedarf es einer Offenlegung über angebotene

[9] Das Vorhandensein einer standardisierten Fehlerbehandlung stellt einen wesentlichen Vorteil gegenüber proprietären Vertretern wie XML-RPC oder REST Ansätzen dar. Diese sind zwar aufgrund ihres weniger komplexen Aufbaus unter bestimmten Umständen performanter, vernachlässigen aber für einen nachhaltigen Einsatz entscheidende Faktoren, wie beispielsweise die Ausnahmebehandlung.

Operationen und Datenformate, die einen Dienst erbringen. Dieser Aufgabe widmet sich eine eigens dafür entworfene Sprache, die Web Services Description Language (WSDL), die im folgenden Abschnitt vorgestellt wird.

3.1.3 WSDL – von Nachrichten zu Schnittstellen

Der **W**eb **S**ervice **D**escription **L**anguage (WSDL) Standard bildet einen weiteren Baustein der Web Service-Technologie. Er definiert eine Sprache, die es gestattet, die Schnittstelle eines Web Services in standardisierter Form zu veröffentlichen. WSDL Schnittstellenbeschreibungen sind grundsätzlich unabhängig von konkret eingesetzten Programmiersprachen oder der verwendeten SOAP Kodierung. Die Schnittstellenbeschreibungen sind so leichter in unterschiedlichen heterogenen Plattformen verwendbar und sich eventuell ergebende Interoperabilitätsprobleme zwischen verschiedenen Web Service Rahmenwerken sind besser beherrschbar.

Wie SOAP wird auch WSDL durch das W3C beschrieben und in Form von Empfehlungen standardisiert. WSDL liegt zum Zeitpunkt dieser Arbeit in der Version 2.0 aktuell lediglich als *Draft* vor, schickt sich aber an die Nachfolge der weit verbreiteten WSDL Version 1.1 anzutreten. Die wesentlichen Neuerungen von WSDL Version 2.0 umfassen dabei unter anderem das Hinzufügen von Message exchange patterns oder einer verständlicheren Benennung der Schnittstellenkonzepte. Eines der wesentlichen Konzepte von WSDL, die Trennung der abstrakten von der konkreten Beschreibung eines Web Services, findet sich auch in Version 2.0 wieder. Die Sprache selbst ist eine Anwendung der XML und folgt somit den entsprechenden Regeln zur Konstruktion von Dokumenten. Die inhaltliche Beschreibung eines Web Services (nach Version 1.1) besteht dabei aus den sechs Bestandteilen: Definition (*definition*), Typen (*type*), Nachrichten (*messages*), Schnittstelle (*portType*), Abbildung (*binding*), Dienst (*service*). Üblicherweise werden WSDL Beschreibungen aus vorhandenem Programmcode heraus erzeugt. Dieses Vorgehen wird auch als *Code-first* Variante bezeichnet. Es zeichnet sich durch seinen verhältnismäßig geringen Aufwand zur Erzeugung des WSDL Dokuments aus, da es automatisiert durch eine entsprechende SOAP Implementierung erreicht werden kann. Dem gegenüber steht allerdings eine Einschränkung der Interoperabilität zwischen den beteiligten Kommunikationspartnern unterschiedlicher Plattformen, da beispielsweise Datentypen einer zugrunde liegenden Programmierplattform zur abstrakten Beschreibung herangezogen werden und nicht die plattformunabhängigen XML Schema Typen. Das hinsichtlich der Interoperabilität zwischen den beteiligten Kommunikationspartnern bessere Vorgehen wird auch als *WSDL-first* oder *Contract-first* Variante bezeichnet. Hierbei wird zunächst der Entwurf der Schnittstelle in Form einer entsprechenden Beschreibung eines WSDL Dokuments zur Verfügung gestellt und danach eine Implementierung erzeugt. Gerade in Hinsicht auf ein solches Vorgehen ist es wichtig ein Verständnis für den Aufbau einer Web Service Beschreibung zu haben. Abbildung 3-3 stellt den Aufbau eines WSDL Dokuments anhand seiner Bestandteile vor und nimmt eine Klassifikation in die drei wesentliche Bereiche Adresse (*address*), Anbindung (*binding*) und Schnittstellenvertrag (*contract*).

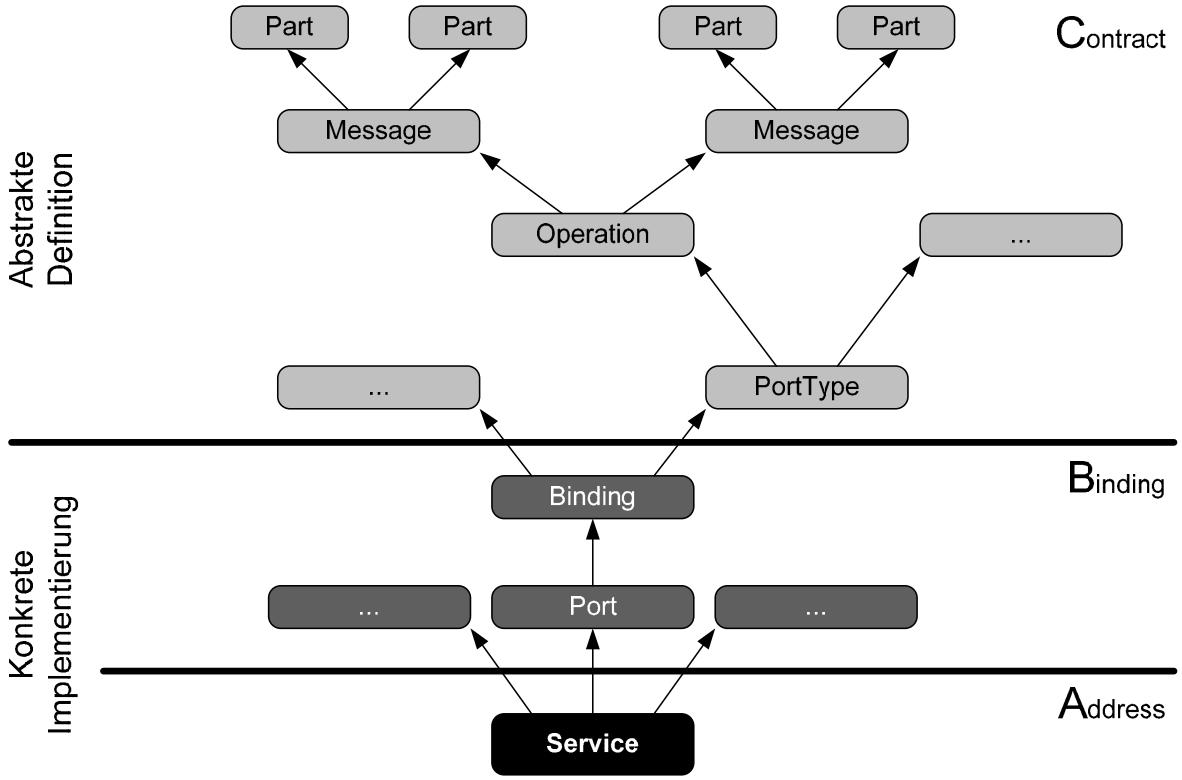


Abbildung 3-3: Ein WSDL Dokument und seine zugehörigen Komponenten klassifiziert in das Address-Binding-Contract Schema.

Die funktionale Beschreibung eines Web Services basierend auf WSDL wird durch eine Menge von abstrakten und konkreten Dienstspezifikation weiter detailliert. Diese Definitionen sind in einem eigens dafür vorgesehenen Element (*definitions*) enthalten und bilden den Ausgangspunkt für die Dienstbeschreibung. Das Wurzelelement ermöglicht hierbei die globale Vorhaltung sämtlicher anwendungsspezifischer Namespace-Definitionen, die für einen Dienst benötigt werden. Dadurch kann die Leserlichkeit erhöht und gleichzeitig die Größe des gesamten Dokuments deutlich reduziert[10] werden.

Die für den Vertrag wesentlichen Bereiche umfassen die Datentypen (*types*), die für die Konstruktion der entsprechenden Nachrichten (*messages*) verwendet werden. Jegliche Nachricht, die mit Hilfe von SOAP transportiert werden soll, muss als korrespondierender Datentyp beschrieben sein. Die Beschreibung dieser Datentypen wird dort mit Hilfe des XML Schema Standards vorgenommen. WSDL gestattet zwar theoretisch die Verwendung weiterer Typsysteme neben XML Schema, in der Praxis wird dies allerdings kaum genutzt. Der Aufbau einer Nachricht kann dabei durch Wiederverwendung bereits definierter Typen in Form einzelner Teile (*message parts*) zusammengesetzt werden. Die Trennung von Typdefinitionen und Nachrichtenformaten bildet einen guten Ansatzpunkt, um bereits existierende Datenbeschreibungen in Form von XML Schemas wiederzuverwenden. Die zusammengefass-

[10] Die Größe eines WSDL Dokuments kann durchaus zu Problemen im Umgang mit der automatischen Erzeugung von Dienstkomponenten führen. So führte dies beispielsweise in der IBM WebSphere Implementierung zum vorzeitigen Abbruch bei der Generierung eines SOAP Clients. Details dazu finden sich unter http://www-1.ibm.com/support/docview.wss?uid=swg1PK26998.

ten Nachrichten wiederum definieren die eigentliche Schnittstelle des Dienstes. Dies geschieht durch Kombination einzelner Nachrichten zu einer Operation (*operation*) und der Zuordnung zu einer Ein- bzw. Ausgabe. Eine Operation realisiert dabei ein sogenanntes *Message Exchange Pattern*, das die Kommunikation zwischen Sender und Empfänger beschreibt. Die wohldefinierte Schnittstelle (*portType*) selbst stellt eine Zusammenfassung aller Operationen dar. Dabei können innerhalb eines WSDL Dokuments durchaus mehrere Schnittstellen[11] definiert werden. Die Beschreibung des Vertrags eines Web Services geschieht somit abstrakt, ohne eine Konkretisierung hinsichtlich des verwendeten Transportprotokolls oder der Angabe wo ein Dienst publiziert wird. Es wird lediglich definiert *was* ein Dienst leistet.

Die Frage *wie* ein Dienst erbracht werden soll, wird in der konkreten Anbindung (*binding*) definiert. Hier wird die im Vertrag beschriebene abstrakte Schnittstelle mit einem Transportprotokoll verknüpft. Hierbei werden zu jedem abstrakten Endpunkt des Vertrags (durch den *portType* definiert) die genauen Zugriffsaspekte in Form der eingesetzten Protokolle und Kommunikationsmuster spezifiziert. Ein Service besteht schließlich aus einer Menge von ports, die jeweils durch ein binding mit einem portType verknüpft sind und je eine Netzwerkadresse spezifizieren, unter der die gebündelten Operationen erreichbar sind. Dadurch wird ein Dienst in die Lage versetzt, prinzipiell unter verschiedenen Netzwerkadressen erreichbar zu sein. Dadurch ergeben sich Potenziale um die Ausfallsicherheit durch Vorhalten entsprechender Redundanzen zu erhöhen. In der Praxis gruppiert ein WSDL-Dokument jedoch zumeist zusammenhängende *ports*, die unter einer Netzwerkadresse erreichbar sind (Alonso, Casati, Kuno und Machiraju, 2003).

Außerdem werden die Kodierungsregeln (*encodingStyle*) definiert, die zur Konstruktion der Nachrichten verwendet werden sollen. Hier ergeben sich abhängig von den verwendeten Kodierungen unterschiedliche Gefahrenpotenziale hinsichtlich der Interoperabilität von Web Services (Prommer, 2006). Es bedarf daher noch bestimmter zusätzlicher Mechanismen, die es gestatten die Interoperabilität von Web Services hinsichtlich der Kodierung zu verbessern. So existieren neben dem SOAP encoding noch zwei weitere Stile, der RPC-Bindung und der Dokumenten-Bindung.

Kodierung, abstrakter Endpunkt und die Interoperabilität am Beispiel

Im direkten Vergleich zwischen den beiden Bindungen ist die Dokumentenbindung vorzuziehen, da sie verschiedene Vorteile gegenüber der RPC-Kodierung besitzt. Einer dieser Vorteile ist, dass der Aufbau der SOAP-Nachricht komplett über XML Schema-Definitionen spezifiziert wird. Dadurch können bereits vorhandene und bewährte Validierungswerkzeuge verwendet werden, um die Nutzdaten in Nachrichten anhand des WSDL-Dokuments zu prüfen. Einen weiteren Vorteil bildet die Fähigkeit der besseren Gewährleistung der Konformität mit Interoperabilitätsrichtlinien der WS-I. Dieses Industriekonsortium beschäftigt sich mit der Definition von Richtlinien, um Entwicklern Regeln an die Hand zu geben, wie *interoperable* Web Services gebaut werden sollen. Zusammengefasst bilden diese Regelwerke die so genannten Basisprofile, die als Grundvoraussetzung für die Interoperabilität erfüllt sein müssen. Allerdings ist die Konformität zu solchen Richtlinien, die zuweilen recht unklar definiert sind, eine

[11] Auf diese Weise lassen sich beispielsweise auch Dienstansätze auf Basis des REST-Ansatzes (REpresentational State Transfer) realisieren, indem entsprechende Nachrichten und Formate beschrieben werden, die dann mit Hilfe des HTTP Protokolls auf Basis der Methode GET übertragen werden können.

nicht triviale Aufgabe und erfordert methodische Hilfestellungen, um die Entwickler von Web Services besser zu unterstützen. In (Butek, 2005) werden unterschiedlichen Bindungen in Hinsicht auf die Interoperabilität diskutiert und die literale Dokumentenbindung empfohlen[12]. Hierbei sind jedoch Einschränkungen hinsichtlich der Schnittstelle des zu realisierenden Web Services zu beachten. Dies soll exemplarisch anhand einer Regel des WS-I Basic Profile 1.1 (Ballinger, Ehnebuske, Ferris, Gudgin et al., 2006) aufgezeigt werden.

R2210: »If a document-literal binding in a *DESCRIPTION* does not specify the *parts* attribute on a *soapbind:body* element, the corresponding abstract *wsdl:message* MUST define zero or one *wsdl:parts*.«

Die obenstehende Regel (Regel-Nr. 2210) legt fest, dass im Falle der *literalen Dokumentenbindung* maximal ein Übergabeparameter verwendet werden darf. Das führt dazu, dass sämtliche Methoden mit mehr als einem Parameter unter dieser Bindung nicht den Interoperabilitätsrichtlinien entsprechen. Abhilfe schafft hier eine Spezialisierung der literalen Dokumenten-Bindung, nämlich der wrapped Ausprägung. Hierbei wird der Methodenname als Hülle verwendet, um das Problem zu umgehen. Allerdings besitzt diese Bindung den Nachteil, dass das Überschreiben von Methoden nicht möglich ist. Da die Hüllenelemente für Methoden mit dem jeweiligen Methodennamen benannt werden, müssten beim Überschreiben zwei gleichnamige Schema-Definitionen im WSDL-Dokument existieren. Laut der Schema-Spezifikation des W3C (Thompson, Beech, Maloney und Mendelsohn, 2001) ist dies jedoch nicht erlaubt. Aus diesem Grunde untersagt die Spezifikation WSDL 2.0 auch das Überschreiben von Methoden, die in Web Services gesetzt werden.

Zusammenfassend bedeutet dies, dass nach dem aktuellen Stand der Technik:

- ein interoperabler Web Service *keine Polymorphie* unterstützt und dies auch spätestens mit WSDL 2.0 zu Gunsten der besseren Interoperabilität verboten wird.
- die literale Dokumentenbindung die bessere Wahl für interoperable Web Services im Gegensatz zur literalen RPC Bindung darstellt.
- eine literale Dokumentenbindung nie gleichzeitig mehrere Parameter und die Methodenüberladung gestattet.

Eine offene Frage bleibt damit, wie ein polymorpher Web Service mit multiplen Parametern realisiert werden kann, der zusätzlich auch interoperabel im Sinne der WS-I ist.

[12] Auf die durch SOAP selbst unterstützte Kodierung soll nach WS-I Basic Profile 1.1 komplett verzichtet werden. Begründet wird dies mit der eingeschränkten Interoperabilität, da den Dienstimplementierungen die Datentypumsetzungen von Parametern bereits vorgegeben werden und so zu Datentyp-Missinterpretationen führen kann.

3.1.4 Serviceverzeichnisse - Suchen , Finden und Binden

Dienstverzeichnisse stellen einen wichtigen Pfeiler innerhalb einer SOA dar und unterstützen vier zentrale Aspekte (Chiusano, Schmitt und Crawford, 2001):

- Eine *einheitliche Sicht* auf die in einer Organisation vorhandenen Dienste durch Publikation mittels eines Kataloges zur Registrierung von Metadaten über diese Dienste.
- Die *Wiederverwendbarkeit* und *lose Kopplung* von Diensten durch die Möglichkeit, Dienste nach bestimmten Kriterien zu klassifizieren, zu gruppieren, zu suchen und dynamisch einzusetzen.
- Die flexible *Konfigurierbarkeit und Adaptivität* von Diensten durch die Spezifikation von Zugriffsrichtlinien oder Vorhaltung entsprechender Eigenschaften.
- Die dynamische *Verwaltung und Kontrolle* der Dienste und der Beziehungen zwischen ihnen durch Bereitstellung einer zentralen Managementkomponente zur Definition und Überwachung von Richtlinien.

Die zwei wichtigsten Standards im Bereich der Serviceverzeichnisse für Web Services sind UDDI, das im Rahmen der WSA entwickelt wurde, sowie die durch die *Organization for the Advancement of Structured Information Standards* (OASIS) standardisierte ebXML Registry. Im Folgenden wird UDDI als ein Vertreter für ein Dienstverzeichnis vorgestellt, das eine stärkere Verbreitung durch die Industrie genießt als es bei ebXml der Fall ist[13].

Das UDDI-Protokoll bildet ein plattformunabhängiges Rahmenwerk für die Beschreibung und Suche nach Diensten in einer SOA. Der Standard definiert ein Informationsmodell zur Beschreibung von veröffentlichten Diensten und beschreibt Programmierschnittstellen zur Nutzung der Funktionalität des Serviceverzeichnisses (OASIS, 2004).

[13] http://uddi.org/solutions.html listet die nahezu 20 Produkte auf Basis von UDDI auf. Unter http://www.ebxml.org/implementations/index.htm finden sich dazu zwei Realisierungen einer ebXML Registry. Allerdings existieren auch hybride Ansätze, die jeweils UDDI und ebXML umsetzen, wie im Fall der Infravio Registry (http://www.infravio.com).

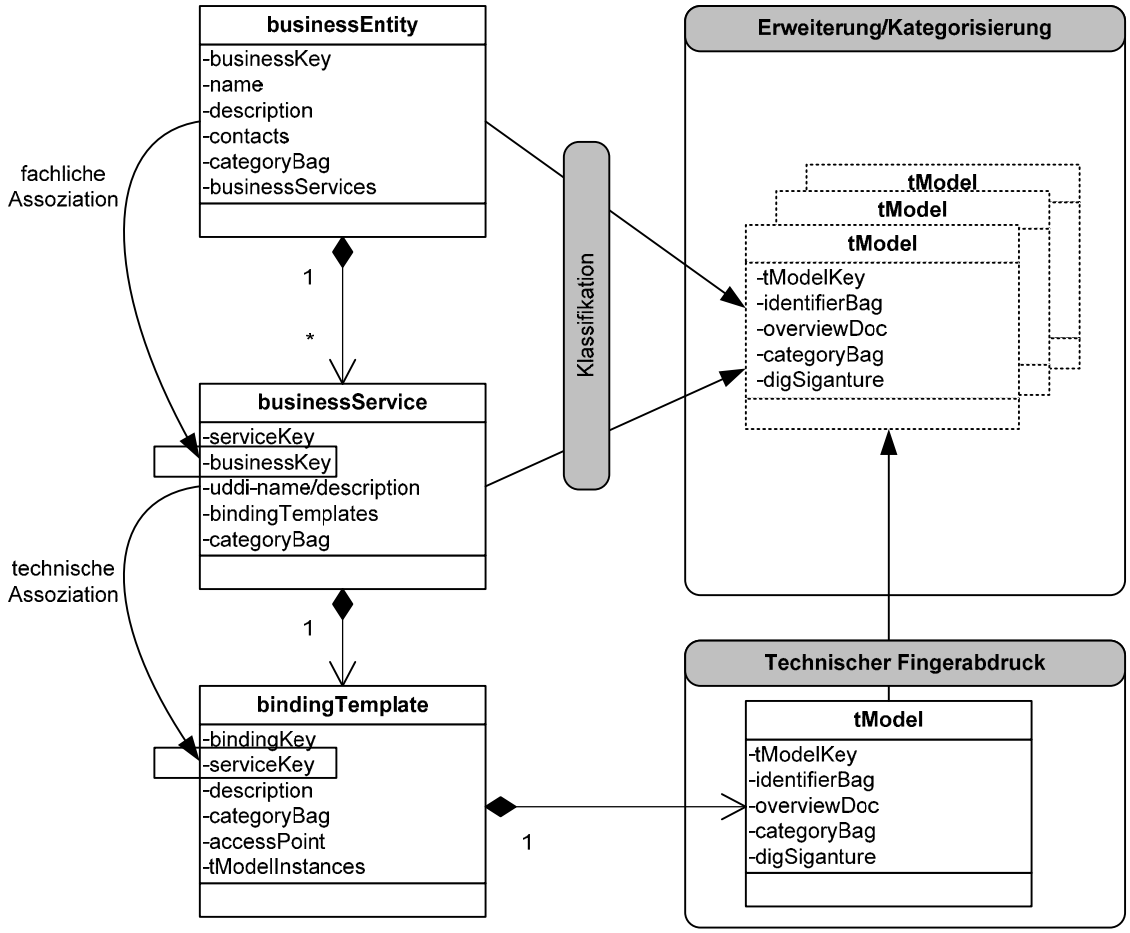


Abbildung 3-4: Das UDDI Datenmodell im Zusammenspiel mit seinen Klassifikationsmerkmalen.

Das UDDI Datenmodell stellt vier in XML Schema definierte Datenklassen vor, um die Beschreibung, Suche und Auswahl eines Services zu unterstützen:

- Eine Beschreibung des Unternehmens oder der Organisation, die einen Dienst zur Verfügung stellt, wird in Form einer Spezifikation der *businessEntity* realisiert. Dieser Datentyp ermöglicht die Hinterlegung von Kontaktinformationen und einer informalen Beschreibung der Aufgaben und angebotene Leistungen eines Dienstes.
- Weitere überwiegend betriebswirtschaftliche Aspekte eines Dienstes werden durch *businessServices* spezifiziert. Ein businessService stellt dabei eine Art logischen Container für eine Menge von fachlich zusammenhängenden technischen Services dar, die von einer businessEntity angeboten werden.
- Zur Beschreibung der technischen Aspekte eines Dienstes gibt es die so genannten *bindingTemplates*. Sie entsprechen den ports in einem WSDL-Dokument (siehe hierzu Abschnitt 3.1.3) und spezifizieren Informationen über die technische Realisierung eines Dienstes. Dazu gehören unter anderem die Netzwerkadresse, unter der ein technischer Service zu erreichen ist (durch ein dediziertes Attribut, dem *accessPoint*). Weitere technische Aspekte des Dienstes werden über eine Assoziation der so genannten »technical Models« (tModels) im *tModelInstances* Attribut realisiert. Die dadurch gegebene Beschreibung ermöglicht noch keinen direkten Aufruf des Web Services, verweist aber auf weitere Quellen (wie etwa ein WSDL-Dokument oder ei-

ner semantischen Beschreibung mittels OWL (siehe Abschnitt 3.1.5.1.4)), mit deren Informationen die technische Nutzung realisiert werden kann.

- tModels wiederum repräsentieren zwei Arten von Informationen. Zum einen verweisen sie auf Taxonomien, die entweder auf Klassifikationssystemen wie dem weltweit umspannenden UNSPSC[14] beruhen oder beliebige andere Kategorisierungen, die etwa geografische Angaben spezifizieren (Zimmermann, Tomlinson und Peuser, 2005). Auf diese Weise lassen sich für alle Daten in UDDI Taxonomien in Form von tModels in den categoryBags der Elemente hinterlegen (vgl. Abbildung 3-4, jedes Datenobjekt in UDDI besitzt ein Attribut *categoryBag*). Dies erlaubt eine Strukturierung der Informationen durch weitere Metadaten. Mit Verweisen durch tModels sind zusätzlich das Attribut *keyValue*, das den konkreten Wert innerhalb des Klassifikationssystems spezifiziert, und das optionale Attribut *keyName*, das eine textuelle Beschreibung des referenzierten Wertes ermöglicht, verbunden (siehe Listing 3-3).

```
<categoryBag>
 <keyedReference keyName="uddi-org:types:nations" keyValue="Germany"
        tModelKey="uddi:uddi.org:categorization:nations"/>
 <keyedReference keyName="kit-org:types:kim-services"
        keyValue="StudentInformation"
        tModelKey="kit:kit.edu:categorization:kim-services"/>
</categoryBag>
```

Listing 3-3: UDDI categoryBag mit Verweisen auf Landes- und Industrieklassifikation sowie organisationsrelevanter (erweiterter) Klassifikationen.

Außerdem repräsentieren tModels Spezifikationen, um wiederkehrende Konzepte bei der Nutzung von Web Services zu kennzeichnen. So können die Nutzung bzw. die Konformität zu Standards, Transportprotokollen oder XML Namespaces durch ein tModel repräsentiert werden (vgl. Listing 3-4), auf die eigentliche Spezifikation wird im jeweiligen tModel über das Feld *overviewURL* verwiesen. Dadurch lassen sich alle Arten unterschiedlicher Beschreibungsformen zusätzlich zu den durch UDDI zur Verfügung gestellten anbinden. Gerade hinsichtlich domänenspezifischen Wissens, das bestimmten Diensten inhärent gegeben ist, können so beispielsweise semantische Beschreibungen auf Basis von Ontologien genauso hinzugefügt werden, wie semiformale oder natürlich-sprachliche Dienstbeschreibungen.

```
<tModel tModelKey="uuid:aa254698-93de-3870-8df3-a5c075d64a0e">
  <name>uddi-org:protocol:soap</name>
  <description xml:lang="en"> tModel that represents the SOAP 1.1 protocol</description>
  <overviewDoc>
```

[14] Der UNSPSC (United Nations Standard Products and Services Code) bildet ein international eingesetztes Klassifikationssystem für die Warenwirtschaft. Der Code besteht aus fünf Ebenen, die Produkte hierarchisch klassifizieren (http://www.unspsc.org/).

```
        <overviewURL>
         http://www.oasis-open.org/../uddi-spec-tc-tn-wsdl-v2.htm#soap
        </overviewURL>
      </overviewDoc>
      <categoryBag>
        <keyedReference
          tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
          keyName="uddi-org:types" keyValue="protocol"/>
      </categoryBag>
    </tModel>
```

Listing 3-4: UDDI tModel zu Kennzeichnung der Nutzung des SOAP-Protokolls.

Die tModels stellen in UDDI das zentrale Instrument zur Spezifikation von Daten zu und Metadaten über einen Service dar. Durch sie können beliebige Dokumente und Spezifikationen im Serviceverzeichnis mit einem Dienst verbunden werden. Das macht sie zu einer idealen Unterstützungstechnologie, die einerseits bereits identifizierte Bereiche (kanonische tModels) für eine Dienstspezifikation umfasst und zur Verfügung stellt. Andererseits ist sie offen für Erweiterungen und lässt sich entsprechend fachspezifischer Domänen mit den dort vorherrschenden Beschreibungsstandards ergänzen. Die tModels selbst verweisen dann auf weitere Ressourcen, die eigentliche Beschreibung des Services findet also in einem externen Dokument statt. UDDI ist somit ein Bezugssystem, das komplementär zu anderen Lösungen und Technologien einen Integrationspunkt verschiedenster Beschreibungsaspekte bietet. Ein weiterer wichtiger Aspekt der tModels ist die Ermöglichung der Suche nach bestimmten Kriterien. So kann durch Taxonomien in einem Serviceverzeichnis nach allen Objekten, die einer bestimmten Kategorie angehören, gesucht werden.

Ähnlich zum *documentation* Element in WSDL, bietet UDDI ein *description* Element, mit dem eine beliebige textuelle Dokumentation, also Beschreibung in natürlicher Sprache, mit einem Objekt verknüpft werden kann. Analog zur Vorgehensweise bei WSDL soll auch hier diese Beschreibung nur bei einem direkten Bezug zum jeweiligen Element eingesetzt werden. Da UDDI aber nicht nur technische Aspekte eines Services abbildet, kann der Rahmen weiter als bei WSDL gefasst werden und beispielsweise auch Vermarktungsaspekte einschließen.

Die beschränkte Granularität bei der Darstellung von Services ist ein Nachteil von UDDI. Die Operationsebene wird durch das Datenmodell von UDDI standardmäßig nicht erfasst und entzieht sich somit einer weiteren Beschreibung. Dies ließe sich durch die individuelle Erweiterung des Spezifikationsrahmens (konkret der XML Schemas, die diesem Spezifikationsrahmen zu Grunde liegen) beheben, wodurch die Standardkonformität aber nicht mehr gegeben wäre.

Anfangs lag der Fokus bei UDDI in der Etablierung der so genannten *Universal Business Registry* (UBR). Damit sollte eine zentrale Stelle für die Registrierung öffentlich zugänglicher Web Services im Geschäftsbereich geschaffen werden. Die UBR wurden ab dem Jahr 2001 von den Firmen SAP, NTT Com, Microsoft und IBM betrieben aber Ende 2005 eingestellt. Gründe dafür waren die mangelnde Qualität der Daten und ein Wandel im Konzept des Einsatzes von Serviceverzeichnissen. Mit Einführung von UDDI Version 3.0 wurde das Hauptaugenmerk des Standards stark auf die Interaktion vieler und eigenständiger Dienstverzeichnis-

se gelegt. Diese bestehen als Infrastrukturkomponenten in Unternehmen und setzen daher kein weltweit zentrales System voraus. Somit wurde der Fokus in UDDI speziell mit Hinblick auf die Unterstützung von *Föderationen* erweitert (OASIS, 2004).

Trotz der eher nachlässigen Behandlung im Vergleich zu anderen Basistechnologien für Web Services (Newcomer und Lomow, 2004), stellt es mittlerweile einen integralen Baustein jeder SOA-Implementierung dar. Alle großen SOA Anbieter sind an der Weiterentwicklung beteiligt und bieten in ihrem Produktportfolio auch Implementierungen des Standards.

3.1.5 Semantische Annotation von Web Services

WSDL-S ist eine XML-basierte Sprache, die WSDL-Elemente mit semantischen Informationen annotiert, um das dynamische Auffinden, die Komposition und letztendlich den Aufruf von Web Services zu ermöglichen. Semantik meint im Rahmen von Web Services eine Spezifikation des Kontexts und der Effekte des Aufrufes eines Services sowie Informationen über das Informationsmodell der ausgetauschten Nachrichten (Sivashanmugam , Verma, Sheth und Miller, 2003).

Dazu wird weder ein Verfahren zur formalen Beschreibung noch ein eigenständiges Spezifikationsdokument definiert, sondern lediglich ein Mechanismus festgelegt, mit dem beliebige formale Beschreibungen mit beliebigen Elementen eines WSDL-Dokuments verbunden werden. Dazu assoziiert WSDL-S weitere Metadaten mit dedizierten Elementen in WSDL, ohne dieses selbst zu beschreiben. Die Autoren von WSDL-S verstehen das Verfahren als evolutionären, zu bisherigen Implementierungen kompatiblen Ansatz zur Erweiterung von WSDL.

Im Gegensatz zu Verfahren wie OWL-S[15] oder WSMO[16], die jeweils mit Hilfe spezieller Ontologien versuchen *alle* Aspekte eines Web Services formal zu beschreiben, stellt WSDL-S einen pragmatischeren Ansatz dar, in dem auf die *punktuelle* Erweiterung bestehender, weit verbreiteter Standards gesetzt wird. Die Einbindung der Annotation wird über das in WSDL 1.1 vorhandene Feature der *Language Extensibility* möglich. Dadurch können beliebige Objekte eines WSDL-Dokuments um zusätzliche Attribute oder weitere Sub-Elemente erweitert werden.

In WSDL-S sind folgende Bausteine zur Erweiterung der Syntax von WSDL definiert:

- Ein Attribut `modelReference`, mit dem ein beliebiges XML-Element in WSDL mit einer Entität in einem semantischen Modell, wie etwa einer Ontologie, verknüpft werden kann.
- Ein Attribut `schemaMapping`, mit dem auf eine Transformation zwischen Elementen in WSDL und Elementen in einer Ontologie verwiesen werden kann. Dies wird zum Beispiel durch den Verweis auf ein Dokument in einer Sprache wie *XSL Transformation* (XSLT, s. (W3C-SOA, 2004)) erreicht, das rein syntaktische Differenzen in der Definition komplexer Elemente kompensiert.

[15] http://www.daml.org/services/owl-s/

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

- Zwei neue Elemente mit Namen `precondition` und `effect`, die als Kindknoten des Elements `operation` in einem WSDL-Dokument auftreten und Vorbedingungen und Effekte des entsprechenden Operationsaufrufs definieren.
- Eine Erweiterung des Elements `portType` um das Attribut `category`, das eine Kategorisierung des Services, ähnlich zu dem Attribut `categoryBag` in UDDI, ermöglicht. Dadurch kann ein Web Service bei der Registrierung in einem Serviceverzeichnis direkt entsprechenden Kategorien zugeordnet werden.

Die tatsächliche Beschreibung findet durch eine Verknüpfung der WSDL-Elemente über URIs mit Ontologien oder anderen formalen Spezifikationen statt.

3.1.5.1 Formale Beschreibungsverfahren – der Semantic Web Stack

Zur formalen Beschreibung von Sachverhalten können im Bereich der Web Services Ontologien genutzt werden. Dazu existiert eine durch das W3C standardisierte Menge von Spezifikationen, die ursprünglich im Kontext des *Semantic Web* entwickelt wurden (Tim Berners-Lee, Hendler und Lassila, 2001b). Obwohl viele Teile des Semantic Web noch als Vision angesehen werden müssen, finden sich hier zahlreiche Ansätze, die zukünftig eine wichtige Rolle spielen werden (Behrendt, 2003). Gerade die Vielzahl an wissenschaftlichen Veröffentlichungen und eigens dafür etablierter Konferenzen und Workshops spiegeln diesen Trend wieder. Im Folgenden werden die einzelnen Bestandteile des Semantic Web vorgestellt, um das Potenzial der Beschreibungssprachen, gerade auch hinsichtlich einer Verwendung für eine dienstorientierte Web-Anwendung, aufzuzeigen. Dazu wird zunächst der allgemeine Begriff der Ontologie eingeführt und danach die Schichtenarchitektur des Semantic Web vorgestellt, das eine Systematik zur Definition von Ontologien vorsieht.

Der Begriff Ontologie bildete sich in der Philosophie und beschreibt seit Aristoteles die Lehre vom Seienden bzw. die Studie der Existenz. Berners-Lee ergänzt den Begriff außerdem mit der Aussage: »(...) die Ontologie als akademische Disziplin studiert die Fragen der Art, welche Typen von Dingen überhaupt existieren.« (Tim Berners-Lee, Hendler und Lassila, 2001a). Eine Ontologie erklärt die Beschaffenheit der Welt, indem sie Begriffe spezifiziert. Ausgehend von den drei Dimensionen eines Zeichens in der Semiotik (Syntax, Semantik, Pragmatik) ergeben sich im Alltag häufig semantische Konflikte bei der Zuordnung von Wörtern zu realen Gegenständen. Da der durch ein Wort beim Rezipienten erweckte Begriff sich auf einen anderen realen Gegenstand beziehen kann, als der Sender es beabsichtigt hat, kann dies zu Missverständnissen zwischen beiden führen. Eine Ontologie kann den erzeugten Begriff durch die Interpretation des Wortes in einem bestimmten Kontext so spezifizieren, dass diese Missverständnisse vermieden und der Begriff somit eindeutig einem realen Gegenstand zugeordnet werden kann.

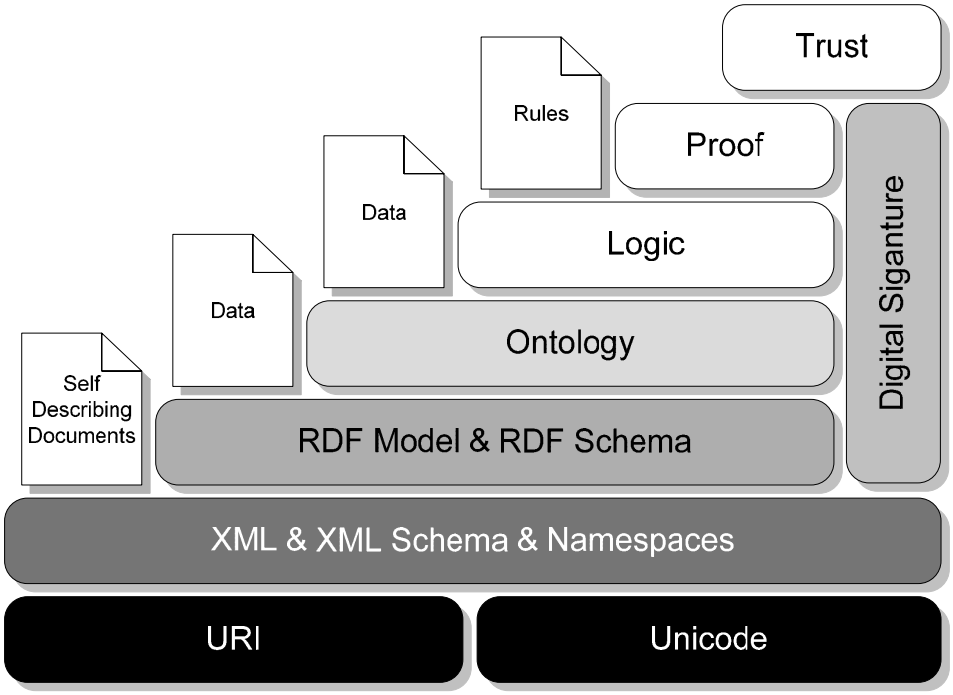


Abbildung 3-5: Architektur des Semantic Web (Berners-Lee, Xml 2000)

Auf unterster Ebene (vgl. Abbildung 3-5) erlaubt die XML-Syntax in Verbindung mit XML Schema die Definition von Elementen und Attributen sowie der passenden Codierungen und Referenzierung von Dokumenten oder Teilen davon. Die Eigenschaften dieser einzelnen Entitäten beschreibt das Resource Description Framework (RDF) in maschinell verarbeitbarer Form. Dazu werden in den so genannten RDF-Tripeln der Gestalt »Subjekt, Prädikat, Objekt« Aussagen über Elemente in einer Domäne gemacht, die in XML-Syntax darstellbar sind (Klyne und Carroll, 2004). RDF bietet allerdings weder Mechanismen, um die angegebenen Eigenschaften näher zu bestimmen noch um genaue Beziehungen zwischen Objekten zu definieren. Zu diesem Zweck stellt RDF-Schema (RDFS) ein standardisiertes Vokabular zur Verfügung, durch das eine detailliertere Angabe von Beziehungen und Hierarchien in einer Domäne realisiert werden kann. RDFS spezifiziert dazu ein Datenmodell – in etwa vergleichbar mit dem einer Objekt-orientierten Programmiersprache – und versteht sich als semantische Erweiterung von RDF (Brickley und Guha, 2004). Die Web Ontology Language (OWL) führt diesen Ansatz konsequent fort. Dem auf Ebene von RDFS bereits definierten Vokabular werden dafür weitere Konzepte und Ausdrucksmöglichkeiten hinzugefügt, so dass eine vollständige Erfassung der Beziehungen der Objekte der abgebildeten Welt in einer Ontologie möglich ist (Bechhofer, Harmelen, Hendler, Horrocks et al., 2004). Das OWL Dokument in Listing 3-5 zeigt die Generalisierung in OWL mit Hilfe von RDF Schema. Hier wird ein Maler als eine Unterklasse von Künstler definiert.

```
01.  <owl:Class rdf:ID=”Painter”>
02.    <rdfs:subClassOf rdf:ressource=”#Artist”/>
03.  ...
04.  </owl:Class>
```

Listing 3-5: Generalisierung von OWL-Klassen beschrieben mit RDFS-Konstrukten.

Die Semantik der in RDF, RDFS und OWL eingeführten Konzepte sind in einer formalen Sprache definiert und ermöglichen den Übergang von einem Datenmodell zu einem Wissensmodell (Daconta, Smith und Obrst, 2003).

Im Folgenden werden die einzelnen Elemente der verschiedenen Schichten vorgestellt, wobei das Hauptaugenmerk der Darstellungen auf dem Ontologievokabular in Form der Web Ontology Language (OWL) liegen wird. Dieses Vokabular bildet eine gute Voraussetzung zur umfassenden Beschreibung von semantischen Assoziationen. Es erlaubt neben dem Definieren einer Begriffswelt – ähnlich einer Taxonomie – die Einführung von Regeln, wie diese Begriffe in Beziehung miteinander stehen. Dadurch eignen sich Ontologien ideal zur Beschreibung von inhaltlichen Zusammenhängen von Informationen in einem Informationsraum.

3.1.5.1.1 URI und Unicode

Die Symbol- bzw. Referenzschicht wird benutzt, um Symbole bzw. Identifikatoren und Referenzen auf Objekte, die in Ontologien beschrieben werden, zur Verfügung zu stellen. Erreicht wird dies durch die so genannten Unified Resource Identifier (URI) (T. Berners-Lee, Fielding, Irvine und Masinter, 1998), die beliebige Ressourcen in eindeutiger Weise identifizieren, sowie durch Unicode, ein mehrsprachiges Character-Encoding System, das einen standardisierten Zeichensatz zur Beschreibung von Web-Ressourcen zur Verfügung stellt und dadurch den Austausch von Informationen über nationale und kulturelle Grenzen hinweg ermöglicht. Identifikatoren sind ein Schlüsselaspekt des Semantic Web, da mit ihnen Ressourcen benannt werden können.

3.1.5.1.2 XML, Namespaces und XML Schema

Wie in Abbildung 3-5 deutlich zu erkennen ist, bildet XML die Grundlage des Semantic Web. XML wird neben URIs und Unicode-Technologien auch noch von Namespaces (Tim Bray, Hollander und Layman, 1999) unterstützt. Hiermit können Elemente innerhalb eines XML-Dokumentes eindeutig identifiziert werden. Zusätzlich vermeidet man dadurch Namenskollisionen. XML bildet zusammen mit Namespaces und XML Schema die Transport- bzw. Syntaxebene innerhalb des Semantic Web Stapels.

XML (T. Bray, Paoli, Sperberg-McQueen, Maler et al., 2006) verfolgt die Intention die starre Zuordnung von Beschreibungselementen und deren Bedeutung, wie es in HTML der Fall ist, zu umgehen. Die Sprache ist erweiterbar und erlaubt es eigene Elemente (*engl. tags*) zu definieren. XML ist inspiriert von SGML (Cover, 2000) ohne jedoch dessen Komplexität zu übernehmen (Lobin, 2003). Es wurde jedoch schnell deutlich, dass eine der wesentlichen Stärken von XML in der Möglichkeit liegt, Daten in strukturierter Weise auszutauschen. Mit der Einführung von XML Schemas (Thompson, Beech, Maloney und Mendelsohn, 2001) wurde eine Spezifikation geschaffen, die eine syntaktische und in Hinblick auf Datentypen auch teilweise semantische Validierung von XML Dokumenten gestattet. Um allerdings XML-Dokumente Maschinen verständlich und damit autonom verarbeitbar zu machen, bedarf es weiterer Ebenen, um dieses Ziel zu erreichen. Diese Ebenen umfassen die eigentliche semantische Bedeutung der XML strukturierten Daten und werden in ihrer Gesamtheit oft als das Semantic Web bezeichnet.

3.1.5.1.3 RDF und RDF Schema

Das Resource Description Framework (RDF) ist ein Modell zur Repräsentation von Metadaten, das vom World Wide Web Consortium (W3C) entwickelt wurde. RDF basiert auf den drei Konzepten: Ressourcen (Subjekt), Eigenschaften (Prädikat) und Aussagen (Objekt). Ressour-

cen sind durch URIs identifizierbar und können faktisch sämtliche Entitäten beschreiben. Eine Eigenschaft stellt ein spezifisches Merkmal oder eine Beziehung dar, die eine Ressource beschreibt. Jede Aussage über eine Ressource wird in einem Tripel (*triplet*) ausgedrückt, das aus Subjekt, Prädikat und Objekt besteht. Ist das Objekt wiederum eine Ressource, so kann diese durch weitere Tripel beschrieben werden.

RDF-Aussagen beschreiben zusätzliche Fakten über ein XML-Vokabular in einem expliziten, maschinen-lesbaren Format und erlauben deshalb Rechnern, Bedeutungen in einem Kontext zu verstehen. Mittels der Vokabularien von RDF Schema (Brickley und Guha, 2004) können Klassen und Eigenschaften von RDF-Ressourcen beschrieben werden. RDFS fügt den darunter liegenden Schichten somit zusätzliche ontologische Primitive hinzu, die eine Grundlage für die semantische Beschreibung von Daten und Metadaten im Web darstellen. Des Weiteren stellt es Mechanismen bereit, um Eigenschaften zusammenzufassen (Klassenbildung) und grundlegende Einschränkungen, wie beispielsweise Kardinalitäten, festzulegen. Dadurch ermöglicht RDFS mit seiner Ausdrucksstärke ein Grundmaß zur logischen Folgerung auf Basis der semantischen Annotationen. RDFS stellt jedoch keine ausreichenden Mechanismen zur Verfügung, um eine ontologische Beschreibung zu realisieren.

3.1.5.1.4 OWL – die Web Ontology Language

Die Web Ontology Language (OWL) (Bechhofer et al., 2004) wurde entwickelt, um dem Bedürfnis nach einer Ontologiesprache für das WWW nachzukommen. Dahinter verbirgt sich die Vision, Maschinen zu befähigen, dass sie autonom miteinander kommunizieren und relevante Informationen aus dem Web verarbeiten und zusammenführen können. OWL bildet dabei einen wichtigen Schritt in einer langen Kette von Bemühungen unterschiedlicher Behörden und Gremien. Es markiert einen gemeinsamen Standard, der durch zunächst unabhängige Bemühungen seitens der EU mit OIL (Ontology Interface Layer) und des Department of Defense der US-Behörden mit DAML (DARPA Agents Markup Language) vorangetrieben wurde. Bei der Standardisierung von OWL durch das W3C dienten diese beiden Sprachstandards maßgeblich als Startpunkt. Daneben stellte die nahtlose Integration mit existierenden Technologien des W3C wie XML, RDF oder XML Schema eine weitere wichtige Anforderung dar. Das Ergebnis bildet eine umfangreiche Spezifikation, die drei Dialekte mit der Intention beschreibt, verschieden komplexe Szenarien zu unterstützen, die variierende Ausdrucksebenen benötigen. Diese Sprachebenen bilden OWL Full, OWL DL, und OWL Lite und wurden speziell für unterschiedliche Gruppen von Entwicklern und Anwendern konzipiert.

OWL Lite richtet sich dabei in erster Linie an Nutzer, die Klassifikationshierarchien und einfache Mechanismen für Restriktionen benötigen. So werden zwar Kardinalitätsrestriktionen erlaubt, allerdings wird nur der Wertebereich 0 und 1 abgedeckt. Ein weiteres Ziel lag in einer besseren Werkzeugunterstützbarkeit aufgrund der Einfachheit im Gegensatz zu den ausdrucksstärkeren Verwandten. Zusätzlich bietet OWL Lite eine schnelle Möglichkeit zur Migration von Thesauri und anderen Taxonomien (McGuinness und Harmelen, 2004).

OWL DL unterstützt die Nutzer, die ein Maximum an Ausdrucksstärke benötigen und gleichzeitig die vollständige Verarbeitbarkeit (alle Folgerungen können garantiert gezogen werden) sowie die Entscheidbarkeit (jede Verarbeitung wird in endlicher Zeit durchgeführt) gewährleistet brauchen. OWL DL schließt alle OWL Sprachkonstrukte ein, diese können aber nur unter bestimmten Bedingungen angewandt werden. Der Namenszusatz »DL« symbolisiert den Zusammenhang zum Forschungsgebiet der Beschreibungs-Logik (engl. *Description Logic*), die auch die formale Basis für OWL begründet (Baader, Calvanese, McGuinness, Nardi et al., 2003).

OWL Full ist für die Nutzer gedacht, die maximale Ausdrucksstärke und die syntaktische Freiheit von RDF benötigen, allerdings ohne eine Garantie der Verarbeitbarkeit und Entscheidbarkeit. OWL Full erlaubt es einer Ontologie die Bedeutung von vordefiniertem Vokabular zu erweitern. Mit der Mächtigkeit des Sprachumfangs sinkt allerdings auch die Wahrscheinlichkeit, dass ein Schlussfolgerungswerkzeug das komplette Schlussfolgern für alle Eigenschaften von OWL Full unterstützen können wird.

Für die drei Sprachebenen gelten folgende Kompabilitätsbeziehungen bezüglich Ausdrucksstärke und Sprachumfang:

- Jede korrekte OWL Lite Ontologie ist eine korrekte OWL DL Ontologie.
- Jede korrekte OWL DL Ontologie ist eine korrekte OWL Full Ontologie.
- Jede gültige OWL Lite Schlussfolgerung ist eine gültige OWL DL Schlussfolgerung.
- Jede gültige OWL DL Schlussfolgerung ist eine gültige OWL Full Schlussfolgerung.

Dadurch ist OWL für eine Verwendung in sich stark ändernden Umgebungen gut geeignet. Die unterschiedlichen Komplexitätsgrade unterstützen einen sukzessiven Zuwachs an Komplexität, falls dieser in einer konkreten Anwendung erforderlich sein sollte. So lassen sich dedizierte Sachverhalte zunächst in einer einfacheren Komplexitätsstufe modellieren und aufgrund der Aufwärtskompatibilität der Sprachstandards untereinander jederzeit erweitern, falls eine veränderte Situation dies erfordern sollte. Dadurch können Problemstellungen effektiv angegangen werden, indem Entwickler zunächst mit einem überschaubaren Maß an Komplexität konfrontiert werden. Umgekehrt lassen sich auch logische Schlussfolgerungen beschleunigen.

3.1.5.2 Logic, Proof und Trust

Damit ein Bedeutungsnetzwerk von Maschinen verarbeitet werden kann, muss dieses nicht zwangsläufig in der Lage sein, die wahre Bedeutung eines Wortes genau zu verstehen. Es ist ausreichend ein Regelwerk vorzuhalten, mit dem logische Rückschlüsse gezogen werden können. Die Fakten, wie sie in Ontologien ausgedrückt werden, können hierbei als Grundlage für diese logischen Folgerungen herangezogen werden. Ein Schlussfolgerungssystem (engl. *Reasoner*) kann dabei eine Ontologie auf ihre Konsistenz prüfen oder aber durch Hinzunahme weiterer Ontologien den Wissensvorrat erweitern (Antoniou und Harmelen, 2004).

Um eine aufgestellte Behauptung zu beweisen sieht der Proof-Layer vor, dass eine *Heuristic Engine* das Semantic Web so lange nach Regeln und Ontologien durchforstet, bis eine Aussage entweder be- oder widerlegt werden kann. Die automatische Beweisführung kann sich allerdings als problematisch erweisen, da es unsicher ist, ob ein Beweis überhaupt erbracht werden kann[17]. Die automatische Beweisführung wird dabei von unterschiedlichen Einflussfaktoren beeinträchtigt, wie einer unzureichenden Menge an Regeln, oder der Unentscheidbarkeit, ob ein Ergebnis existiert.

Der Trust-Layer bildet abschließend die Voraussetzung, um die Integrität von Fakten wie sie in Ontologien aufgestellt werden und den darauf aufbauend gefolgerten Aussagen herzustellen. Die Hoffnung des Semantic Web liegt hierbei in der Realisierung des »Web of Trust«, in

[17] Dieses Problem ist äquivalent zu dem in der Theoretischen Informatik bekannten „Halteproblem“.

dem mit nur wenigen transitiven Schritten eine Vertrauensbeziehung zwischen zwei beliebigen Agenten beschrieben werden kann.

3.2 Wissenschaftliche Ansätze im Web Engineering

Da die Verwendung von Methoden aus der Softwareentwicklung nicht ausreichend für die Entwicklung und Evolution webbasierter Systeme ist, wurden methodische Vorgehensweisen speziell für den Entwurf von Web-Anwendungen entwickelt. Diese Ansätze bilden in ihrer Breite einen wesentlichen Teil des Forschungsbereichs Web Engineering. Sie finden sich in der Literatur sowie den einschlägig thematischen Konferenzen wie der »World Wide Web Konferenz« (WWW) und der »International Conference on Web Engineering« (ICWE) sowie in entsprechenden Workshops (IWWOOST) oder dedizierten wissenschaftlichen Journalen wie dem »Journal on Web Engineering« (JWE). Damit bilden sie den wissenschaftlichen Rahmen und das methodische Grundgerüst für unterschiedliche Bereiche und Prozessabschnitte bei der Entwicklung von Web-Anwendungen.

Aufgrund der Multidisziplinarität im Web Engineering werden diverse Methoden mit unterschiedlichen Modellierungsschwerpunkten diskutiert. Sie finden ihre Wurzeln in unterschiedlichen Bereichen, die alle Einfluss auf Entwurf und Konstruktion von Web-Anwendungen haben. Dementsprechend fokussieren sie auch verschiedene Entwicklungsaspekte von Web-Anwendungen. In Anlehnung an (Schwinger und Koch, 2003) lassen sich die methodischen Ansätze im Web Engineering dabei, je nach ihren unterschiedlichen ideologischen Schwerpunkten[18], in die folgenden Kategorien einordnen:

- **Hypertextorientierte Ansätze:** Hypertext-orientierte Methoden konzentrieren sich auf den Hypertext-Charakter von Web-Anwendungen und finden konsequenterweise ihren Ursprung im Bereich der Hypertext-Systeme. Beispiele für diesen Ansatz sind das Hypertext Design Model (HDM), das die Grundlage zu W2000 und HDM-Lite bildet, sowie die Web Site Design Method (WSDM).
- **Datenorientierte Ansätze:** Datenorientierte Modellierungsansätze kommen aus dem Bereich der Datenbanksysteme und bauen deshalb auf dem ER-Modell auf, das um spezielle Hypertext-Konzepte erweitert wird. Sie werden vornehmlich für die Entwicklung datenintensiver Web-Anwendungen eingesetzt. Bekannte Vertreter dieser Ansätze sind das Relationship Management Model (RMM) und die Web Modeling Language (WebML).
- **Objektorientierte Ansätze:** Die Objekt-orientierten Modellierungsmethoden haben ihren Ursprung entweder in OMT oder UML und erweitern diese gegebenenfalls um

[18] Ein ideologischer Schwerpunkt bestimmt den entsprechenden Hauptfokus, den eine dedizierte Methode in den Vordergrund rückt. Es findet also keine Ausschließlichkeit statt, sondern vielmehr eine ausgeprägte Affinität des entsprechenden Ansatzes zu grundlegenden Modellen, Werkzeugen und Herangehensweisen innerhalb eines Forschungsbereiches. Dabei herrscht innerhalb der Forschergemeinschaft über den *richtigen* Hauptschwerpunkt keine Einigkeit, was in der Vielschichtigkeit und Multidisziplinarität des Web Engineering begründet liegt.

eigene Konstrukte, wobei immer häufiger UML als Grundlage gewählt wird. Zu dieser Kategorie gehören die Object-Oriented Hypermedia Design Method (OOHDM), das UML-based Web Engineering (UWE) und die Object-Oriented-Hypermedia Method (OO-H).

Anhand dieser Kategorien werden nachfolgend jeweils prominente Vertreter einer entsprechenden Kategorie vorgestellt und abschließend anhand der in Kapitel 1 erarbeiteten Anforderungen an dienstorientierte Web-Anwendungen bewertet.

3.2.1 Hypertextorientierte Methoden

3.2.1.1 Hypertext Design Model (HDM)

Das Hypertext Design Model (HDM) stellt eine Strukturierungssprache auf Basis der Entity-Relationship (E-R) Methode (Chen, 1976) dar. Die Entwicklung von Web-Anwendungen wird mit Hilfe eines zweistufigen Modells beschrieben. Eine Anwendung besteht aus einer sogenannten Hyperbasis und einer Menge von Zugriffsstrukturen. Die Hyperbasis (*hyperbase layer*) stellt den Kern der Anwendung dar und besteht aus Elementen wie Entitäten, Komponenten, Einheiten und Verweisen. Die Zugriffsstrukturen (*access layer*) definieren wie die Knoten zusammenhängen und die Navigation untereinander gestaltet werden kann. Grundsätzlich unterscheidet HDM zwei unterschiedliche Entwurfsansätze, die eine unterschiedliche Granularität des Entwurfs fokussieren:

- Der »authoring-in-the-large«-Ansatz sieht den Entwurf globaler Klassen und Informationseinheiten sowie der Navigationsstrukturen vor. Dabei steht die Ausbildung der Anwendungsstruktur und ihrer topologischen Eigenschaften im Vordergrund.
- Der »authoring-in-the-small«-Ansatz fokussiert die Implementierungsdetails, die bei der Umsetzung von Struktur und Zugriffsstrukturen berücksichtigt werden müssen. Hierzu gehört Entwurf und Realisierung von einzelnen Knoten, Dokumenten und den Inhalten.

Die grundlegenden Elemente von HDM sind Entitäten (*entities*), Entitätsklassen (*entity types*), Komponenten (*components*), Einheiten (*units*), Verweise (*links*) sowie Verweistypen (*link types*). Sie repräsentieren autonome Informationseinheiten bzw. -strukturen, die aus einem Hypertext extrahiert werden können. Die Entitäten gehören zu Entitätsklassen, in denen ähnliche Entitäten zusammengefasst und gruppiert werden[19].

Das Hypertext Design Model stellt einen strukturellen Entwurf einer Anwendung dar. Für die Realisierung der Anwendung sind allerdings Mechanismen nötig, welche die grundlegenden

[19] Das folgende Beispiel verdeutlicht die einzelnen Bestandteile von HDM und seiner Relationen: Die Entität »Louvre« gehört zur Entitätsklasse »Museum«. Sie besteht unter anderem aus der Komponente »Mona Lisa«. Diese Komponente wiederum setzt sich aus den Einheiten »Gemälde« (visuelle Präsentation) und »Beschreibung« (textuelle Präsentation) zusammen. Eine einzelne Entität besteht aus einer oder mehreren hierarchisch aufgebauten Komponenten, die wiederum aus Einheiten bestehen. Diese Einheiten reflektieren die verschiedenen Perspektiven einer Komponente. Es sind also mehrere Darstellungen für die gleiche Information (Multimedialität) möglich.

Elemente von HDM auf Basis der so genannten *browsing semantic* in konkrete Implementierungsstrukturen umwandeln. Die *browsing semantic* legt fest, welche Informationsobjekte dem Benutzer angezeigt werden, welche Verweise es zwischen diesen Objekten gibt und was bei der Auswahl eines Verweises geschehen soll. Dadurch können aus einer HDM-Spezifikation zwar mehrere Anwendungen erstellt werden, allerdings existieren keine Mechanismen, mit denen man *browsing semantic* definieren und festlegen kann.

3.2.1.2 HDM-Lite

HDM-Lite (Fraternali und Paolini, 1998) stellt eine Weiterentwicklung von HDM (vergleiche Abschnitt 3.2.1.1), das um Web-spezifische Elemente erweitert wurde. HDM-Lite fokussiert neben den aus HDM bekannten Aspekten Struktur und Navigation auch die Präsentation einer Anwendung. Hierfür wird noch zusätzlich die Entität Präsentationsschema (*presentation schema*) eingeführt. Ein Präsentationsschema besteht aus einer Sammlung von sogenannten *Stylesheets*, die textuelle Beschreibungen des Erscheinungsbildes einer Web-Anwendung darstellen und in SGML-Syntax verfasst sind. Es kann dabei mehr als ein Präsentationsschema an die beiden anderen Schemata gebunden werden. Die Entwurfsgranularität einer Anwendung wird durch Seiten (*Page*) bestimmt. HDM-Lite verfügt über eine abgeschlossene Menge an Präsentationsstrukturen, die mit Hilfe des Autoweb Editors (Piero Fraternali und Paolini, 2000) zur Konstruktion von Web-Anwendungen herangezogen werden können. Dabei werden Implementierungsschablonen erstellt, die dann durch Entwickler entsprechend verfeinert werden können. Dadurch kann ein Teil der Web-Anwendung durch Spezifikation entsprechender konzeptioneller Modelle erzeugt werden und eignet sich damit gut zur Erstkonstruktion von Web-Anwendungen. Allerdings fehlt Unterstützung für die Phase der Evolution: Änderungen müssen direkt in den erzeugten Anwendungen durchgeführt werden, wodurch sich die ursprünglichen Modelle verfälschen. Als Alternative lassen sich Änderungen in den Modellen direkt durchführen und die Anwendungen neu erzeugen. Dadurch gehen allerdings Implementierungsdetails.verloren.

3.2.1.3 Web Site Design Method (WSDM)

Die Web Site Design Method (WSDM) (De Troyer und Leune, 1998) (Casteleyn, Troyer und Brockmans, 2003) stellt einen benutzerzentrierten (*user centered*) Ansatz vor. Dementsprechend bilden die Menge der zukünftigen Benutzer einer Webseite den Ausgangspunkt. Die Methode untergliedert sich in die folgenden Phasen:

User Modeling (*User Classification* und *User Class Description*):
Die Benutzermodellierung (*User Modeling*) umfasst zwei Aktivitäten. In der Benutzerklassifizierung (*User Classification*) werden die Benutzer einer Webseite identifiziert und kategorisiert. Hierfür wird in einem ersten Schritt darauf geachtet, für welche Organisationen die Webseite erstellt wird bzw. welche Prozesse sie unterstützen soll. Die ermittelten Organisationen und Prozesse werden dann in Aktivitäten aufgeteilt. Jede Aktivität involviert Menschen - die potenziellen Benutzer der Webseite. Diese Benutzer werden in einem weiteren Schritt in Klassen (*User Classes*) eingeteilt. Im zweiten Teil der Benutzermodellierung, der so genannten Benutzerklassenbeschreibung (*User Class Description*), werden die ermittelten Klassen noch detaillierter analysiert. Die Benutzer, die zu der gleichen Klasse gehören, haben vorwiegend das gleiche Informationsbedürfnis (*information requirement*), unterscheiden sich aber in der Art und Weise, wie ihnen diese Information dargestellt wird. In diesem Schritt werden die Charakteristika einer Klasse wie Nutzungshäufigkeit, Erfahrung, Sprache

oder Bildung bestimmt. Haben Benutzer innerhalb einer Klasse unterschiedliche Charakteristika, so wird die Benutzerklasse weiter in Perspektiven (*perspectives*) unterteilt.

Der *konzeptuelle Entwurf* umfasst eine Domänenmodellierung in Form von Objektmodellen und den Navigationsentwurf. In der Objektmodellierung werden mit Hilfe von Benutzerobjektmodellen die zuvor ermittelten Informationsbedürfnisse formal modelliert. Hierzu wird zu jedem Geschäftsobjekt und jeder Nutzergruppe ein entsprechendes Benutzerobjektmodell entwickelt. Die Objektmodelle beschreiben die verschiedenen Objekttypen, ihre Beziehungen untereinander, sowie Regeln und Einschränkungen auf den Objekten und Beziehungen. Die Wahl der Modellierungssprache bleibt offen, es wird jedoch OMT empfohlen. Eine Dienstorientierung durch Anbindung von Geschäftsprozessen ist genauso wenig vorgesehen, wie die Verwendung unterschiedlicher Datenquellen.

Um verschiedene Perspektiven innerhalb einer Benutzerklasse zum Ausdruck zu bringen können noch zusätzlich perspektivische Varianten der ursprünglichen Objekttypen erstellt werden. Im Navigationsentwurf wird ein konzeptuelles Navigationsmodell aus einer Reihe von Navigationspfaden, einer für jede Perspektive, konstruiert. Sie beschreiben, wie ein Benutzer einer bestimmten Perspektive über die verfügbaren Informationen navigieren kann. Dadurch entstehen eine Vielzahl an Modellen, welche die entsprechenden Eigenschaften der unterschiedlichen Nutzerklassen und deren Perspektiven zum Ausdruck bringen. Allerdings fehlen Ansatzpunkte zur Wiederverwendung solcher Modelle oder gar Teilaspekten, um das gesammelte Wissen über die einmalige Konstruktion einer Web-Anwendung hinaus zu konservieren.

Das Implementierungsmodell (*implementation model*) beschäftigt sich mit den »Look and feel«-Aspekten einer Webseite. Es dient der Vorbereitung der eigentlichen Konstruktion der Web-Anwendung. Im Site Structure Design werden abstrakte Seiten definiert, die den *Pageflow* bestimmen. Sie werden dann mit Hilfe eines Template Ansatzes unter Berücksichtigung der konzeptionell erarbeiteten Entwürfe realisiert. Offen bleibt, ob und wie eine Validierung der wichtigen Ergebnisse aus den Konzeptentwürfen vorgenommen werden kann und wie ein Entwickler über die grafischen Modelle hinaus unterstützt werden kann. Somit bleibt auch offen, wie die Evolution der entstandenen Web-Anwendung unterstützt werden kann.

3.2.2 Datenorientierte Methoden

3.2.2.1 Relationship Management Methodology (RMM)

Die Relationship Management Methodology (RMM) (Thomas Isakowitz, Stohr und Balasubramaninan, 1995) basiert auf dem in 3.2.1.1 vorgestellten Hypertext Design Model (HDM) und dem Entity-Relationship Modell. RMM unterteilt die Anwendungsentwicklung in sieben Phasen. Eine genaue Umsetzung der einzelnen Phasen wird nicht vorgegeben, allerdings wird darauf hingewiesen, dass sie zwingend berücksichtigt werden sollten. Grundlage der RMM bildet das Relationship Management Data Model (RMDM), das wiederum auf der E-R Modellierung basiert. Es stellt eine Sprache zur Verfügung, mit der man Informationsobjekte und Navigationsmechanismen einer Hypermedia-Anwendung beschreiben kann, allerdings sehr eingeschränkt auf die Verwendung einer relationalen Datenbank.

Eine Besonderheit in RMM sind die sogenannten m-slices[20]. Sie bezeichnen definierbare Bausteine und können auf Ebene der Anwendungsmodellierung durch Schachtelung wiederverwendet werden. Im *Slice Design* wird spezifiziert, wie die Information einer gewählten Entität präsentiert und der Zugriff realisiert wird. Dabei entsprechen die Slices Sichten über den Entitäten, die über Links verknüpft werden. Das im E-R-Design entwickelte E-R-Diagramm wird erweitert und als E-R+ Diagramm bezeichnet. Es modelliert die Navigation zwischen Slices mittels uni- und bidirektionalen Links, den so genannten strukturellen Links (*structural links*). Sie unterscheiden sich von assoziativen Beziehungen dadurch, dass sie Informationen innerhalb der gleichen Entität verknüpfen, während assoziative Beziehungen verschiedene Entitäten verbinden. Herausforderungen in dieser Phase sind das Einteilen einer Entität in Slices, die Auswahl eines Slices als Kopf der Entität, die Verlinkung dieser Slices sowie das Beschriften der Links.

In der Phase des Navigationsentwurfes (*navigation design*) werden die Pfade, welche die Navigation im Hypertext ermöglichen, entworfen. Dafür wird jede assoziative Beziehung im E-R+ Diagramm analysiert. RMM stellt drei musterhafte Navigationselemente zur Verfügung, die dies ermöglichen: *conditional indices*, *conditional guided tours* und *conditional indexed guided tours*.

RMM lässt allerdings unberücksichtigt wie Benutzereingaben verarbeitet werden. Die m-slice Bausteine können zwar zur Wiederverwendung hergezogen werden, allerdings sind Assoziationen zwischen m-slice-Bausteinen fest kodiert und erlauben so keine flexible Einbettung in andere Bausteine. Dadurch kann zwar eine automatische bottom-up-Generierung durch Nutzung dieser Informationen herangezogen werden, um ein Anwendungsskelett zu erzeugen (Tomas Isakowitz, Kamis und Koufaris, 1998), die allgemeine Wiederverwendbarkeit ist allerdings dadurch eingeschränkt. Die Abbildung des RMDM-Diagramms ist darüber hinaus nicht reversibel, so dass sich die Evolution der Anwendungen nur auf Daten einer zuvor definierten Struktur bezieht.

[20] Dabei führt sich Abkürzung »m« auf die Bezeichnung der russischen Puppen »Matrjeska« zurück. Diese Puppen haben die Eigenschaft, dass sie ineinander gestapelt werden können.

3.2.2.2 Web Modeling Language (WebML)

WebML ist eine datengetriebenes Methodik zum Entwurf von Web-Anwendungen und wurde maßgeblich durch Stefano Ceri, Piero Fraternali und Aldo Bongio spezifiziert (Ceri, Fraternali und Bongio, 2000). Insgesamt werden sechs verschiedene Modelle unterschieden, die jeweils aufeinander aufbauen und sich ergänzen können. Das *Strukturmodell* definiert die für die Webseite relevanten Entitäten und Beziehungen. WebML zielt auf einen allumfassenden Informationsansatz, d.h. alle Informations- und Datentypen, die eine Web-Anwendung bestimmen, sind a priori bekannt.

Im *Hypertext-Modell*, das aus dem Kompositionsmodell (Composition Model) und dem Navigationsmodell (Navigation Model) besteht, werden Sichten (views) auf die Website definiert. Im *Kompositionsmodell* wird der Hypertext in Seiten (pages) aufgeteilt und die Einheiten (content units) beschrieben, aus denen die Seiten bestehen. WebML unterscheidet sechs verschiedene Typen solcher Kompositionseinheiten: Dateneinheiten (data units), Multi-Dateneinheiten (multi-data units), Indexeinheiten (index units), Filtereinheiten (filter units), Scrolleinheiten (scroller units) und direkte Einheiten (direct units). Diese Einheiten sind sehr gut geeignet, um Daten strukturiert darzustellen oder über Datenmengen zu iterieren. Allerdings finden sich keine Erweiterungsmechanismen, die zusätzliche Definitionen erlauben. So bleibt die Frage nach der Integration von Bearbeitungsprozessen für die Verarbeitung von Benutzereingaben ungeklärt. Die zunächst gewonnene Einfachheit und Abgeschlossenheit des WebML Ansatzes verliert hinsichtlich Erweiterungen zusehends an Bedeutung, wenn es um die Realisierung fortgeschrittener Web-Anwendungen geht, die nicht direkt zu der ursprünglich rein datenorientierten Sicht passen (Clemens Kerer, 2003). Jüngste Erweiterungen hinsichtlich der Generierung Workflow-basierter Anwendungen auf Basis von BPMN-Modellen erweitern WebML um eine Modellebene mehr. Allerdings wird nur eine Abbildung auf das Hypertextmodell vorgenommen. Die Frage nach fachlicher Integration von Geschäftsprozessen oder funktionalen Komponenten bleibt allerdings weiterhin offen. Sie werden lediglich in Form eines Knotens im Hypertext-Modell charakterisiert, dessen *».. operations are conceived for connecting application object(s) to an activity instance, typically to specify the future job to be performed on it.«* Somit wird die eigentliche Integration von fachlichen Komponenten oder Anwendungsbausteinen in die Implementierungsphase verlegt.

Im *Navigationsmodell* wird festgelegt, welche Seiten und Einheiten verknüpft werden, um den Hypertext zu formen. WebML unterscheidet kontextuelle (Unit-verbindende) und nicht-kontextuelle (Seiten-verbindende) Verweise. Das übersichtlich gehaltene *Präsentationsmodell* besteht aus zwei Kategorien von Stylesheets, den seitenspezifischen und den allgemein gültigen. Der Fokus liegt dabei auf der verbesserten Wiederverwendbarkeit von Stylesheets. Allerdings können weder Eigenschaften anderer Modelle von WebML für das Präsentationsmodell herangezogen werden, wodurch die Komplexität des Entwurfs komplett in die Implementierungsphase fällt. Zusätzlich bleibt die Frage offen, wie eine Unterstützung der Barrierefreiheit in WebML gewährleistet werden kann.

Von seiner grundsätzlichen Prägung ist WebML stark auf eine datenbankorientierte Konstruktion von Web-Anwendungen ausgelegt. Das bedeutet, eine klar umrissene und abgeschlossene Informationsdomäne wird durch WebML gut abgedeckt. Die Konstruktion von Web-Anwendungen folgt dabei allerdings grundsätzlich den Daten, nicht den Funktionen; eine Modellierung von dienstorientierten Web-Anwendungen wird dadurch deutlich erschwert. Der wichtige Aspekt einer Integration bestehender Anwendungslogik durch Dienste (Kreger, 2003) wird damit durch WebML nur unzureichend berücksichtigt. In (Brambilla, Ceri,

Comai, Fraternali et al., 2003) werden zwar Ansätze einer Modellerweiterung von WebML diskutiert, die es gestattet auf Basis konkreter Request-Response Kombinationen Web Services aufzurufen. Die eingeführten Spracherweiterungen entsprechen dabei den in WSDL definierten Service Typen. Dadurch wird WebML grundsätzlich befähigt Web Services zu integrieren. Die Wiederverwendung und Konvertibilität werden allerdings dadurch erschwert, dass Teile der Nachricht direkt im Entwurf spezifiziert werden müssen. Für die Anwendungsintegration führt WebML die Konzepte *Materializer* und *Dematerializer* ein, deren Aufgabe in der Überführung komplexer Web Service Nachrichten in das eigens dafür eingeführte XML-ER bestehen, einer kanonischen XML Spezifikation für Entity Relationship Schemas. Der Fokus der Web Service Einbindung liegt also lediglich auf einer datenorientierten Ergänzung und technischer Integration und weniger auf der fachlichen Integration von Anwendungslogik und Geschäftsprozessen.

3.2.3 Objektorientierte Methoden

3.2.3.1 Object-Oriented Hypermedia Design Method (OOHDM)

Die Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe, Rossi und Barbosa, 1996; Schwabe und Rossi, 1998; Schwabe, Pontes und Moura, 1999) stellt einen Objektorientierten Ansatz für den Entwurf hypermedialer Anwendungen dar. Es basiert auf dem hypertext-orientierten Ansatz HDM (siehe Abschnitt 3.2.1.1). Die Methode hebt sich von HDM durch die Hinwendung zur Objektorientierung ab und stellt ähnlich wie RMM (siehe Abschnitt 3.2.2.1) spezielle Modellierungsprimitive sowohl für den Navigations- als auch für den Präsentationsentwurf bereit. Der Unterschied zwischen den beiden Ansätzen besteht zum einen in der Einführung sogenannter Navigationskontexte (navigational contexts). Zum anderen verwendet OOHDM eine explizite Modellierung der Benutzungsschnittstelleninteraktion (Abstract Data Views, ADV). OOHDM gliedert sich in die vier Phasen konzeptueller Entwurf (conceptual design), Navigationsentwurf (navigational design), abstrakter Schnittstellenentwurf (abstract interface design) und Implementierung (implementation).

Im konzeptuellen Entwurf wird mittels Objekt-orientierter Modellierungstechniken ein Domänenmodell erstellt. Das Ergebnis bilden ein Klassen- und ein Instanzschema, die jeweils aus Subsystemen, Klassen und Beziehungen bestehen. Sie reflektieren, aus welchen Objekten die Anwendungsdomäne besteht und in welchen statischen Beziehungen sie zueinander stehen. Das daraufhin entwickelte Navigationsmodell stellt eine kontextabhängige Sicht auf das zuvor entwickelte konzeptuelle Modell der Anwendungsdomäne. Zu einem Domänenmodell können mehrere Navigationsmodelle, also mehrere Sichten, existieren.

Um die semantischen Bezüge gerade zwischen dem Domänenmodell und der Navigation besser unterstützen zu können, wurde OOHDM um den Einsatz von Beschreibungssprachen des Semantic Web (siehe 3.1.5.1) ergänzt. Die Semantic Hypermedia Design Method (SHDM) (Lima und Schwabe, 2003) (Schwabe, Szundy, Moura und Lima, 2004) stellt eine Methode für das Design und die Implementierung von Web-Anwendungen im Semantic Web zur Verfügung. SHDM bildet die in UML modellierten OOHDM Konzepte mit Hilfe von Heuristiken in OWL Dokumente der Komplexitätsstufe DL ab.

Das Hauptaugenmerk der Methodik liegt auf dem Navigationsmodell. Das Navigationsdesign untergliedert sich in die Entwicklung des Navigationsklassenschemas (navigational class schema) und des Navigationskontextschemas (navigational context schema). Im Klassen-

schema werden die navigierbaren Objekte mit den gleichen Modellierungstechniken wie im Domänenmodell definiert. Es reflektiert daher die Sicht auf die Anwendungsdomäne. Wie in HDM und RMM definiert auch OOHDM eine Menge vordefinierter Typen von Navigationsklassen:

- Knoten (nodes) sind als Objekt-orientierte Sicht auf die im konzeptuellen Entwurf definierten Klassen zu betrachten.
- Verweise (links) repräsentieren die Beziehungen im konzeptuellen Modell.
- Zugriffsstrukturen (access structures) definieren komplexere Mechanismen wie Indizes oder Guided Tours.

Dadurch können a priori bekannte Datenquellen gut modelliert und Zusammenhänge zwischen einzelnen Klassen abgebildet werden. So lassen sich auch hypermediale Entwurfsmuster im Navigations-Kontext-Schema gut modellieren. Allerdings bietet OOHDM keine Abbildungsvorschriften oder wenigstens Implementierungshinweise an, *wie* solche Modelle umgesetzt werden. In der Implementierungsphase muss also zusätzlich analysiert werden, in welcher Weise die Objekte in Verbindung stehen und über welche Zugriffstruktur diese realisiert werden kann. Entwickler müssen also verstehen, wie ein Benutzer den Informationsraum durchsuchen möchte, um zu verhindern, dass er redundante Informationen vorfindet oder sich im Informationsraum verliert. Aufgrund des Bruchs zwischen Konzeptentwurf und der Implementierung wird auch die Evolution der Web-Anwendung erschwert, da alle Phasen erneut durchlaufen werden müssen, was faktisch einer Neuentwicklung gleichkommt.

Im abstrakten Schnittstellenentwurf wird die Benutzungsschnittstelle mit Hilfe von Abstract Data Views (ADV) (Cowan und Lucena, 1995) konzipiert. Der Entwurf legt fest, wie die Navigationsobjekte in einem bestimmten Navigationskontext dargestellt werden. Da in OOHDM eine klare Trennung von Navigation und Darstellung verfolgt wird, kann es auch mehrere Schnittstellenmodelle für einen Navigationsentwurf geben. Allerdings haben die ADV Diagramme nur konzeptionellen Charakter und haben für die Implementierung der resultierenden Web-Anwendung lediglich informellen Charakter, da weder Werkzeuge noch Abbildungsvorschriften existieren (Schwabe, Pontes und Moura, 1999).

Ein weiteres Problem von OOHDM betrifft die Erweiterbarkeit der Modellierungsgrundlage. So offeriert die Methode keinen Ansatz zu einer Erweiterung des zugrunde gelegten Informationsraums im Zuge der Evolution einer Web-Anwendung. Dies gilt sowohl hinsichtlich der unterstützten Navigationsmodelle, als auch einer Integration oder Wiederverwendung von fachlichen Komponenten oder Geschäftsprozessen.

Der hohe Abstraktionsgrad von OOHDM ermöglicht eine vielseitige Verwendung in unterschiedlichen Anwendungsdomänen. OOHDM verfügt zwar über gut ausgeprägte Konzeptualisierungsphase, vernachlässigt darüber aber die Phasen wie Wartung und Evolution. Eine orthogonale Trennung der Modelle für verschiedene Aspekte wird zwar auf konzeptueller Ebene erreicht, verliert sich aber nach der Implementierungsphase in der Content, Darstellungsinformationen und Anwendungslogik wiederum vermischt werden. Wiederverwendung kann dadurch lediglich bis zu einem gewissen Grad durch die Trennung von Konzeptionellem Entwurf und Navigationsklassen auf Modellebene erreicht werden.

3.2.3.2 UML-based Web Engineering (UWE)

UML-based Web Engineering (UWE) (N. Koch, Kraus und Hennicker, 2001) stellt eine Objekt-orientierte Entwurfsmethode für eine systematische Entwicklung von Web-Anwendungen

dar. Dieser Ansatz vertraut durchgehend auf UML als Modellierungsgrundlage. Er untergliedert sich in die vier Phasen Anforderungsanalyse, konzeptueller Entwurf, Navigationsentwurf und Präsentationsdesign. Die einzelnen Modelle bauen jeweils aufeinander auf (vgl. hierzu Abbildung 3-6).

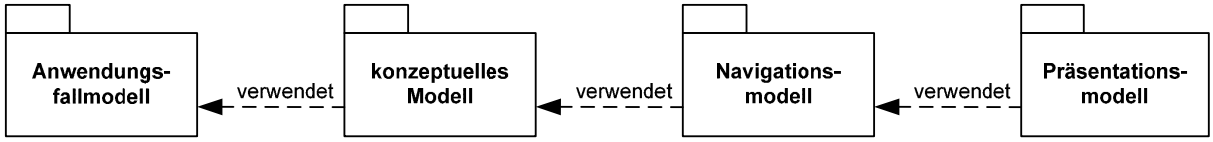


Abbildung 3-6: Modellierungsreihenfolge der aufeinander aufbauenden Modelle

In der Anforderungsanalyse werden mit Hilfe von UML-Anwendungsfalldiagrammen die Anforderungen an die Anwendung bestimmt. UWE empfiehlt, sich in dieser Phase auf die Diagrammtypen Anwendungsfall, Aktivitäts und Klassendiagramm zu konzentrieren. Anschließend wird aus den Anwendungsfällen ein konzeptuelles Modell abgeleitet, das alle Entitäten abbildet, die in typischen Aktivitäten involviert sind, wie sie der Benutzer mit der Webanwendung durchführt. Das konzeptuelle Modell wird als UML-Klassendiagramm realisiert. Basierend auf dem konzeptuellen Modell wird das Navigationsmodell entwickelt, welches den Navigationsraum und die Zugriffsstrukturen, die für die Navigation genutzt werden können, repräsentiert. Im Präsentationsentwurf wird basierend auf dem Navigationsstrukturmodell und den zusätzlich in der Anforderungsanalyse gesammelten Informationen ein Präsentationsmodell entwickelt. Hier wird eine strukturelle Organisation der Präsentation in Form von Kompositionsstrukturdiagrammen modelliert, die eine Anordnung[21] von bestimmten in UWE vordefinierten stereotypisierten Klassen wie «form» oder «text» erlaubt. Ähnlich zu den ADV in OOHDM kann hier aber lediglich eine abstrakte Darstellung schematisiert werden. Die Umsetzung von Design und Layout wird dabei auf die Implementierungsphase verlagert.

UWE unterstützt keine Dienstorientierung. Dies gilt zum einen für die Einbindung von verteilten Diensten, wie beispielsweise Web Services, im Sinne der Anwendungsarchitektur. Zum anderen sieht UWE keine Möglichkeit vor, Strukturelemente zu definieren, die in loser Kopplung mit einander kommunizieren bzw. interagieren. Hier wird lediglich im Navigationsmodell die Möglichkeit zur Strukturbildung in Seiten («navigation class») und Verweise («direct navigability») gegeben. Prozesse lassen sich zwar mittels dafür vorgesehener Modelle spezifizieren (Nora Koch, Kraus, Cachero und Melià, 2003), allerdings bleibt unklar, wie eine automatisierte Abbildung auf Komponenten oder deren Wiederverwendung erreicht werden kann.

Die Evolution wird durch UWE nur bedingt unterstützt. Es bleibt unklar, wie durchgeführte Änderungen in einer bereits ausgelieferten Web-Anwendung so konserviert werden können, dass die durch Evolution bedingten Änderungen mit den entsprechenden Modellen konsistent gehalten werden können. UWE bietet zwar eine Werkzeugunterstützung (Knapp, Koch,

[21] Eine Festlegung der räumlichen Anordnung von Elementen eines Kompositionsstrukturdiagramms ist allerdings kein Bestandteil der UML Superstructure Specification (http://www.uml.org/) und wird daher im Allgemeinen von UML Werkzeugen nicht unterstützt.

Zhang und Hassler, 2004) an, allerdings können hier nur Teile der Modelle zur Anwendungserzeugung herangezogen werden. Es bleibt also ein Bruch zwischen der Modellebene und der Anwendungsebene. Obwohl die Wichtigkeit einer Wartungsphase betont wird, bleibt unklar *wie* die kontinuierliche Weiterentwicklung unterstützt werden soll.

3.2.3.3 Object-Oriented-Hypermedia Method (OO-H)

Die Hauptbestandteile der Object-Oriented-Hypermedia Method (OO-H) (Gomez, Cachero und Pastor, 2001) sind das navigational access diagram (NAD), das eine Navigationssicht auf die zu entwickelnde Anwendung darstellt, und das abstract presentation diagram (APD), mit dem Präsentationsaspekte modelliert werden. Beide Diagramme greifen schnittstellenbezogene Entwurfsinformationen mit Hilfe einer Menge von Mustern (patterns) auf, die in einem Katalog (pattern catalog) festgehalten sind. Die Methode besteht damit aus einem Entwurfsprozess (design process), einem Musterkatalog (pattern catalog), einem NAD, einem APD und einem CASE Tool, mit dem aus der modellierten Anwendung ein Implementierungsskelett generiert werden kann.

Eine Besonderheit des OO-H Ansatzes bildet das Vorhalten eines Musterkatalogs. Die in dem Katalog enthaltenen Muster bieten alternative Lösungen zu bekannten Hypermedia-Problemen an, welche die verschiedenen Sichtweisen der Benutzer berücksichtigen. Dadurch kann die für eine bestimmte Anwendungsdomäne passende ausgewählt werden. Die Muster im Musterkatalog lassen sich grob in drei Kategorien unterteilen:

- Informationsmuster (information pattern) stellen dem Benutzer hilfreiche Kontextinformationen der Anwendung zur Verfügung.
- Interaktionsmuster (interaction pattern) betreffen Kommunikationsaspekte an der Benutzungsschnittstelle, die sowohl Funktionalität als auch Navigation involvieren.
- Benutzerschema-Evolutionsmuster (user schema evolution pattern) hingegen befassen sich mit erweiterten Features, welche die Struktur betreffen.

Dieser Musterkatalog kann bei der Erstellung sowohl der Navigationsdiagramme (NAD) als auch der Präsentationsdiagramme (APD) verwendet werden. Für NADs können Muster angewendet werden, welche die Selektion der Benutzerinformationen und das Navigationsverhalten betreffen, während für APDs Muster bezüglich der Benutzungsschnittstelle interessant sind und die Benutzbarkeit der Anwendung verbessern.

Das APD besteht aus Templates, die das Erscheinungsbild und die Struktur der Anwendung bestimmen. OO-H hat hierfür fünf verschiedene Templatetypen definiert, die in Form von XML-Dokumenten vorliegen. Für jeden Templatetyp existiert auch eine DTD bzw. ein XML Schema. OO-H gibt eine Menge von Regeln vor, mit dem die verschiedenen im NAD enthaltenen Elemente auf äquivalente Strukturen im APD transformiert werden können. Dieses rudimentäre APD wird anschließend noch weiter verfeinert. Ein Problem bezüglich der Flexibilität stellt hierbei die fehlende Verknüpfung zwischen dem Informationsraum und den verwendeten NAD und APD dar. So werden lediglich auf Basis konzeptueller Modelle (UML Klassen) die Spezifikationen von Navigations- und Präsentationsmustern durchgeführt. Die Verknüpfung mit entsprechenden Datenquellen des Informationsraums wird erst *nach* der Modelltransformation durch den Compiler – also in der Implementierung – vorgenommen. Es bleibt unklar, wie die Verknüpfung zwischen einem APD und den entsprechenden Datenquellen in der Anwendungslogik kombiniert werden. Dementsprechend findet keine nahtlo-

se Modelltransformation statt, was eine systematische und werkzeuggestützte Evolution erschwert.

Abbildung 3-7 stellt die einzelnen Teile und Schritte von OO-H im Überblick dar. Wie man daran deutlich erkennen kann bietet OO-H zwar die Möglichkeit Teile des Modells automatisiert auf programmierbare Schnittstellen und Templates abzubilden, die dann entsprechend angepasst werden können. Fraglich bleibt aber, wie die Evolution der in einem Schritt erzeugten Anwendung unterstützt werden kann. Außerdem bleibt offen, wie fachliche Prozesse eingebunden oder Wiederverwendung ermöglicht werden kann. Durch die Verlagerung des *konkreten* Benutzungsschnittstellendesigns auf die Implementierung wird auch eine Unterstützung für ein proaktives Barrierefreies Design beispielsweise durch Hinzunahme der Modellierungsartefakte erschwert und kann nur im Nachhinein durch reaktive Mechanismen geprüft werden.

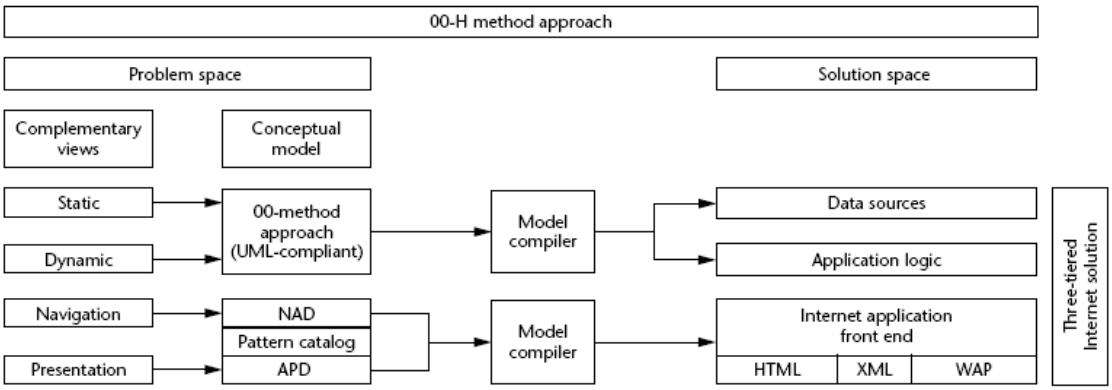


Abbildung 3-7: Das zugehörige Werkzeug zur Modellierung von Web-Anwendungen und Generierung von Schnittstellen

3.3 Zusammenfassung und Bewertung

Die Bewertung der existierenden Ansätze bezüglich der im Abschnitt 2.3 aufgestellten Anforderungen an die Entwicklung und Evolution dienstorientierter Web-Anwendungen ist zum besseren Vergleich in tabellarischer Form (vgl. Tabelle 3-2) dargestellt. Die wesentlichen Ergebnisse dieser Untersuchung, die zu dieser Bewertung geführt haben, sind in den folgenden Punkten festgehalten:

- **Fehlende Berücksichtigung der Dienstorientierung – kein Architekturansatz**: Bei der Analyse der unterschiedlichen Methoden zur Modellierung von Web-Anwendungen fällt auf, dass eine dienstorientierte Entwicklung keine Berücksichtigung findet. In einzelnen Ausnahmen, wie beispielsweise WebML finden sich erste Ansätze zur *Verwendung* von Web Services. Allerdings schlägt sich dies lediglich aus datenorientierter Sicht in einer Erweiterung der Datenbank nieder und weniger auf eine fachliche Integration (siehe Abschnitt 3.2.2.2). Insgesamt finden sich die für Web-Anwendungen relevanten Aspekte Daten und Hypertext sehr ausgiebig behandelt. Genauso wird die Entwurfsdimension von Web-Anwendungen durch Objektorientierte Methodiken gut abgedeckt. Die mangelnde Berücksichtigung der Dienst-

orientierung liegt hauptsächlich in der Art und Weise begründet wie diese Methoden aufgebaut sind. Eine systematische Verwendung von Diensten sieht vor, dass relevante Aspekte wie Daten, Navigation, Dialog und Darstellung eigenständig – im Sinne des Separation of Concerns (Dijkstra, 1982a) – behandelt werden können. Dazu bedarf es geeigneter Modellierungs- und Beschreibungsformen sowie einer holistischen und architekturbezogenen Anwendungssicht, um diese Aspekte gezielt bei Entwicklung *und* Evolution wiederverwenden zu können.

- **Forward Engineering – Neuentwicklung und keine Evolution:** Aktuelle Ansätze des Web Engineering fokussieren vornehmlich die Entwicklung einer Web-Anwendung und vernachlässigen ihre Wartung, Weiterentwicklung und Anpassbarkeit. Das wird besonders an Vertretern der Daten- und Objekt-orientierten Methodiken (siehe Abschnitte 3.2.2 und 3.2.3) deutlich. Hier finden sich klare Phasenabläufe – ähnlich dem Wasserfallmodell – die auf die zumeist einmalige Erzeugung von Implementierungsvorlagen abzielen. So legt OO-H zwar sehr viel Wert auf die Verwendung von Mustern aus denen diese Vorlagen erzeugt werden können. Die Anwendungslogik – im Sinne einer dienstorientierten Web-Anwendung also die Verknüpfung mit Diensten – muss dann im Nachhinein allerdings »hinzuprogrammiert« werden. Eine Änderung am Modell zieht somit zwangsläufig eine Änderung der Erzeugung mit sich und erschwert die Verwendung bereits existierendenden Programmcodes durch Anpassungen in den Vorlagen erheblich. Gerade in Hinsicht auf die Realisierung von dienstorientierten Web-Anwendungen stellen aber Anforderungen nach lose gekoppelten, wieder verwendbaren Diensten erhebliche Anforderungen an eben diese Phasen. So spielt gerade die Evolution von Diensten eine wichtige Rolle, da in der Regel eine komplette Neuentwicklung nicht wünschenswert ist. Vielmehr sollen nur die relevanten Teile einer Web-Anwendungen verändert – und so den aktuellen Erfordernissen – angepasst werden.
  Wünschenswert ist also eine Architektur, die im Zuge der Evolution mit Hilfe von passenden Modellen und Methoden zu einer dedizierten konvertiblen Anwendung ausgeprägt werden kann. Somit herrscht zwischen den Modellen einerseits und der Anwendung andererseits eine permanente Harmonie.

- **Fehlende Flexibilisierung der schwergewichtigen Entwurfsmethoden:** Ein wesentliches Problem bei den betrachteten Ansätzen liegt in der Schwierigkeit, eine ausgewogene Verhältnismäßigkeit zwischen beteiligten Stakeholdern wie Kunden, Nutzern oder Entwicklern zu definieren. So bergen beispielsweise rein modellgetriebene Entwicklungsmethoden wie UWE oder OOHDM oftmals die Gefahr, ein abstraktes Modell durch ein anderes zu ersetzen ohne dabei die Aussagekraft zu erhöhen (Tolvanen, 2006); dies gilt umsomehr, wenn das Ziel verfolgt wird, lauffähigen Programmcode zu erzeugen. Gerade aber in SOA-Projekten, in denen die Projektmitglieder oftmals sogar organisationsübergreifend miteinander kommunizieren müssen, bedarf es geeigneter und flexibler Entwicklungsmethoden. Für wenig erfahrene Personengruppen im Bereich der Softwareentwicklung stellt sich der notwendige Lernaufwand für diese schwergewichtigen Prozesse oftmals als unverhältnismäßig hoch dar. Dies wird durch die Tatsache verstärkt, dass die meisten Entwickler von Software gar keine ausgebildeten Softwareingenieure sind. Es besteht also die Notwendigkeit für eine technisch-methodische Unterstützung, die Hand in Hand mit einer flexiblen und kommunikationsfördernden Herangehensweise geht.

- **Fehlende Unterstützung für das Präsentationsdesign:** In der überwiegenden Zahl der betrachteten Ansätze wird das Präsentationsdesign komplett in die Phase der Implementierung verlagert (siehe beispielsweise WSDM, RMM, HDM). Vereinzelte Ausnahmen wie OOHDM, WebML oder UWE untersützen zwar den abstrakten Entwurf von Darstellungsstrukturen im Sinne von Schablonen. Allerdings lässt sich auf diese Weise kein durchgängiges Präsentationsdesign unterstützen; es wird höchstens wie im Fall von WebML die Struktur von einzelnen Anwendungsbereichen fokussiert. Konsequenterweise werden präsentationsrelevante Aufgaben wie Ästhetik und Zugänglichkeit in die Implementierung nachverlagert. Das geht Hand in Hand mit der Sichtweise, dass die Präsentationsschicht im Sinne einer 3-tier Architektur eine eigenständige Schicht bildet. Dadurch kommt es zu einer grobgranularen Abbildung der vormals oft sehr feingranular betrachteten Aspekte wie *Content* und *Navigation*. Aufgrund der fehlenden Reversibilität der vorgestellten Methoden, die mit einer vernachlässigten Betrachtung der Evolution einhergeht, lassen sich Änderungen an den Entwurfsartefakten nicht mehr durchgängig durchführen. Dadurch kann die Darstellung nicht *dediziert angepasst*, *verändert* oder *stabilisiert* werden.
- **Barrierefreie Benutzungsschnittstelle**: In engem Zusammenhang mit einer unzureichenden Unterstützung für das Darstellungsdesign stehen Qualitätsaspekte wie eine barrierefreie Benutzungsschnittstelle. In Hinsicht auf die Gewährleistung und Durchsetzung der Barrierefreiheit fehlen daher konsequenterweise Ansätze und zumeist sogar Ansatzpunkte zu deren Unterstützung. Zwar wird das Problem inzwischen vereinzelt erkannt, aber durch die konsequente Verlagerung des Präsentationsdesign in die Implementierungsphase lassen sich *stabilisierende* Änderungen nur im Sinne des Forward Engineering lösen und nicht als Entwurfsartefakt im Entwicklungsprozess wieder verwenden. Forderungen nach einer methodischen Unterstützung zur Qualitätssteigerung der Nutzerschnittstelle existieren (G. Costagliola, F. Ferrucci, C. Gravino, G. Tortora et al., 2004), finden sich in den vorgestellten Ansätzen aber nicht berücksichtigt. Das Thema und seine stark wachsende Bedeutung wird dabei gleichwohl in Forschung, Gesellschaft und Politik erkannt, was in einschlägigen Veranstaltungen wie dem W4A Workshop oder der aktuellen Gesetzgebung von Industrienationen wie den USA mit Section 508 (ITAW, 1998), Japan JIS X 8341-3, (Japanese Standards Association, 2004) oder Deutschland (BITV) (Bundesministerium der Justiz, 2002) erkennbar ist.

Die Tabelle 3-2 stellt die aufgeführten Punkte noch einmal detailliert vor, wobei die zuvor vorgestellten Ansätze anhand der aufgestellten Anforderung mit den Kriterien aus Tabelle 3-1 bewertet werden.

Anforderung	Bedeutung
SRV-LK	Lose Kopplung zwischen Diensten. Das betrifft einerseits die Dienste der Benutzungsschnittstelle, als auch verteilte Dienste im Sinne der Web Services.
SRV-FED	Homogener Zugriff auf den Informationsraum der Web-Anwendung sowie die Fähigkeit zur Realisierung einer organisationsübergreifenden Integration von Diensten.
RU-KOMP	Komponentenorientierte Vorgehensweise zur effizienten Steigerung der Wiederverwendung bei der Konstruktion von Web-Anwendungen.
RU-KONV	Fähigkeit zur Konvertibilität einer Anwendung durch Konfiguration in Form von Anpassungen spezifischer Eigenschaften bis hin zum dedizierten Austauschen einzelner Komponenten.
EVOL-RUN	Unterstützung eines *composition frameworks* zur laufzeit-gestützten Evolution von Web-Anwendungen.
EVOL-KPZ	Unterstützung von kurzen Produktionszyklen, um agil auf Änderungen reagieren zu können.
EVOL-FLEX	Ein flexibler Entwicklungsprozess, der unterstützt durch kurze Produktionszyklen, agil und situationsbedingt agieren kann.
UIX-DIAG	Ein orthogonaler Dialogentwurf, der Interaktion mit Formularen und Gestaltung der Benutzungsschnittstelle unterstützt.
UIX-PRES	Ein orthogonaler Präsentationsentwurf, der die spezifischen Charakteristika einer Benutzungsschnittstelle hinsichtlich Barrierefreiheit und ästhetischem Design unterstützt.
UIX-NAV	Unterstützung der musterbehafteten Navigation in Web-Anwendungen mit dem Fokus auf die Verwendung von Diensten.

Wert	Bedeutung
-	Wird nicht unterstützt / Erfüllt die Anforderung nicht
0	Wird bedingt unterstützt / Erfüllt die Anforderung teilweise
(+)	Wird teilweise gut unterstützt / Erfüllt die Anforderung teilweise gut
+	Wird gut unterstützt / Erfüllt die Anforderung gut

Tabelle 3-1: Anforderungsabkürzungen und Bewertungskriterien

		Dienstunterstützung		Wiederverwendung		Evolution			Benutzungsschnittstelle		
		SRV-LK	SRV-FED	RU-KOMP	RU-KONV	EVOL-RUN	EVOL-KPZ	EVOL-FLEX	UIX-DIAG	UIX-PRES	UIX-NAV
Data	RMM	-	-	(+)	-	-	(+)	-	-	0	(+)
	WebML	0[22]	(+)	-	-	-	(+)	0	-	-	+
Hypertext	HDM	-	-	-	(+)	-	0	-	-	-	+
	HDM-lite	-	-	-	(+)	-	-	-	-	0	(+)
	WSDM	-	-	-	(+)	-	(+)	-	-	0	+
Objekt-Orientiert	OOHDM	-	-	-	-	-	(+)	0	0	-	+
	UWE	-	0	(+)	-	-	(+)	-	0	0	+
	OO-H	(+)	-	(+)	0	-		0	-	0	+

Tabelle 3-2: Überblick der vorgestellten wissenschaftlichen Methoden bezogen auf die Anforderungen an dienstorientierte Web-Anwendungen

[22] WebML verfügt zwar über ein Composition Model, allerdings werden hier lediglich auf Ebene der Konzeption Einheiten (insgesamt sechs) identifiziert, die dann in einer konkreten Sprache grobgranular abgebildet werden. Ein Beispiel dafür sind Pages und Units, die in Seiten und Frames in der Zielsprache HTML abgebildet werden können.

4 Informationsraum-Entwurf

In diesem Kapitel wird der Zugriff auf den Informationsraum durch Dienste entwickelt, der den Ansprüchen an die gesetzten Anforderungen entspricht. Um die Wiederverwendbarkeit von Dienstschnittstellen zu erhöhen, werden universale Schnittstellen konzipiert, die den Zugriff auf den zugrunde gelegten heterogenen Informationsraum homogenisieren. Dabei werden speziell Dienstspezifikationen untersucht, die geeignet sind, eine dynamische Interaktion mit Web-Anwendungen effizient zu gewährleisten. Neben der Lösung technischer Aspekte wie Sicherheit und Interoperabilität von Diensten spielen auch nicht-funktionale Anforderungen wie die Granularität von Dienstschnittstellen eine Rolle. Eine Dienstspezifikation setzt sich aus einer Menge von Dienstprimitiven und Regeln zusammen. Eine grundlegende in dieser Arbeit betrachtete Dienstspezifikation orientiert sich an Dienstprimitiven zur Verwaltung und Manipulation des Informationsraums einer Web-Anwendung, die mit Hilfe von Web Services umgesetzt werden.

Die Evolution von Dienstspezifikationen vollzieht sich hierbei in zwei klar definierten Richtungen. Während die horizontale Evolution eine Erweiterung durch Hinzufügen weiterer Dienstspezifikationen darstellt, wird die vertikale Evolution durch eine fortwährende Weiterentwicklung und Anpassung einer Dienstspezifikation an ihre sich wandelnde Umgebung ermöglicht. Dies wird durch dedizierte Übergabekontexte (parametrisierte Polymorphie) erreicht, die entsprechend vorherrschender Anforderungen angepasst werden können.

4.1 Ausgangspunkt

In seinem Proposal zum World Wide Web (WWW) hat sein Erfinder Tim Berners-Lee die Vision eines globalen Informationsraums, in dem Computer *überall* auf der Welt Informationen speichern und sie *jedermann* verfügbar machen, beschrieben (Tim Berners-Lee, 1990). Diese Datensenken oder auch Informationsquellen können hierbei von unterschiedlichster Natur sein. Am gebräuchlichsten sind derzeit Informationssysteme wie Datenbanken oder das lokale Dateisystem eines Web Servers. Durch die zunehmende Zahl an Web-Anwendungen, die Altsysteme integrieren (EAI, Enterprise Application Integration) wird ein Einsatz von Diensten, wie sie bspw. durch Web Services bereitgestellt werden, zunehmend attraktiver. Neben einer funktionalen Integration existieren auch Anforderungen nach einer Informationsintegration auf Geschäftsebene (EII, Enterprise Information Integration) (Leymann und Roller, 2002). Kombiniert mit beobachtbaren Trends hin zur prozessorientierten

Anwendungsentwicklung (C. Mayerl, Tröscher und Abeck, 2006) werden Entwicklung und Betrieb von *dienstorientierten Web-Anwendungen* in den nächsten Jahren weiter an Bedeutung gewinnen.

Ein wesentlicher Vorteil, der bei der Gestaltung des Informationsraums durch Dienste entsteht, ist die Erfüllung der Anforderung »lose Kopplung« von Informationsquellen. Sie ermöglicht ein effizientes Anpassen, Austauschen und Wiederverwenden von Informationen, die mittels Diensten bereitgestellt werden. Aus kommerzieller Sicht bietet eine dienstorientierte Schnittstelle darüber hinaus weitere Möglichkeiten. So ist im Gegensatz zur *monolithischen* Vermarktung in Form von Anwendungsportalen eine gezielte Vermarktung *dedizierter* Informationen möglich. Beispiele hierfür finden sich im Vertrieb von Nachrichten durch Content Syndikation mittels standardisierter Austauschformate wie RSS oder ATOM (Nottingham und Sayre, 2005).

4.1.1 Dienstorientierter Zugriff auf den Informationsraum

Eine lose Kopplung stellt auch für die Realisierung einer Service-orientierten Architektur (SOA) (siehe Abschnitt 3.1.1) ein wichtiges und grundsätzliches Qualitätsmerkmal dar, das unter anderem durch technologische Faktoren wie selbstbeschreibende Nachrichten, Zustandslosigkeit und Plattformunabhängigkeit gewonnen wird (Orchard, 2003). Die Web Service Technologie (Booth et al., 2003) verfügt über diese Eigenschaften (Christian Mayerl, Vogel und Abeck, 2005), wodurch sie sich ideal als Schnittstellen-Technologie zur Dienstbeschreibung für den Einsatz in Service-orientierten Architekturen (Dostal, M., Melzer und Zengler, 2005) eignet. Des Weiteren verfügen Web Services durch die Entkopplung des Nachrichten- vom Transportprotokoll über gute Eigenschaften für einen flexiblen Einsatz in heterogenen Umgebungen. Wird SOAP (Box, Ehnebuske, Kakivaya, Layman et al., 2000) eingesetzt, bieten sich zusätzlich Mechanismen zur Erweiterung durch Kodierungsregeln und Nachrichten sowie erweiterbare Sicherheitsmechanismen (Nadalin, Kaler, Hallam-Baker und Monzillo, 2003) bis hin zur Föderationsfähigkeit (Bajaj, Della-Libera, Dixon, Dusche et al., 2003). Die implizierte Schichtung der unterschiedlichen Funktionalitäten bietet einen äußerst geeigneten Ausgangspunkt für eine systematische Herangehensweise zur Beschreibung des Informationsraums durch Web Services. Ferner lassen sich für unterschiedliche relevante Aspekte wie Sicherheit und Vertrauen (Meinecke, Nussbaumer und Gaedke, 2005) oder Föderation (Meinecke, Gaedke und Nussbaumer, 2005) gezielt methodische Verfahren und Modelle entwickeln, die dazu beitragen, den Informationsraum einer dienstorientierten Anwendung zu systematisieren. Das macht sie zu idealen Basisbausteinen, die auf höheren Schichten zu mehrwertigen Anwendungen im Sinne von *Definition 2.2* kombiniert werden können.

Gerade im Bereich der Informations-Integration ist es von entscheidender Bedeutung, dass eventuell existierende Dienste wiederverwendet oder existierende Funktionalität leicht integrierbar gemacht wird. Die Verwendung eines Service-orientierten Ansatzes offeriert viele Vorteile, die jedoch nur bei einer disziplinierten und sachgemäßen Umsetzung erreicht werden können. Die zentrale Gefahr liegt darin, dass Dienste entwickelt werden, die bereits vorhandene Funktionalität (z.B. Lokationsdienst, Prüfungsdienst etc.) erneut anbieten. Ein wesentlicher Ansatz zur Lösung dieses Problems stellt eine »Beaufsichtigung« der neu zu entwickelnden Dienste in Form eines Vorgehensmodells dar. Um den Informationsraum in seiner

Gesamtheit zu kontrollieren, bedarf es Konventionen, die es gestatten, den Informationsraum auch hinsichtlich seiner semantischen Bedeutung zu erfassen. Ontologien erlauben einerseits die Repräsentation von Wissen über Informationen und deren Beziehungen. Andererseits bilden sie eine Grundlage zum Austausch von Informationen durch die Möglichkeit zur gemeinsamen Konzeptualisierung. Diese explizite formale Spezifikation einer gemeinsamen Konzeptualisierung durch existierende standardisierende Ontologiebeschreibungen ermöglicht eine offene und effektive Verknüpfung unterschiedlicher Informationsräume.

4.1.2 Lösungsansatz

Im Folgenden wird ein methodisches Vorgehen entwickelt, das es gestattet den Informationsraum zu formalisieren und basierend auf Informationsraum-Diensten zu partitionieren. Aufgrund der Evolution von Informationen durch Änderung an den Daten selbst, die Art und Weise wie auf sie zugegriffen wird oder die Erweiterung ihrer Konzepte, sind der Informationsraum und seine Dienste auch Gegenstand der Evolution. Dementsprechend wird in Abschnitt 4.2.3 die Dimension der Evolution dieser Dienste aufgezeigt, um konsequenterweise eine Systematik zur Entwicklung und Evolution des Informationsraums zu entwerfen.

Im nächsten Abschnitt wird zunächst eine formale Grundlage für die Beschreibung des Informationsraums an Hand des Globalen Typ Systems gegeben. Sie basiert auf den fundamentalen Axiomen der durch Tim Berners-Lee aufgestellten Axiome des World Wide Web Informationsraums. Ein zentrales Merkmal des neuen Informationsraums ist die Einführung von kanonischen Dienstschnittstellen. Diese werden exemplarisch am Beispiel des Schnittstellentyps CRUDS erläutert.

Um eine methodische Systematik zu gewährleisten wird ein Vorgehensmodell entwickelt, das basierend auf der formalen Definition des GTS die Entwicklung und Evolution von kanonischen Diensten beschreibt. Hierbei wird Wert auf kurze Produktionszyklen gelegt, indem Automatisierungsmöglichkeiten aufgezeigt werden. Ferner werden die Anforderungen Wiederverwendung und Evolution, die ja wesentliche Merkmale für Anwendungen im WWW darstellen, explizit berücksichtigt.

4.2 Konzeption eines Globalen Typ Systems (GTS)

Nachfolgend werden grundlegende Definitionen und Vereinbarungen den Informationsraum und seine Dienste betreffend vorgestellt. Die Vereinigung dieser Definitionen wird an der Axiomatik des Informationsraums einer Web-Anwendung ausgerichtet und definiert in ihrer Gesamtheit die formalen Kriterien und Voraussetzungen für den Informationsraum einer dienstorientierten Web-Anwendung. Wo nötig werden die Axiome Lee’s erweitert, um der Dienstorientierung gerecht zu werden. Zusätzlich werden im Globalen Typ System Voraussetzungen an zu verwendende Schnittstellentypen, deren Identifikation und die grundlegende Spezifikation von Informationseinheiten gestellt.

4.2.1 Grundlegende Axiomatik des Informationsraums Web

Nach (Tim Berners-Lee, 1990) wird der Informationsraum einer Web-Anwendung als eine Menge von Informationsressourcen angesehen. Diese Ressourcen sind untereinander beliebig verknüpfbar und bilden so mit Hilfe einer entsprechenden Auszeichnungssprache einen *Hypertext* (Nelson, 1965). Dabei spielt die Identifikation dieser Ressourcen eine Schlüsselrolle. Das Konzept des URI (Uniform Ressource Identifier) (T. Berners-Lee, Fielding, Irvine und Masinter, 1998) als Ausgangspunkt für den Informationsraum des Web wird durch die folgenden, grundlegenden Axiome beschrieben (Tim Berners-Lee, 1996):

Axiom 1 - *Universal*: »*Any resource anywhere can be given a URI. Any resource of significance should be given a URI*«: Die Universalität bezieht sich auf die Anwendbarkeit auf eine Informationsressource. Jeder Ressource ist durch einen universalen Identifikator referenzierbar und im Falle einer erhöhten Signifikanz dieser Ressource sollte ihr ein Identifikator gegeben werden. Im Sinne einer mathematischen Definition stellt sich die Frage, welche Elemente des Universums in Betracht gezogen werden, also der Definitionsmenge hinzugefügt werden. Im weiteren Sinn stellt sich hier die Frage, welche *relevanten* Informationen zum Beispiel in Form von Geschäftsobjekte eines Prozesses zum Informationsraum[23] hinzugefügt werden sollen.

Axiom 2 - *Global*: »*It doesn't matter to whom or where you specify that URI, it will have the same meaning.«:* Die Gültigkeit eines universalen Identifikators innerhalb des Informationsraums muss eindeutig sein. Das bedeutet für alle funktionalen Abbildungen dieses URI, dass diese Abbildung wiederum eindeutig ist. Verschiedene funktionale Abbildung können beispielsweise durch unterschiedliche Agenten erbracht werden, die die Identifikatoren auflösen. Es ist folglich unabhängig für wen, wo und in welchem Kontext ein Identifikator definiert wird, er hat immer dieselbe Bedeutung.

Axiom 3 - *Verlässlich*: »*a URI will repeatably refer to the same thing«.*
Ein universaler Identifikator referenziert immer die gleiche Informationseinheit. Diese Informationseinheiten werden dadurch idempotent, indem sie erfordern, dass ein mehrmaliges Anfordern der Ressource denselben Effekt hat wie das einmalige. Das bedeutet, dass auch eine Trennung des funktionalen Verhaltens, das durch einen Identifikator referenziert wird sich nicht verändern darf.

Axiom 4 - *Föderierbar*: »*URI space does not have to be the only universal space«:*
Die Föderierbarkeit des Informationsraums beschreibt die Grenze des Informati-

[23] Der Informationsraum des Web besteht aus adressierbaren Informationsobjekten. Im Umkehrschluss bedeutet das, eine nicht adressierbare Informationseinheit ist nicht Bestandteil des Informationsraums: »[..] An information object is *on the web* if it has a URI.«

onsraums Web. Damit wird der Ausgangspunkt für die Föderierbarkeit, wie Content Syndikation zwischen unterschiedlichen *Universen* oder die Bildung von virtuellen Anwendungen über verteilte Informationsräume gelegt. Die Globalität des Informationsraums wird nicht eingeschränkt, sondern vielmehr die Existenz unterschiedlicher, evtl. föderierbarer Partitionen ermöglicht.

4.2.2 Definition eines Globalen Typ System

Das Globale Typ System (GTS) wird anhand der folgenden Definitionen spezifiziert. Es beschreibt die Informationsraum-Dienste als eine Menge von Dienstschnittstellen, die auf einzelne Segmente des Informationsraum angewendet, einen Dienstnehmer mit Informationen versorgen. Basierend auf den Axiomen zum Informationsraum des Web werden die grundlegenden Eigenschaften, die für einen dienstbasierten Zugriff auf den Informationsraum nötig sind, abgeleitet. Die für das GTS wesentlichen Basisinformationseinheiten *ContentObject* und *ServiceCard* bilden die Grundlage für Erweiterungen und Anpassungen.

Forderung 4-1: **Informationsraum**
Ein Informationsraum ist das mit Hilfe einer Ontologie definierte Modell eines Weltausschnitts und bildet die Gesamtheit der Daten und Informationen, die diesen Weltausschnitt anhand des Modells syntaktisch[24] und semantisch[25] beschreiben und umfassen. Ein Informationsraum stellt die nötige Infrastruktur bereit, die eine Zuordnung der in ihm enthaltenen Elemente (Informationen) erlaubt im Sinne einer universalen, globalen, verlässlichen und föderierbaren Art und Weise.

Forderung 4-2: **Informationsraum-Dienst (IR-Dienst)**
Ein Informationsraum-Dienst stellt eine wohldefinierte und kanonische Schnittstelle zur Verfügung, die es gestattet eine spezifische Menge von Anwendungsszenarien mit Informationen zu versorgen. IR-Dienste unterscheiden sich von anderen Bestandteilen einer dienstorientierten Web-Anwendung darin, dass sie in mehreren Anwendungsszenarien wieder verwendet werden können, beispielsweise zur Unterstützung der Navigation über große Mengen von Informationen, zur situationsbezogenen Darstellung von Daten oder zur anwendungsübergreifenden Kosten- und Lizenzkontrolle (Gaedke, Juling und Nussbaumer, 2004). Ein IR-Dienst ermöglicht den Zugriff auf einen Ausschnitt des Informationsraums einer Web-Anwendung durch seine wohldefinierte Dienstschnittstelle. Die Wohldefiniertheit einer Schnittstelle zeichnet sich durch einen universalen und eindeutigen Zugriff im Sinne des Entwurfsmusters »Fassade« (Gamma, Helm, Johnson und Vlissides, 1995) aus. Der kanonische Zugriff auf den Informationsraum zeichnet sich

[24] Syntaktisch genügen Daten einem zugrundegelegten Schema wie z.B. ein relationales Datenschema einer Datenbank, einer Document Type Definition (DTD) oder einem XML Schema (XSD).

[25] Semantik umfasst die Bedeutung eines Datum und kann beispielsweise in einer Web-Anwendung durch Auszeichnung mittels eines dedizierten Markup erfolgen. Semantisch ausgezeichnete Daten werden daher oft auch als »Content« bezeichnet.

durch ein möglichst hohes Maß an Abstraktion und Regelhaftigkeit[26] hinsichtlich seiner Schnittstelle aus; in diesem Zusammenhang spricht man auch von den »constrained interfaces« (Henkel und J, 2005). Die dabei entstehende *Facettierung*[27] des Schnittstellenraums bildet den Ausgangspunkt für die horizontale Evolution.

Forderung 4-3: **Informationsraum-Identifikator**
Ein Informationsraum-Identifikator (Information Space Identifier, ISID) bildet einen global eindeutigen Identifikator, der es ermöglicht Objekte (Informationen) im verteilten Informationsraum eindeutig zu identifizieren. Ein Informationsraum Identifikator ermöglicht darüber hinaus die eindeutige (Rück-) Zuordnung (bijektive Informationsabbildung) einer Information zur Informationsquelle, dem Informationsraum-Dienst. Ein ISID zeichnet sich durch die Fähigkeit aus, verschiedene Informationsräume zu überspannen (Axiom 4) und so Objekte auch in verteilten Informationsräumen *verlässlich* und *eindeutig* (Axiom 3) zu identifizieren.

Bijektive Informationsabbildung

Mit der Einführung des ISID kann eine *bijektive Informationsabbildung* erreicht werden. Die Grundlage dazu bildet die *injektive Informationsabbildung*, die eine eindeutige Abbildung durch einen Identifikationsschlüssel auf Informationen gestattet (konform zu Axiom 3). Erlaubt ein Identifikationsschlüssel darüber hinaus auch die eindeutige Rückabbildung auf die versorgende Informationsquelle, so wird eine bijektive Informationsabbildung erreicht. Ein Beispiel für die injektive Informationsraumabbildung stellen die aus dem Web bekannten URIs (Uniform Resource Identifier) (T. Berners-Lee, Fielding, Irvine und Masinter, 1998) dar. Sie erlauben eine eindeutige Abbildung von Suchbegriffen (Identifikationsraumschlüsseln) auf Informationen. Eine bijektive Informationsabbildung gestattet darüber hinaus auch den Rückschluss einer Informationseinheit – durch einen ISID gekennzeichnet – auf den entsprechenden Informationsversorger.

[26] Die Regelhaftigkeit bezeichnet die Fähigkeit ein Protokoll auf Basis der abstrakten Kommunikationsprimitive, die durch die wohldefinierte Schnittstelle zur Verfügung gestellt werden, zu erfüllen.

[27] Der Begriff der Facettierung lehnt sich hierbei an die Facettenbildung von XML Schemas (XSD) an. Facetten bilden dort definierte funktional zusammenhängende Teilmengen auf einem primitiven Datentypen. So kann z.B. das Format einer Internationalen Standardbuchnummer (ISBN) als Facette auf dem primitiven Datentyp *string* definiert werden.

Konzeptualisierung des Informationsraums

Basierend auf der eingangs gestellten Forderung des Informationsraums (siehe Forderung 4-1) wird eine Konzeptualisierung vorgestellt, die im Verlauf des Informationsraum-Entwurfs zur Beschreibung dedizierter Szenarien verwendet wird. Dabei spielen die unterschiedlichen Ausdrucksformen der Konzeptualisierung sowie der Grad der Formalisierung eine wesentliche Rolle. Die Modellbildung vollzieht sich hierbei in den drei Ebenen »Semantische Konzeptualisierung«, »Syntaktische Konzeptualisierung« und »Physische Ausprägung«.

Aufgrund dieser Trennung lassen sich die unterschiedlichen Konzepte einzeln erweitern und auf eine dedizierte Problemstellung anpassen. So können formal definierte Ontologien über die Zeit mit zusätzlichen Regeln ausgestattet werden, um das modellierte Wissen kontinuierlich zu mehren. Aufgrund der Spezifika der unterschiedlichen Konzeptualisierungen lassen sich auch dediziert Rollen für deren Bearbeitung bestimmen, um beispielsweise in einem Team ausgeprägte Modellierungsspezialisten zu unterstützen.

- **Semantische Konzeptualisierung** umfasst die Entwicklung, Evolution und Wiederverwendung von Ontologien, um *Sinnhaftigkeit* und *Formalismen* zu erreichen. Die Konzeptualisierung des Informationsraums wird mit Hilfe von Ontologiebeschreibungssprachen des Semantic Web erreicht (vgl. Abschnitt 3.1.5.1). Dadurch lassen sich bedeutungstragende Assoziationen zwischen einzelnen Informationseinheiten hinsichtlich der *Inferenz* besser ausdrücken, als dies durch funktional-logische Beschreibungsformen wie UML der Fall ist. Um eine Maximierung der Ausdrucksstärke zu erlangen und eine nahtlose Einbindung von Informationsraum-Diensten zu ermöglichen, wird die *sinnhafte* Ebene von der *syntaktischen* Ebene in der Konzeption getrennt. Dadurch lassen sich einzelne Informationskonzepte der Informationsraum-Dienste formal durch Ontologien beschreiben. Zusätzliche Bedeutungsinhalte einzelner Begriffe oder Relationen können durch regelhafte Zusammenhänge erfasst werden. Darüber hinaus erlauben Ontologien auch die Bildung spezieller semantischer Relationen wie beispielsweise die Inversbildung oder transitive Berechnung von Merkmalen. Dadurch kann »fehlendes« Wissen durch Inferenzprozesse logisch abgeleitet und dem Informationsraum hinzugefügt werden.
- **Syntaktische Konzeptualisierung**: Sie sieht die Entwicklung eines spezifischen UML Modells, das die Beziehungen und Abbildungen zwischen konkreten Entitäten beschreibt. Die syntaktische Konzeptualisierung bildet die Schnittstelle zu einem plattformabhängigen Typsystem. Im Falle der Web Services wird dies durch XML Schemas (XSD) (Peterson, Biron, Malhotra und Sperberg-McQueen, 2004) beschrieben. Dazu gehören die Schematisierung der ausgetauschten Nachrichten, sowie die Konkretisierung der Geschäftsobjekte, die ein Basisdienst verwendet. Ein Beispiel für eine Konkretisierung solcher Datenobjekte in Form von UML Diagrammen findet sich in Abschnitt 4.2.4. Daneben wird die Ausprägung der Schnittstelle – der abstrakte Endpunkt – beschrieben. Eine Abbildung von UML in eine für Web Services korrespondierende Schematisierung durch XSD kann mittels XMI und XSL(T) Transformationen einfach erreicht werden (siehe (Carlson, 2001)).
- **Physische Ausprägung** nimmt die Abbildung der UML Modelle auf korrespondierende XML Schemas und die Realisierung von kanonischen Dienstschnittstellen vor. Die konkrete Anbindung eines Informationsraum-Segments wird in der physischen Ebene vollzogen. Hierbei können unterschiedliche Informationsquellen, wie Datenbanken oder das lokale Dateisystem zum Einsatz kommen. Denkbar sind aber auch Altsyste-

me (*Legacy*), die über heterogene Schnittstellen verfügen und durch »Wrapper« angeschlossen werden können.

Basisinformationseinheiten des Globalen Typ Systems

Im Folgenden werden die zwei grundlegend benötigten Ressourcentypen definiert, die gebraucht werden, um eine dienstorientierte Web-Anwendungen beschreiben zu können: das ContentObject als bedeutungstragende Informationsressource und die ServiceCard als Dienstbeschreibung. Beide Basisinformationseinheiten werden abstrakt definiert und beschreiben somit lediglich die Art und Weise was in einem konkreten Globalen Typ System umgesetzt werden muss.

Forderung 4-4: **ContentObject**
Ein ContentObject stellt eine auf standardisierten Metadaten basierte Informationseinheit dar, die durch einen ISID eineindeutig adressierbar ist. Ein ContentObject stellt innerhalb des GTS den Ausgangspunkt für die fortwährende Ableitung von Anwendungsbezogenen Daten, wie beispielsweise Geschäftsobjekte, dar. Die konkrete Ausprägung der Metadaten eines ContentObjects obliegt der entsprechenden Anwendungssituation. Der DublinCore (Andresen, 2003) Metadatenstandard bildet dabei die Ausgangsbasis für allgemein gültige Merkmale. DublinCore[28] stellt gerade aus Sicht der Beschreibung webspezifischer Daten eine ideale Grundlage dar, da auch multimediale Datentyp-Beschreibungen unterstützt werden (DCMI, 2006). Ein Beispiel für eine dedizierte Anwendungsdomäne stellt bspw. der IEEE Standard Learning Objects Metadata (LOM) (Hodgins und Duval, 2005) dar.

Forderung 4-5: **ServiceCard**
Eine ServiceCard bezeichnet eine dedizierte Menge von Metadaten, die einen IR-Dienst über seine funktionale Darstellung hinaus auch semantisch beschreiben. Eine ServiceCard bildet eine Projektion auf die Phase der vertikalen Evolution, in der sich ein Dienst befindet. Im Gegensatz zu rein funktionalen Dienstbeschreibungen wie sie durch WSDL (Chinnici, Gudgin, Moreau und Weerawarana, 2003), WIDL (M.G. Wales, 1999) oder IDL (Coallier, 1999) bietet die ServiceCard aufbauend darauf auch Informationen über die zugrunde liegenden Geschäftsprozesse und die zu verarbeitenden Geschäftsobjekte zur Verfügung. Sie stellt eine Brückenfunktion zwischen der funktionalen Ausprägung und der semantischen Konzeptualisierung sowie den nicht-funktionalen Eigenschaften eines Informationsraum-Dienstes dar. Dazu zählen Eigenschaften etwas Eigenschaften wie Vertrauen, Qualitätsparameter oder Dienstnutzungsbedingungen genauso wie ontologische Zugehörigkeiten.

4.2.3 Informationsraum Evolution

[28] Der Name DublinCore rührt aus einer Initiative am Rande der World Wide Web Konferenz im Jahre 1994 in Chicago her. Dort wurde die DCMI gegründet und eine erste Konferenz in Dublin/Ohio organisiert. Die Teilnehmer einigten sich auf dieser Konferenz auf eine Kernmenge von 15 Attributen zur Beschreibung und Einordnung von Webressourcen. Diese Kernmenge trägt passenderweise den Namen *DublinCore*.

Der Informationsraum wird anhand der IR-Dienste segmentiert. Diese Segmente werden durch die zugrunde liegenden Informationssysteme, Anwendungen, Prozesse oder Datenquellen definiert. Ein Informationsraum-Segment zeichnet sich durch die Fähigkeit aus, Informationen zur Verfügung zu stellen. Das schließt Informationsraum-Dienste – vornehmlich aus entfernten Informationsräumen – explizit mit ein. Auf diese Weise kann bspw. durch Wrapping und Proxy Ansätze eine Föderation von Informationsräumen realisiert werden. Ein Beispiel für eine einfache Informationsraum Föderation auf Basis eines entfernten Informationsraum-Dienstes ist die Verwendung der Google Web Services[29] durch eine CRUDS-Schnittstelle oder die Anbindung von Hochschul-Informations-System (HIS)[30] Diensten im Projekt Karlsruhe Integriertes Informationsmanagement (KIM) (Juling, 2005). Zwei grundlegende Konzepte hinsichtlich einer systematischen Unterstützung der Evolution bilden die Betrachtung der horizontalen und vertikalen Evolution.

4.2.3.1 Horizontale Evolution

Die horizontale Evolution findet auf dem *Schnittstellenraum* $\Re = (\Im_0, \Im_1, \ldots, \Im_n)$ des Informationsraums statt. Die dabei entstehenden Partitionen bilden Schnittstellenverbünde, die eine definierte funktional zusammenhänge Menge an Methoden und Protokollen bilden. Eine Partition bezeichnet in ihrer Gesamtheit einen Schnittstellentyp $\Im_i$. Beispiele für eine Auswahl solcher Typen sind z.B. CRUDS (IBM, 2006), STAIVE (Gaedke, Meinecke und Heil, 2006) oder WSRP (Kropp, Leue und Thompson, 2004). Die horizontale Evolution setzt einen initialen Vorgang der Kreativität und fachlichen Kompetenz zur Klassenbildung voraus, durch den auch die Facettierung des Informationsraums vorangetrieben wird. Diese fachliche Kompetenz findet sich z.B. in Standardisierungsgremien wie OASIS, dem W3C, universitären Forschungseinrichtungen oder Firmen. Es ist durchaus denkbar, dass in unterschiedlichen Informationsräumen diverse Funktions- und Protokollbündel existieren und somit auch unterschiedliche Facettierungen. Um eine Föderation solcher Schnittstellenverbünde zu gewährleisten, bedarf es einer eindeutigen Beschreibung und Zuordnung eines Verbunds. Im Globalen Typ System (GTS) wird dieser Eindeutigkeit durch die *ServiceCard* Rechnung getragen. Sie dient als Beschreibung eines Dienstes mit entsprechenden Vermerken zu verwendeten Ausgabeschemas, Kontexten und weiteren Protokollspezifika.

Die horizontale Evolution bildet die Klassenbildung im Schnittstellenraum weiter aus. Ein Schnittstellentyp der im Zuge der horizontalen Evolution durch Klassenbildung entstanden ist, definiert dabei wiederum den Ausgangspunkt zu einer weiteren Form der Evolution – der vertikalen Evolution. Der entstandene Schnittstellentyp wird als der initiale Evolutionsschritt $\Im_0$ bezeichnet. Abbildung 4-1 illustriert beispielhaft einen Informationsraum dessen Schnittstellenraum aus den Elementen CRUDS, STAIVE und WSRP besteht.

4.2.3.2 Vertikale Evolution

Ein Informationsraum-Dienst passt sich den speziellen, vorherrschenden Begebenheiten innerhalb der Informationsebene an. Vertikale Evolution kann durch Erweiterung der Schnitt-

[29] http://code.google.com/

[30] http://www.his.de/

stelle entstehen. Dazu gehören weitere Outputschemas, Kontextdefinitionen oder weitere Methoden, die den Evolutionsschritt$_0$ stärker typisieren. Gerade Web Services sind dafür äußerst geeignet, da durch die Entkopplung von Typdefinitionen und Nachrichtenformaten neu hinzukommende Typdefinitionen leicht in die Dienstbeschreibung integriert werden können (Abschnitt siehe 3.1.3). Dadurch können Dienste variiert – in Richtung einer speziellen Anwendung ausgeprägt – werden, ohne dass sie die generischen Eigenschaften ihres zugrunde liegenden Schnittstellentyps verlieren. Wichtig hierbei ist, dass im Sinne der vertikalen Evolution nie Semantik und Funktion eines früheren Evolutionsschrittes geändert werden – sie wird lediglich erweitert. Die dadurch entstehende Versionierung vollzieht sich also durch Erweiterung der Schnittstelle, verursacht durch beispielsweise Anforderungsänderungen oder, im Falle einer Integration von Legacy Systemen, einer Änderung an diesen. Die entstehende Versionierung wird in der zugehörigen ServiceCard beschrieben. Diese kann dann beispielsweise mit Hilfe einer UDDI Registrierungsdatenbank (siehe Abschnitt 3.1.4) synchronisiert werden, indem für jede Version ein entsprechendes tModel vergeben wird (green page Suche) oder aber kumulierte tModels für den jeweiligen Evolutionsschritt. Ein Vorgehen zur Umsetzung mittels einer UDDI Registrierung findet sich in (Kyle Brown und Ellis, 2004).

Die vertikale Evolution findet iterativ statt und basiert zu Beginn auf einem initialen Evolutionsschritt $\Im_i$ eines dedizierten Schnittstellentyps. Danach vollzieht sie sich sukzessive, basierend auf den vorherrschenden Gegebenheiten von Anwendungsszenarien, Nutzungskontexten oder Geschäftsprozessen. Der initiale Schnittstellentyp bleibt durch die Iterationen hindurch vorhanden. Auf diese Weise begegnet die vertikale Evolution dem Problem der Versionierung und Rückwärtskompatibilität, dem Web Services unterliegen, durch eine systematische und zielführende Evolutionsmethodik. Abbildung 4-1 illustriert die vertikale Evolution in Form der trichterförmigen Erweiterung des initialen Schnittstellentyps.

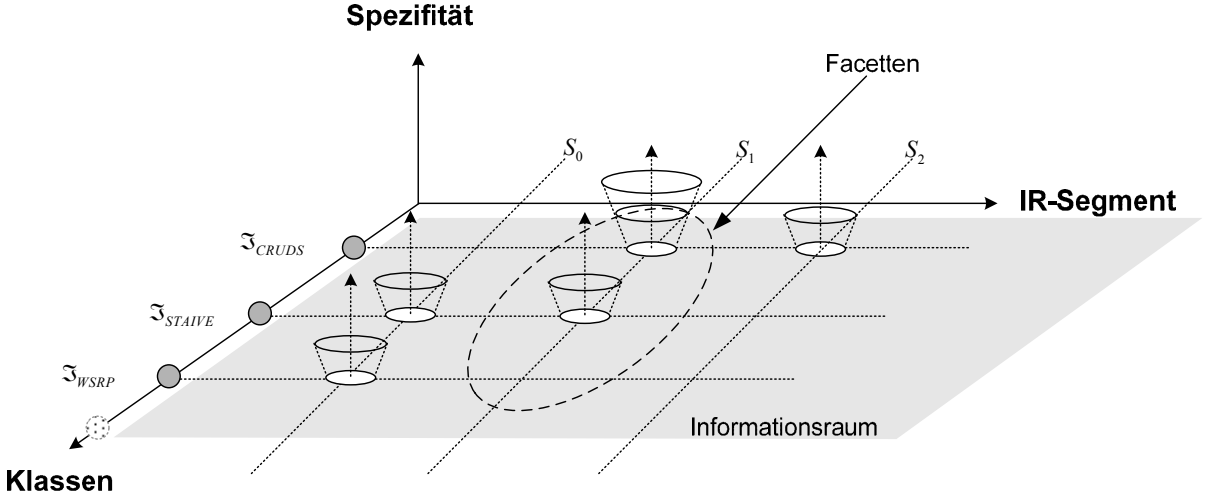


Abbildung 4-1: Evolutionskarte des Informationsraums. Vertikale Evolution beschreibt eine zunehmend anwendungsorientierte Spezifität, während die horizontale Evolution die Klassenbildung funktional zusammengehöriger Operationen bezeichnet. Der Schnittstellenraum umfasst die Klassen CRUDS, STAIVE und WSRP. $\Re = \left(\Im_{CRUDS}, \Im_{STAIVE}, \Im_{WSRP}\right)$

Schnittstellenanpassungen, die aufgrund der vertikalen Evolution entstehen ändern die Schnittstelle lediglich in erweiternder Form bei gleichzeitiger Konservierung ihrer bereits enthaltenen Eigenschaften.

Abbildung 4-2 stellt den Schnittstellentyp CRUDS auf dem Informationsraum-Segment S1 dar. Dabei wurden seit dem initialen Evolutionsschritt bereits zwei weitere Spezialisierungen vorgenommen. Ein projektbezogenes Beispiel für den Vollzug einer anwendungsspezifischen Evolution stellt beispielsweise die Schnittstellenerweiterung von CRUDS hinsichtlich der Anbindung der Hochschul-Informationssysteme im KIM-Projekt dar. Eine Systematik, welche die kontinuierliche Spezialisierung in Form der vertikalen Evolution unterstützt, wird im methodischen Vorgehen in Abschnitt 4.3.5 vorgestellt.

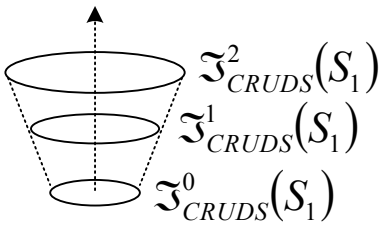


Abbildung 4-2: Vollzogene Vertikale Evolution auf dem IR Segment S1 und dem Schnittstellentyp CRUDS.

4.2.4 CRUDS – Ein Informationsraum-Dienst

Der kanonische Schnittstellentyp CRUDS (**C**reate, **R**ead, **U**pdate, **D**elete und **S**earch) stellt eine abstrakt gehaltene und dadurch universell einsetzbare Schnittstelle als Ausgangspunkt für einen Informationsraum-Dienst dar. In diesem Abschnitt wird vorgestellt wie dieser Schnittstellentyp mit Hilfe der Web Services Technologie umgesetzt werden kann. Dabei wird Bezug genommen auf die horizontale und vertikale Evolution, die Einbettung in das Globale Typ System und das Protokollverhalten der einzelnen Dienstprimitive. Ein Beispiel für den erfolgreichen Einsatz dieses Informationsraums-Dienstes bildet das Projekt Karlsruhe Integriertes Informationsmanagement (KIM).

Um die Konformität zum Globalen Typ System zu erreichen, müssen die grundlegenden Basiskonzepte anhand der in Abschnitt 4.2.2 aufgestellten Definitionen ISID und IR-Dienst umgesetzt werden. Zusätzlich bedarf es der Basiskonzeption der Informationseinheiten ContentObject und ServiceCard.

Das ISID Konzept

Ein ISID bildet nach Forderung 4-3 den Zusammenschluss eines Dienstidentifikators und eines Objektidentifikators. Dieser Zusammenschluss ist zunächst rein konzeptionell und kann durch unterschiedliche konkrete Implementierungsplattformen realisiert werden. Im vorliegenden Fall wird ein ISID auf Basis des URI Standards kodiert. Dadurch lassen sich unterschiedliche Konkretisierungen der ISID erreichen, wie beispielsweise in Form eines URN (Unique Ressource Namespace) oder der XRI (Extensible Resource Identifier) (siehe Abbildung 4-3). So können strukturierte ISIDs formuliert werden, die durch selbstbeschreibende Auszeichnungen (im Beispiel »isbn«) domänenübergreifend interpretierbar sind. Der das

Objekt identifizierende Teil kann, im Falle einer Dienstkaskade, wiederum einen ISID beinhalten.

(ServiceId, ObjektID)
urn:gts:isid:service-id:oject-id
xri://mwrg.tm.uni-karlsruhe.de/(urn:isbn:0-395-36341-1)

Abbildung 4-3: Eine Auswahl unterschiedlicher ISID Notationen als Tupel sowie mit den standardisierten Formaten URN und XRI.

Implementierung eines ISID am Beispiel von UDDI

Bei der Konkretisierung der ISID unter Verwendung einer UDDI Registrierung (siehe Abschnitt 3.1.4) handelt sich um einen Zeiger auf ein Objekt, der dieses eindeutig identifiziert und gleichzeitig lokalisierbar macht. Das wird erreicht, indem sowohl der *UDDI-bindingKey* des entsprechenden Dienstes, als auch der Identifikator des Objekts selbst enthalten ist. Abbildung 4-4 zeigt den dreistufigen ISID-Aufbau anhand des zuvor entwickelten Globalen Typ System im Zusammenspiel mit einem Informationsraum-Dienst.

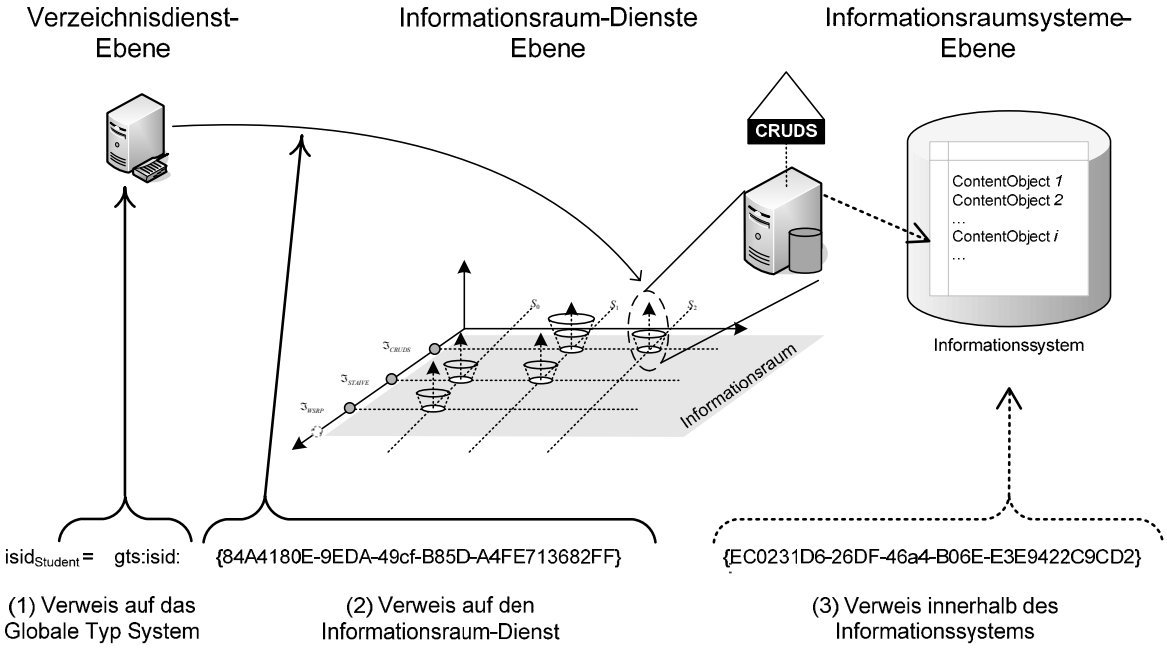


Abbildung 4-4: Verdeutlichung der Funktionsweise eines Informationsraum-Identifikators (ISID) in der und der GTS Architektur mit angeschlossenem Informationsraum. Das Implementierungsschema folgt hierbei dem URN Schema des WWW.

Die Verwendung eines ISID impliziert dabei automatisch einen dedizierten Ablauf unterschiedlicher Aktivitäten, die der Reihe nach durchgeführt werden müssen. Dazu gehört zunächst die Auswahl der entsprechenden Registrierungsdatenbank auf Verzeichnisebene. Im Falle von UDDI bildet diese Ebene eine organisationsweite Abdeckung aller möglichen zu registrierenden IR-Dienste. Föderationen können dabei sehr gut durch die Möglichkeit der Kombination unterschiedlicher Informationsräume realisiert werden. Die in Abbildung 4-4

dargestellte Auflösestrategie löst zunächst auf oberster Ebene (Verzeichnisebene) den passenden UDDI Server auf (1). Dieser gibt Auskunft über den entsprechenden CRUDS Informationsraum-Dienst, der einem speziellen Segment der zu Grunde liegenden Informationssystem Ebene zugeordnet wurde (2). Der IR-Dienst kann durch seine Suchfunktion wiederum die entsprechende Informationseinheit (ContentObject) liefern, die durch den ISID eindeutig spezifiziert wurde (3).

Im Rahmen des WSLS-Ansatzes wurde die CRUDS-Schnittstelle für Web Services spezifiziert, um den wohldefinierten Zugriff auf den Informationsraum zu gewährleisten. Das Interface beschreibt Dienstprimitive und Übergabekontexte zur Manipulation des Informationsraums basierend auf dem CRUD-Muster. Zusätzlich bietet sie ein Suchprimitiv an, das durch einen entsprechend konzipierten Übergabekontext die Suche im angeschlossenen Informationsraum unterstützt.

Jede der in CRUDS angebotenen Methoden besitzt zugehörige Kommunikationsprimitive, die als Verknüpfungen mit denen in der WSDL Spezifikation definierten Nachrichten realisiert werden. Dadurch können formale Beschreibungsverfahren wie die *Endlichen Automaten* verwendet werden, um zugehörige Protokolle zu spezifizieren, wie diese Primitive – in Form der ausgetauschten Nachrichten – miteinander kommunizieren (Tanenbaum, 2002). Durch die Verwendung der generischen Schnittstelle können spezialisierte Komponenten verwendet werden, die Aufgaben wie die Serialisierung und Validierung übernehmen oder gar noch zusätzliche Protokoll-Spezifika umsetzen können. Ein Beispiel für ein Protokollspezifikum bildet in Einführung der Internet Transaction ID (ITX). Sie ermöglicht die Kontrolle über langlaufende Transaktionen innerhalb der CRUDS Schnittstelle.

Durch die hohe Generizität bietet sich die CRUDS Schnittstelle ideal zur Automatisierung von Diensten an. So können mit Hilfe von Entwicklungsumgebungen vordefinierte Muster (engl. *templates*) erstellt werden, die Entwicklungszyklen rapide verkürzen helfen.

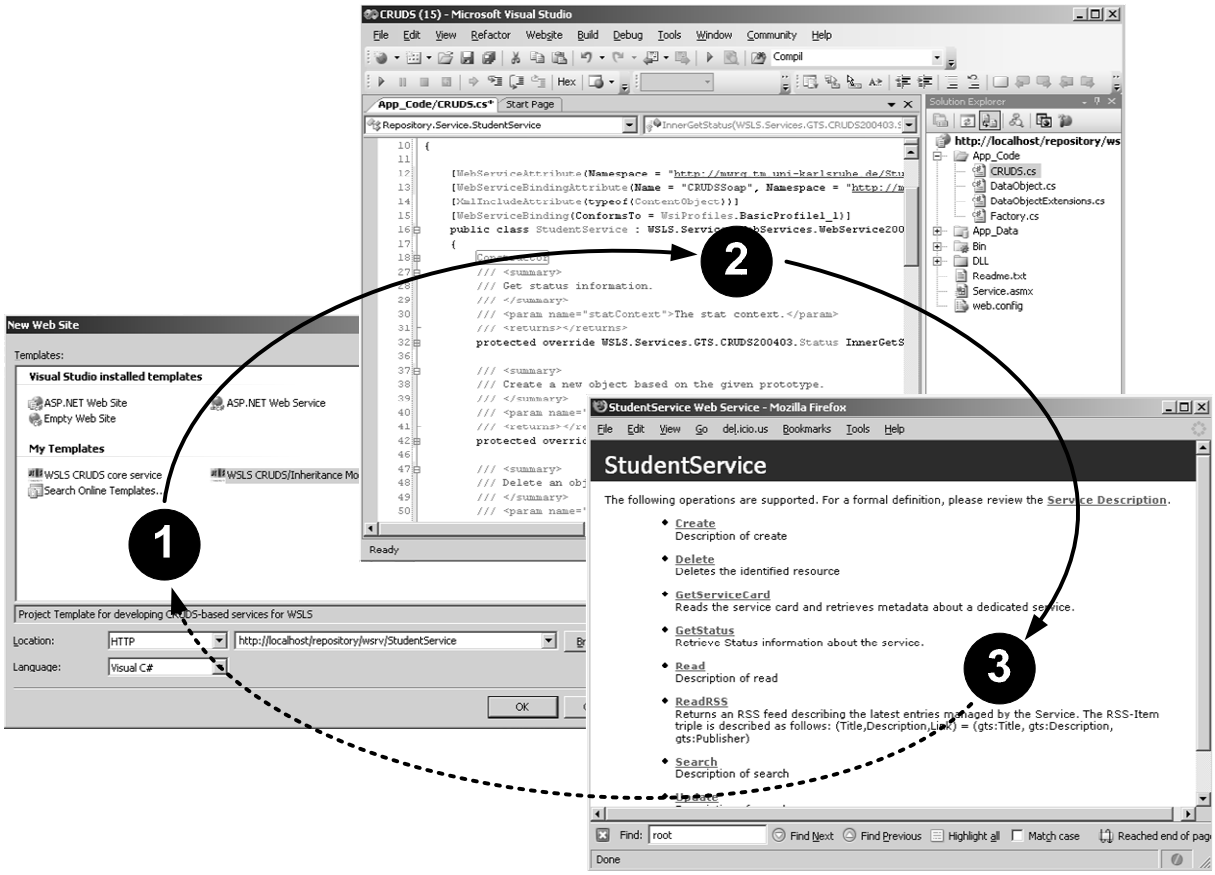


Abbildung 4-5:Rapid Service Development am Beispiel der CRUDS Schnittstelle und eines Templates für die Entwicklungsumgebung Microsoft Visual Studio.

Abbildung 4-5 stellt anhand von drei Schritten dar, wie ein auf Basis der CRUDS Schnittstelle erzeugtes Template eingesetzt werden kann, um schnelle Entwicklungszyklen für die Generierung von Web Services zu ermöglichen.

In Schritt (1) wird aus unterschiedlichen vordefinierten Mustern für Informationsraum-Dienste das passende ausgewählt. Im Zuge der horizontalen Evolution (siehe 4.2.3.1) ist zu erwarten, dass für die unterschiedlichen Segmente des Informationsraums passende kanonische Schnittstellen identifiziert werden. Diese können dann sukzessive der Menge der existierenden Schnittstellen als weitere wohldefinierte Muster hinzugefügt werden. Wichtig ist hierbei, dass zu diesem Zeitpunkt lediglich die Definition der Schnittstelle basierend auf den Nachrichtentypen des WSDL Dokuments erreicht wird. Die anwendungsspezifischen Datentypen wie Geschäftsobjekte oder konkreter Parameter werden explizit ausgegliedert. Dadurch lassen sich die in Abschnitt 3.1.3 aufgezeigten Probleme hinsichtlich der Interoperabilität lösen.

Schritt (2) befasst sich mit der anwendungsbezogenen Typdefinition. Auf Basis des Musters wird zunächst zur kanonischen Schnittstelle konformer Programmcode erzeugt. Dieser kann

entsprechend dem dedizierten Informationsraum Segment angepasst werden. Hierbei wird ein Schema-first Ansatz[31] verfolgt, der die Datenmodellierung in den Vordergrund rückt.

Eine Konkretisierung eines *ContentObject* wird im anschließenden Kapitel innerhalb der WSLS-Referenzarchitektur vorgenommen (vgl. Abschnitt 5.6). Weitere Beispiele für domänenspezifische Konkretisierungen finden sich im KIM-Projekt. Hier wurde die Basisdefinition des *ContentObjects* entsprechend den Vorgaben des Globalen Typ Systems um Anwendungsbezogene Charakteristika erweitert (siehe UML Diagramm Abbildung 4-6). Die Beschreibungen basieren auf den physischen Datenmodellen der durch die HIS bereitgestellten Daten zu Studierenden. Die Detailgrade werden durch entsprechend zugeschnittene *Ausgabeschemas* realisiert und können so passend den Bedürfnissen eines Geschäftsprozesses verwendet werden.

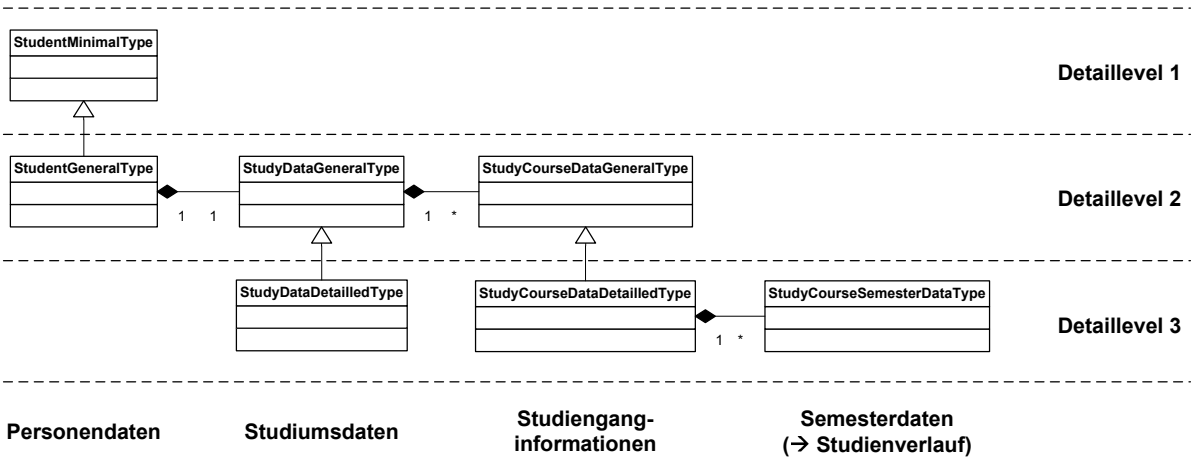


Abbildung 4-6: Eine mögliche Spezialisierung von Domänenspezifischen Daten zur Detaillierung eines Studendatensatzes wie ihr auf Basis der Datenbeschreibungen der HIS Systeme im KIM Projekt zum Einsatz kam.

Der nachfolgende Schemaausschnitt (vgl. Listing 4-1) demonstriert die technische Verknüpfung des vorgestellten ContentObject (Abbildung 5-12) mit den domänenspezifischen Erweiterungen (Abbildung 4-6) auf Basis des XML Schema Standards. Sie erfolgt analog zur UML Spezifikation durch Generalisierung der Basisinformationseinheit ContentObject (Zeile 5). Die im Schaubild dargestellten Detaillierungsgrade können in Form entsprechender komplexer Datentypen spezifiziert werden (Zeile 3, 13). Die Attributierung eines bestimmten Detailgrades kann dabei anhand der vorhandenen Schema Typen (Biron und Ashok, 2004) (Zeile 7-8) erreicht werden.

```
01.  <schema targetNamespace="http://kim.uni-karlsruhe.de/DataSchemes/StudentType.xsd">
02.  <import namespace="urn:wsls:gts:ContentObject" />
03.  <complexType name="StudentMinimalType">
04.    <complexContent mixed="false">
```

[31] Dieses Vorgehen wird zuweilen auch als WSDL-first oder Contract-first bezeichnet.

```
05.       <extension base="ContentObject">
06.       <sequence>
07.         <element name="matrNr" type="xs:int" />
08.         … weitere Spezifizierung des StudentMinimalType …
09.       </sequence>
10.       </extension>
11.       </complexContent>
12.     </complexType>
13.     … weitere Detailgrade …
14.   </schema>
```

Listing 4-1: ContentObject mit domänenspezifischen Erweiterungen

Ein Vorteil dieser Methode ist, dass dadurch bestimmte Interoperabilitätsprobleme schon durch den Entwurf beseitigt werden. Einerseits wird durch den Schema-First Ansatz die Gefahr von Fehlinterpretation durch Abbildung von plattformspezifischen Datentypen auf plattformunabhängige Schematypen verhindert. In (Prommer, 2006) wird der Schnittstellentyp CRUDS unter dem Aspekt der Interoperabilität betrachtet. Dabei wird ein Beispielszenario konstruiert, das die Interoperabilität von Web Services mit unterschiedlichen Plattformen evaluiert. Es werden die unterschiedlichen Möglichkeiten der Kodierungen miteinander kombiniert und deren Auswirkungen auf die Interoperabilität evaluiert. Im Zuge dessen lassen sich verschiedene Probleme identifizieren, die durch Verwendung diverser Plattformen mit der Web Service Technologie auftreten und durch den Einsatz kanonischer Informationsraum-Dienste überwunden werden können.

Übergabekontexte als Triebfeder für die Vertikale Evolution

Einen weiteren zentralen Aspekt der CRUDS-Schnittstelle bilden die Übergabekontexte. Sie werden als Parameter bei den Dienstprimitiven *Read*, *Update*, *Delete* und *Search* in Form eines XML-Dokuments übergeben. Zu jedem der korrespondierenden Kontexte - *ReadContext*, *UpdateContext*, *DeleteContext* und *SearchContext* – existiert je mindestens ein XML Schema. Ein CRUDS Informationsraum-Dienst kann dabei eines oder mehrere solcher Kontextschemas unterstützen. Welche Kontexte ein Dienst unterstützt, wird durch die *ServiceCard* ersichtlich.

Aus technischer Sicht wird ein hohes Maß an Abstraktion durch die Einführung kanonischer Schnittstellen unter Zuhilfenahme von Übergabekontexten erreicht. Sie ermöglichen eine ideale Anpassung der Schnittstelle im Zuge der vertikalen Evolution. Durch diese Kontexte wird das Konzept der *parametrischen Polymorphie* (Milner, 1978) realisiert, das dabei die engen Grenzen des WSDL Standards bezüglich der Polymorphie von Operationen überwindet (siehe dazu Abschnitt 3.1.3). Zusätzlich bildet es den Ausgangspunkt zur vertikalen Evolution durch die dediziert zunehmende Spezifität dieser Übergabekontexte in Richtung der konkreten Anwendungen. Das führt zu einer zunehmend funktional-fachlich ausgeprägten Verwendung und erlaubt so – analog zu einem polymorphen Methodenaufruf in der Objektorientierung – kontextsensitive Operationen.

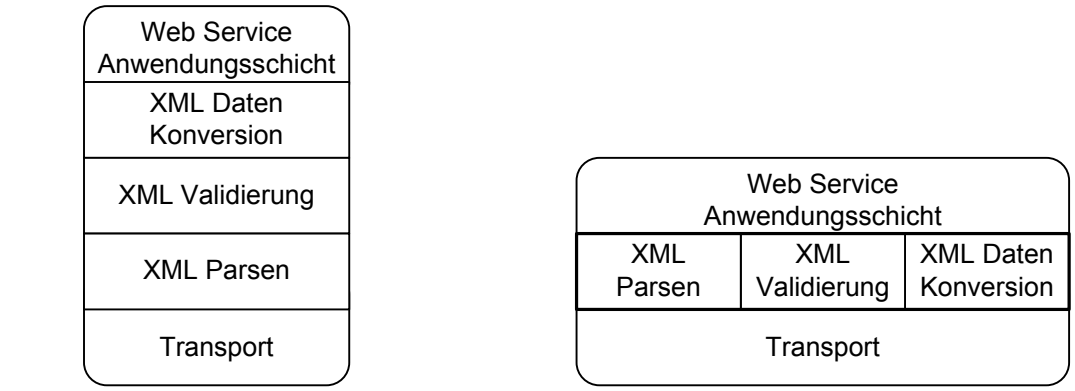
Ein Beispiel für einen Kontext bildet der *SearchContext* im CRUDS-Schnittstellentyp (Listing 4-2). Er definiert den Ausgangspunkt für die Spezifikation des Suchverhaltens der Search-Methode.

```
01.  <schema targetNamespace="http://www.wsls.net/2004/03/gts/cruds">
02.    <element name="SearchContext" type="tns:SearchContext" />
03.    <complexType name="SearchContext">
04.      <sequence>
05.        <element minOccurs="1" maxOccurs="1" name="QueryType" type="string"/>
06.        <element minOccurs="0" maxOccurs="1" name="Query" type="string"/>
07.        <element minOccurs="0" maxOccurs="1" name="Sort" type="string"/>
08.        <element minOccurs="0" maxOccurs="1" name="QueryId" type="string"/>
09.        <element minOccurs="0" maxOccurs="1" name="From" type="int"/>
10.        <element minOccurs="0" maxOccurs="1" name="To" type="int"/>
11.        <element minOccurs="1" maxOccurs="1" name="QueryMode" type="string"/>
12.        <element minOccurs="0" maxOccurs="1" name="OutputSchema" type="string"/>
13.      </sequence>
14.    </complexType>
15.  </schema>
```

Listing 4-2: Die Realisierung der parametrischen Polymorphie durch Übergabekontexte am Beispiel für das Dienstprimitiv Search.

Der Hauptvorteil der Schnittstellengestaltung mittels Kontexten liegt in deren Erweiterbarkeit durch das Hinzufügen neuer Kontextschemas. Die Schnittstelle kann zu einem späteren Zeitpunkt durch Hinzunahme eines neuen Kontextschemas erweitert werden, ohne dass Klienten, die diese Schnittstelle bereits verwenden, angepasst werden müssen. Dadurch können Dienste eines bestimmten Diensttyps flexibel auf vorherrschende geschäftsprozessabhängige Ausnahmen eingehen, *ohne* die gegebene Generalität der Schnittstelle zu verletzen.

Dieses *Ausfaktorieren* der Übergabekontexte als reine XML Elemente bringt aber auch den Nachteil mit sich, dass die in Entwicklungsumgebungen üblicherweise verfügbaren Werkzeuge zur automatischen Erzeugung von Web Service Proxies teilweise umgangen werden. Durch die Trennung der unterschiedlichen Vorgänge lassen sich allerdings bestimmte Phasen effizienter gestalten (vgl. Abbildung 4-7).



a) Herkömmliche Schichtung bei der Xml Verarbeitung

b) flacher und optimierter Stapel für die Xml Verarbeitung

Abbildung 4-7: Verhältnis zwischen optimierter und herkömmlicher XML Verarbeitung in Anlehnung an (Engelen, Zhang und Govindaraju, 2006).

So kann beispielsweise die Serialisierung der Kontexte in Form von XML Graphen durch Einführung von dedizierten Compilern signifikant erhöht werden. In (Engelen, Zhang und Govindaraju, 2006) wird ein Verfahren vorgestellt, das es gestattet die XML Verarbeitung quasi zu parallelisieren und die bisher getrennt (im Schalenprinzip) voneinander fungierenden Komponenten effizient in einer zu realisieren. Dies wird erreicht, indem ein XML Schema als kontextfreie Grammatik aufgefasst wird und dadurch eine effiziente Konstruktion von dedizierten Zerteilern ermöglicht. Dementsprechend müssen, um einen solchen *Compiler* entwerfen zu können, alle Datentyp-Informationen im Vorfeld bekannt sein. Das wird durch das Globale Typ System explizit unterstützt, da hier sämtliche Typinformationen, die zu einer Schnittstelle gehören, durch den Schema-first-Ansatz a priori bekannt sind.

4.3 Systematik zum Informationsraum Design

Durch eine systematische Planung und Integration des Informationsraums lassen sich grundlegende informationstragende Inhalte von Daten und Metadaten festlegen. Im Sinne des Globalen Typ Systems werden alle neu zu spezifizierenden Informationseinheiten auf einen gemeinsamen Nenner gebracht. Das kann z.B. mit Hilfe von speziellen Informationsraum-Diensten, den *Mediator Services*, geschehen. Dabei wird die Einflussgröße Daten entsprechend unterschiedlicher Ausprägungen und Verwendungsformen transformiert und an anfragende Informationsnehmer verteilt.

Eine effizientere Möglichkeit, die auf eine zusätzliche Indirektionsstufe mit Mediatoren verzichtet, besteht in der Vorhaltung entsprechender Metadaten und dazugehöriger Transformationen in den Informationsraum-Diensten selbst. Dadurch lässt sich die *vertikale Evolution* von Informationsraum-Diensten gezielt planen. Um die Wiederverwendung weiter zu erhöhen, wird die Dienstschnittstelle mit ihrer Syntax und Semantik von dem Verarbeitungsprozess getrennt. Auf diese Weise lassen sich beispielsweise Altsysteme effizient integrieren, da vorhandene Verarbeitungsprozesse der erbringenden Plattform unter den vorherrschenden Paradigmen verwendet werden können. Dies kann beispielsweise in Objekt-orientierter Entwurfsmethodik oder durch Aspekt-orientierte Ansätze erfolgen.

Bei der Anbindung des Informationsraums durch Informationsraum-Dienste werden vier unterschiedliche Anforderungsszenarien betrachtet:

- Erstellen neuer Informationsquellen und IR-Dienste / Hinzufügen von IR-Segmenten
- Integration von Altsystemen durch IR-Dienste / Anbindung von IR-Segementen in Altsystemen
- Wiederverwendung und Evolution von IR Segmenten
- Lebenszyklus von IR-Diensten (Verteilung, Registrierung, Entfernen)

Jedem dieser Anforderungsszenarien soll durch ein methodisches Vorgehensschema Rechnung getragen werden, das beschreibt, wie entsprechende Dienste für den Informationsraum bereitgestellt werden müssen. Ein Beispiel für einen Informationsraum-Dienst ist der im vorigen Abschnitt vorgestellte Schnittstellentyp CRUDS (siehe Abschnitt 4.2.4).

4.3.1 Erstellen neuer Informationsquellen

Die Anbindungsmethodik beschreibt das Hinzufügen neuer Informationsquellen durch Informationsraum-Dienste. Sie beschreibt wie auf die Informationen über eine wohldefinierte Web Service Schnittstelle zugegriffen werden kann. Diese Anforderung erlaubt die Nutzung der Daten sowohl durch Anwendungen, die um die Einbettung der einzelnen Dienste in einen umfassenderen Informationsraum – dem Globalen Typ System – wissen, als auch durch beliebige andere Anwendungen[32]. Dies dient gezielt der plattformübergreifenden Interoperabilität (spezifiziert durch WS-I) und der allgemeinen Wiederverwendbarkeit und somit der Unterstützung des Aufbaus komplexerer Anwendungen in Hinblick auf Web Service Orchestrierung[33] oder Choreographie[34].

Durch die Homogenisierung des Informationsraums durch die Informationsraum-Dienste lassen sich beispielsweise Werkzeuge wie BEA WebLogic[35] oder Microsoft BizTalk Server verwenden, um die Basisbausteine zu mehrwertigen Anwendungsdiensten zusammenzuführen. In Hinblick auf die Verwendung der Informationsraum-Dienste in unterschiedlichen Anwendungsszenarien kann die Schnittstelle spezifischer (vertikale Evolution) und damit stärker typisiert werden.

Abbildung 4-8 stellt die notwendigen Elemente dar, die bei der Erzeugung neuer Informationsquellen durch Informationsraum-Dienste benötigt werden. Die Entscheidung über die

---

[32] Im Umfeld des Web 2.0 spricht man in diesem Zusammenhang auch von den so genannten Mashups (engl. *to mash*, vermischen). Diese kombinieren bestehende Informationen, z.B. in Form von Diensten und schaffen dadurch neue Anwendungen.

[33] „Web Service Orchestration“ (engl.): die Beschreibung von Kontroll- und Datenfluss von Web Services auf Nachrichtenebene, auch organisationsübergreifend, aus der Sicht einer der beteiligten Organisationen und resultierend in einem Arbeitsablauf.

[34] „Web Service Choreography“ (engl.): die Beschreibung der Nachrichtenabfolge zwischen Web Services verschiedener Organisationen, allerdings nicht auf eine bestimmte Sichtweise beschränkt.

[35] http://www.bea.com

Granularität – also die Feinheit der Unterteilung – der Web Services ist vor allem wichtig unter dem Aspekt der Wiederverwendung: Hat eine einzige Datenbank und ein entsprechender Web Service den gesamten Informationsraum zum Inhalt, dann ist es schwer, nur Teile davon wiederzuverwenden. Auch Änderungen des Informationsraums kommt eine feinere Granularität zugute. Die Abbildung verdeutlicht den feingranularen Ansatz: Für jede Klasse und jede Beziehung des Informationsraums soll ein eigener Web Service zuständig sein, der über eine jeweils eigene Datenbank verfügt. Die einzelnen Web Services (im Bild als Dreieck dargestellt) können beliebig auf einen oder mehrere Rechner verteilt werden. Die eingezeichneten Verbindungen zwischen ihnen sind nicht physischer Natur, sondern rein konzeptuell: Sie spiegeln die in der zugrunde liegenden Ontologie vorhandenen Beziehungen wider.

Eigene Informationsraum-Dienste für Beziehungen zwischen verschiedenen Objekten sind zunächst nicht zwingend notwendig, denn die Beziehungen könnten auch durch direkte Verweise in den Objekten implementiert werden. Diese mögliche Vereinfachung bringt zunächst Performanzvorteile, verhindert jedoch die direkte Realisierung von Assoziationsklassen[36]. Das wäre eine unnötige Einschränkung, zumal solche Klassen sowohl in UML[37] als auch in OWL (siehe Abschnitt 3.1.5.1.4) darstellbar sind.

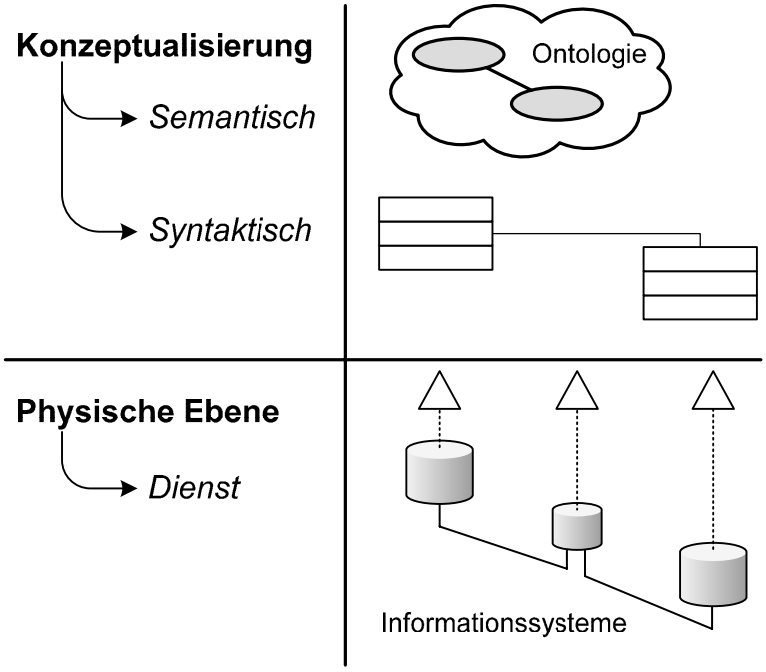


Abbildung 4-8: Hinzufügen neuer Basisdiensten durch Erweiterung des Informationsraums.

Eine derartige Aufsplitterung des Informationsraums auf viele einzelne Web Services wird erwartungsgemäß zu einer im Vergleich zu einem integrierten System verringerten Leistungsfähigkeit führen, wenn Anfragen mehr als eine Klasse des Informationsraums betreffen

[36] Assoziationsklassen in UML sind Beziehungen, die wie Klassen Attribute besitzen.

[37] UML – Unified Modelling Language, http://www.uml.org/

und somit die Verbindung zu mehreren Web Services und die Verknüpfung von Daten erfordern, die sich auf verschiedenen Rechnern befinden. Als Lohn für diesen Kompromiss[38] erhält man allerdings eine hohe Flexibilität und eine erhebliche Vereinfachung der Wiederverwendung.

Zusätzlich bietet auch die vorgeschlagene Lösung das Potenzial für weitere Optimierungen: Je nach Datenmenge und erwarteter Auslastung lassen sich Teile des Informationsraums gemeinsam auf einem einzigen Rechner unterbringen oder auf verschiedene Rechner mit angepasster Leistungsfähigkeit verteilen. Durch geeignete Techniken zur dynamischen Lokalisierung der einzelnen Dienste (siehe Abschnitt 4.3.4) lässt sich diese Verteilung auch im Betrieb ständig nach Bedarf verändern und anpassen[39].

4.3.2 Integration inkompatibler Altsysteme durch IR-Dienste

Nicht kompatible »Altsysteme« existieren in sehr großer Zahl und eine wichtige Anforderung besteht folglich darin, ihre Informationen und Funktionen in neu zu schaffende Anwendungen zu integrieren (Kreger, 2003). In traditionell heterogenen Systemlandschaften, wie sie beispielsweise in Universitäten angetroffen werden, stellt das Integrierte Informationsmanagement die Basis für eine effiziente Abwicklung der Geschäftsprozesse dar. Eine wichtige Anforderung stellt die Möglichkeit zur Integration von Informationsquellen (EII) oder Anwendungsteilen (EAI dar. Das schließt gerade solche Anwendungsszenarien mit ein, in denen heterogene Informationsquellen, die nicht mit der vorgestellten Lösung von Informationsraum-Diensten kompatibel sind, in den Informationsraum einbezogen werden können.

Hier ist vor allen Dingen die Berücksichtigung der eingangs aufgeführten Axiome des Informationsraums von Bedeutung. Analog zur Definition des Informationsraums (vergleiche Forderung 4-1 in Abschnitt 4.2.2 des GTS) muss hier die Universalität des Informationsraums überwunden und mit seiner Nichteindeutigkeit zusammengeführt werden. Dies ist, wie die Evolution auch, ein Sonderfall der Wiederverwendung, allerdings ist hierbei zusätzlicher Aufwand unumgänglich: Ein Wrapper muss als Web Service dieselbe Schnittstelle wie die bisher betrachteten Informationsraum-Dienste haben und darüber Zugriff auf die in der inkompatiblen Datenbank enthaltenen Daten bieten.

[38] In äußerst zeitkritischen Fällen oder bei sehr großen Datenmengen lassen sich jederzeit auf Geschwindigkeit und Skalierbarkeit optimierte Datenbanklösungen einsetzen.

[39] Oftmals wird Performanz nicht durchgängig benötigt, sondern nur zu bestimmten Spitzen, wie durch Werbungsmaßnahmen im Bereich e-Commerce oder auch bei erwartet hohem Verfügbarkeitsbedarf. Ein Beispiel dafür ist die Web-gestützte Einrichtung von Single-Sign-On Benutzerkonten im KIM Projekt.

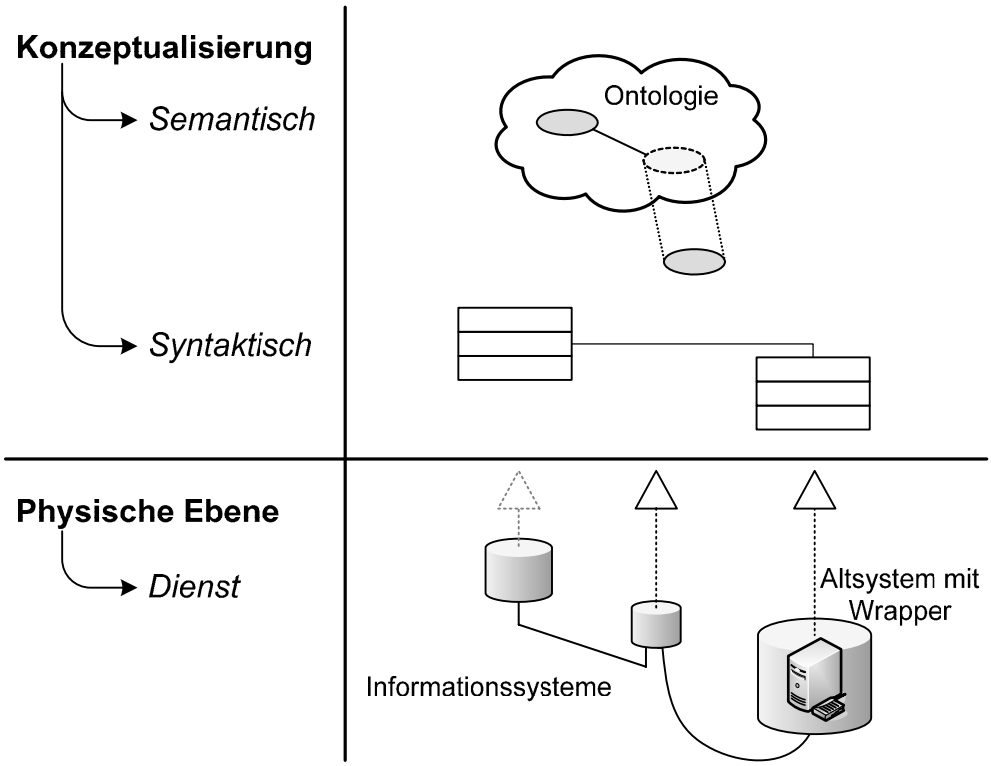


Abbildung 4-9: Integration von Altsystemen durch Wrapping Ansätze

Beispiele für die Integration von Altsystemen finden sich im Rahmen des Projekts KIM in Form einer Anbindung der HIS Systeme. In (Böttger, 2006) wird ein Verfahren entwickelt, das eine Anbindung von Workflowsystemen mit Hilfe der kanonischen Schnittstelle CRUDS vorstellt.

4.3.3 Evolution existierender Informationsraum-Dienste

Die Wiederverwendung bereits existierender Informationsräume soll ohne zusätzliche Softwarekomponenten, Mediatoren oder aufwändig durchzuführende Schema-Anpassungen ermöglicht werden. Informationsräume anderer Organisationen sollen aus technischer Sicht uneingeschränkt wie eigene wiederverwendet werden können, ohne dass sie – abgesehen von der zusätzlichen Nutzung selbst – jeglichen Einfluss erfahren. In (Meinecke, Nussbaumer und Gaedke, 2005) wird ein Verfahren entwickelt, das zusätzlich relevante Aspekte wie die Sicherheit und den Austausch unterschiedlicher Identitäten vorstellt. Die Bausteine zur sicherheitsbezogenen Föderation lassen sich aufgrund der Dienstorientierung des Informationsraums direkt übertragen.

Die Föderation von Informationsquellen unterschiedlicher Organisationen stellt in Hinblick auf die Realisierung einer Service-orientierten Architektur eine wichtige Anforderung dar (siehe Abschnitt 2.3.1). Das Erweitern eines zuvor erstellten Systems im Zuge eines Evolutionsschritts ist aufgrund sich ändernder Anforderungen häufig nötig. Dies wird soweit unterstützt, dass dazu analog zur Wiederverwendung keinerlei Aufwand zusätzlich zur Modellierung der erweiterten Daten notwendig ist.

Anstelle einer Schemaintegration sollen hierfür beim Entwurf eines neuen Informationsraums die wiederzuverwendenden Klassen und Beziehungen als solche *gekennzeichnet* und den entsprechenden, bereits existierenden, Informationsraum-Diensten alter Informationsräume zugeordnet werden. Dadurch lassen sich existierende Informationsräume weiterverwenden und weiterentwickeln bei gleichzeitiger Aktualisierung des föderierten Informationsraums.

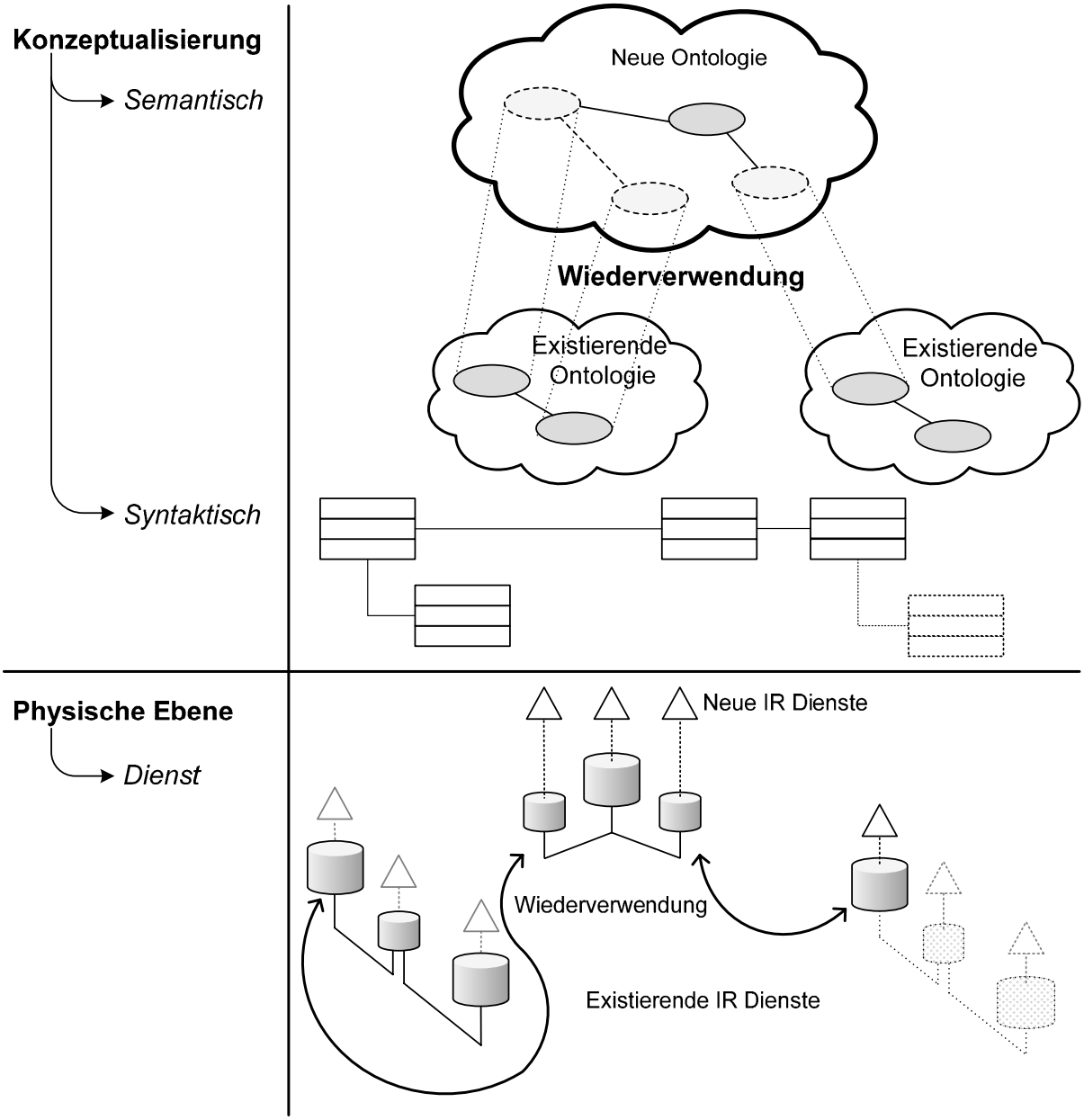


Abbildung 4-10: Evolution bereits existierender Informationsräume durch Wiederverwendung

Abbildung 4-10 zeigt, wie neue Ontologien wiederzuverwendende Konzepte referenzieren und die bereits bestehenden Informationsraum-Dienste mit den neu erstellten verbunden[40] werden. Nicht wiederverwendete Informationsraum-Dienste werden weiterhin im Rahmen ihres Informationsraums genutzt. Die Unterstützung der Evolution des Informationsraums vollzieht sich in Form einer Wiederverwendung des alten Informationsraums – oder großer Teile davon – innerhalb eines neuen.

4.3.4 Verteilung und Registrierung der IR-Dienste

Die Informationsraum-Dienste sind nicht an eine bestimmte Lokalität gebunden, sondern sind unabhängig voneinander (im Sinne der Anforderung an die lose Kopplung, siehe 2.3.1) und daher in hohem Maße verteilbar. Zur Unterstützung der gezielten Wiederverwendung ist die Auffindbarkeit der IR-Dienste wichtig. Die entsprechenden Zugangsinformationen, Lizenzierungsabkommen für die kommerzielle Nutzung oder entsprechende Dienstzugangsabkommen (engl. *Service Level Agreements*) werden in einer Registrierungsdatenbank verfügbar gemacht. In Abschnitt 3.1.4 wurden die grundlegenden Mechanismen und Technologien vorgestellt, die zur Registrierung und Verwaltung von Diensten verwendet werden können. Dies gilt in erster Linie für die Fälle der *neu erstellt*en und *integrierten* Informationsraum-Dienste. Zwei Facetten sind hier von besonderer Bedeutung: die Verteilung und die Registrierung.

Die Verteilung vollzieht sich durch das Aufsetzen neuer Dienste und die Integration von Altsystemen. Da in beiden Fällen zumeist zusätzliche Infrastruktur[41] von Nöten ist, finden sich die Informationsraum-Dienste oftmals bei dieser Infrastruktur.

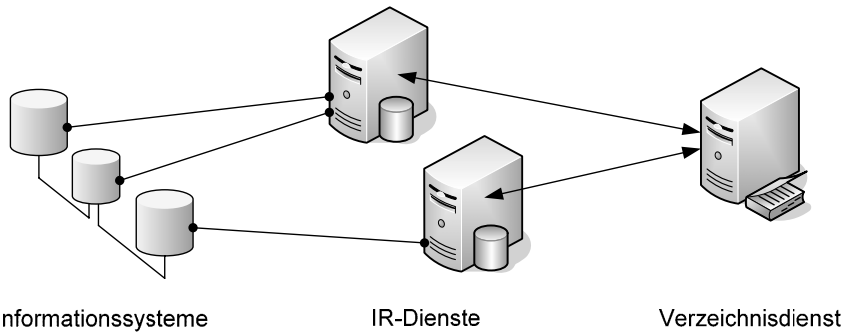


Abbildung 4-11: Registrierung der Informationsraum-Dienste bei einem Verzeichnisdienst

Die Registrierung soll wie in Abbildung 4-11 skizziert erfolgen: Jeder einzelne IR-Dienst soll im Zuge seiner Inbetriebnahme bei einem Verzeichnisdienst registriert werden. Dafür bietet sich die weltweite Registrierungsdatenbank UDDI an, da sie als global verfügbares Verzeich-

[40] Auch diese Verbindungen sind – wie alle zwischen Informationsraum-Diensten gezeichneten Verbindungen – nur konzeptuell.

[41] Zum Beispiel in Form von Datenbanken, Message-Queue Systemen oder den entsprechend zu integrierenden Altsystemen.

nis für das Registrieren von Web Services entworfen wurde und von einer großen Zahl an wichtigen Firmen wie IBM, Intel, Hewlett-Packard, Microsoft und Novell unterstützt wird. Eine mögliche Alternative bildet das von UNECE[42] und OASIS eingeführte ebXML. Dieser Standard, der einen breiteren Bereich als UDDI abdeckt, indem er teilweise auch die Aufgaben von WSDL und SOAP übernimmt, hat aber in der Industrie deutlich weniger Rückhalt als UDDI, was sich auch in der geringeren Unterstützung von Entwicklern – beispielsweise in Form verfügbarer Klassenbibliotheken – widerspiegelt (vgl. Abschnitt 3.1.4).
Für das Auffinden der zu einem Informationsraum gehörigen IR-Dienste sind ferner Verweise auf die entsprechenden Registrierungseintragungen notwendig. Diese sind zusammen mit der Informationsraum-Spezifikation verfügbar und werden in die definierende Ontologie eingebettet. Das Globale Typ System stellt in Form der *ServiceCard* einen Dienstzugangspunkt zur Verfügung, durch den semantische und syntaktische Informationen zusätzlich zu denen, die in UDDI verfügbar sind, über den Dienst geliefert werden können. Diese stellt den Ausgangspunkt für aufbauende Unterstützungsmodelle, beispielsweise in Form einer Dienstelandkarte dar (Gaedke, Meinecke und Nussbaumer, 2005).

4.3.5 Methodisches Vorgehen

Wie in Abschnitt 2.1 erläutert, wurde die Disziplin Web Engineering aus der Erkenntnis heraus begründet, dass auch für die Entwicklung von Anwendungen im Web ein systematisches Vorgehen notwendig ist. Dieser Erkenntnis folgend werden die Ergebnisse der vorangehenden Abschnitte in Form eines methodischen Vorgehens zum Informationsraum-Entwurf gesichert. Die Phasen decken dabei aus Sicht des Informationsraums und seiner Dienste den gesamten Prozess von der Modellierung bis zur Inbetriebnahme ab und berücksichtigen außerdem die Aspekte der Wiederverwendung und Evolution.

Im Unterschied zu Vorgehensmodellen der allgemeinen Softwareentwicklung wird für den Informationsraum-Entwurf der Fokus auf die Datenmodellierung gelegt. Das Vorgehensmodell stellt somit nur eine Phase eines größeren Prozesses dar. Aufgrund der entstehenden Dienstorientierung des Informationsraums und der Fokussierung auf standardisierte Web Services ist das Vorgehensmodell offen und bildet eine Bereicherung für die Integration in bestehende Ansätze.

4.3.5.1 Ermittlung der Phasen

Zunächst müssen die zeitlichen Abläufe erkannt und die verschiedenen Phasen der Nutzung identifiziert werden. Bereits eingangs erwähnt wurde die zu Beginn notwendige Datenmodellierung, deren Ergebnis als Eingabe für die Softwaregenerierung dient. Dieses Ergebnis liegt in Form eines OWL Dokuments vor, das überdies Hinweise auf wiederzuverwendende Klassen und Beziehungen, respektive den schon existierenden Informationsraum-Diensten, enthalten soll. Somit lässt sich die Phase der *Konzeptualisierung* in drei Schritte unterteilen:

[42] UNECE (United Nations Centre for Trade Facilitation and Electronic Business) ist ein Gremium der Vereinten Nationen zur Unterstützung des Handels und elektronisch abgewickelter Geschäfte (http://www.unece.org/cefact).

- Modellierung der Daten der Semantischen und Syntaktischen Ebene.
- Wiederverwendung, also Identifizieren von bereits existierenden Informationsraum-Diensten, die Teile des Informationsraums aufnehmen können.
- Erstellung der Ontologie in Form von OWL, die als Eingabe für Generierung dient.

Durch die Definition des Informationsraums mit kanonischen Schnittstellen lassen sich Softwarekomponenten effizient zur automatischen Erzeugung von IR-Diensten heranziehen. Die Phase der *physischen Ausprägung* verfestigt die durch horizontale Evolution entstehende Facettierung des Informationsraums in Form von dedizierten Softwaremodulen. Diese sind spezialisiert für jeweils eine kanonische Schnittstelle und übernehmen deren Abbildung auf eine Zielplattform. Die Phase umfasst außerdem die Bereitstellung geeigneter Abbildungsmodule basierend auf der vorherrschenden Facettierung des Informationsraums, als auch die Durchführung dieser Abbildung. Die Phase der physischen Ausprägung umfasst die Aktivität:

- Generierung der Informationsraum-Dienste aus der Ontologie.

Die sich anschließende Phase der *Evolution* umfasst Aktivitäten hinsichtlich der Nutzungsaspekte von Informationsraum-Diensten wie den Betrieb. Damit die generierten Informationsraum-Dienste verwendet werden können, müssen sie ferner verteilt und registriert werden. Um die kontinuierliche Spezialisierung in Richtung der Anwendung (vertikale Evolution) zu unterstützen, bedarf es zusätzlicher Schritte zur (iterativen) Anpassung und dem Testen der erzeugten Dienste. Daher berücksichtigt die Phase der Evolution die folgenden Aktivitäten:

- Verteilung und Registrierung der Informationsraum-Dienste.
- Anpassung der Informationsraum-Dienste.
- Testen der angepassten Informationsraum-Dienste.

Abbildung 4-12 stellt nun den zeitlichen Ablauf der zuvor ermittelten Phasen mit entsprechenden Übergängen des Vorgehensmodells dar. Die einzelnen Aktivitäten und die dabei involvierten Rollen werden im folgenden Abschnitt detailliert erläutert.

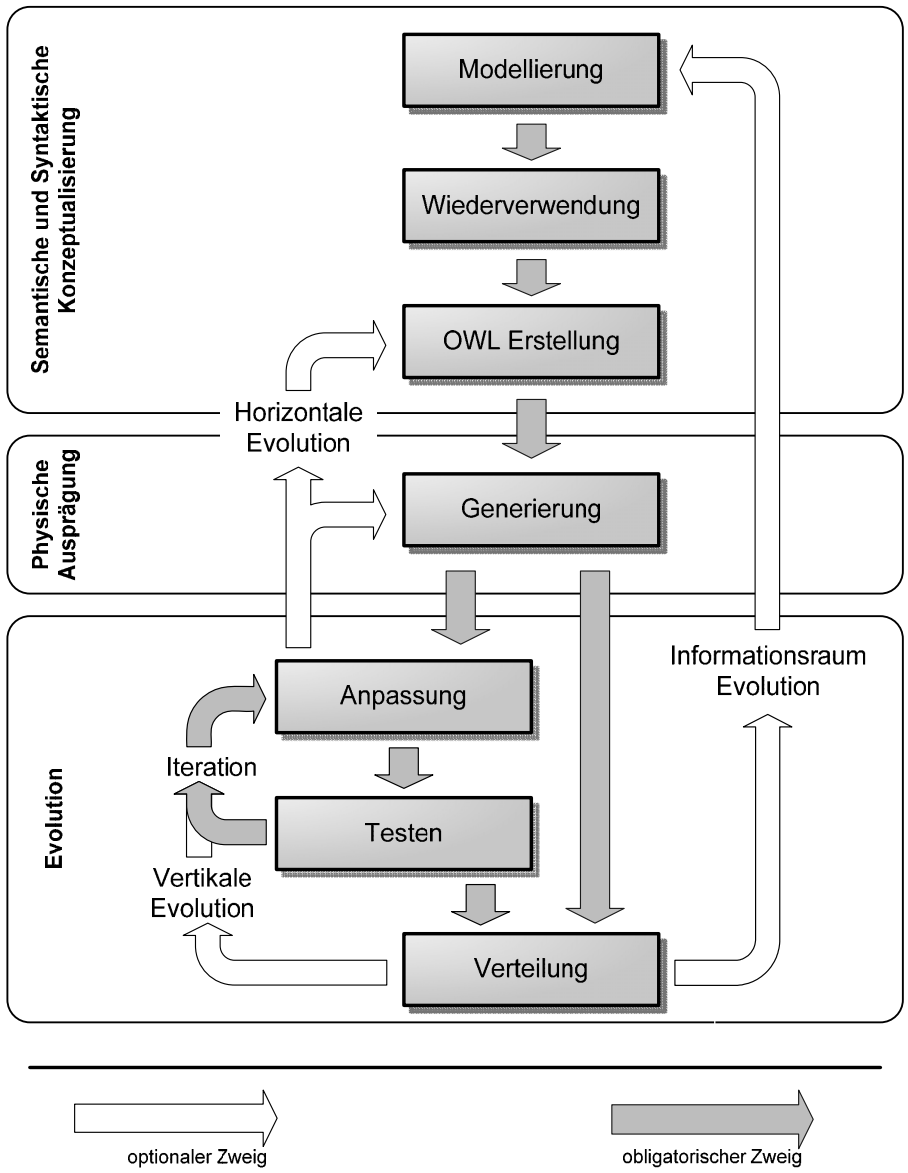


Abbildung 4-12: Methodisches Vorgehen, welches das Zusammenspiel der Phasen Konzeptualisierung, Physische Ausprägung und Evolution beim Informationsraum-Entwurf darstellt.

4.3.5.2 Beschreibung der Aktivitäten

Im Folgenden werden die Aktivitäten beschrieben, die in den einzelnen Phasen des im vorigen Abschnitt entwickelten Vorgehensmodells benötigt werden. Sofern nötig, werden jeweils entsprechende Rollen eingeführt, die dediziert diese Phasen durchführen sollen. Je nach Eignung, Zeitplanung und Größe des Projekts ist es durchaus sinnvoll, dass eine Person mehrere Rollen in Personalunion bekleidet. Umgekehrt kann bei sehr großen Projekten eine Rolle auch von einer Gruppe von Personen eingenommen werden.

Datenmodellierung

Diese Phase beschäftigt sich ausschließlich mit der Modellierung der Daten und dem Festhalten des Ergebnisses dieser Modellierung in Form eines Übergabedokumentes. Dieses Dokument ist gleichzeitig das von dieser Aktivität erwartete Resultat und Bedingung für den Übergang auf den nächsten Schritt. Die für den Informationsraum-Entwurf zentrale Rolle des

IR-Architekten bildet hierbei auch die Schnittstelle in einen umgebenden Gesamtprozess. So werden insbesondere die Ergebnisse der Anforderungsanalyse in Form der Software Requirements Specification[43] festgehalten.

Weder das explizite Vorgehen bei der Datenmodellierung noch das Format des Übergabedokumentes wird im Rahmen dieses Vorgehens fest vorgeschrieben. Im Bereich der Datenmodellierung gibt es eine Vielzahl umfassender Ansätze mit teilweise sehr guter Unterstützung durch Werkzeuge. Außerdem finden sich gerade in EII- oder EAI-Projekten vielfach schon vorhandene Datenmodelle, die nicht kompatibel mit Ontologien sind und auch keine automatische Transformation erlauben.

Im Falle von neu zu erstellenden Informationsraum-Diensten ist zu beachten, dass OWL Ontologien teilweise Konzepte der Objektorientierung umsetzen und deshalb ein Objektorientiertes Datendesign sinnvoll ist. Untersuchungen, die sich speziell auf das Erstellen und Pflegen von Ontologien konzentrieren finden sich beispielsweise in (Sure, Erdmann, Angele, Staab et al., 2002). Hier wird ein umfassender Ansatz beschrieben, der mit *OntoEdit* parallel dazu auch ein entsprechendes Werkzeug zur Verfügung stellt[44]. Eine weitere Möglichkeit besteht in der Datenmodellierung durch die UML. Die Object Management Group[45] (OMG) hat das »Ontology Definition Metamodel« als Standard verabschiedet, mit Hilfe dessen Ontologien durch UML Modellierungsanwendungen erzeugt werden können (IBM und Sandpiper Software Inc., 2006). In gewohnter Weise modellierte Daten lassen sich so als entsprechendes Ontologie-Dokument exportieren (Brockmans, Volz, Eberhart und Löffler, 2004).

Über die Datenmodellierung hinaus kann die Rolle *Informationsraum-Architekt* bei Bedarf in Form zusätzlicher Dokumente Hinweise für die Durchführung der nachfolgenden Phasen geben. Beispiele dafür sind Vorgaben, welche die benötigte Verfügbarkeit oder andere Qualitätsmerkmale der Informationsraum-Dienste, einzustellende Zugriffsrechte oder die Verteilungsaspekte wie die Installationsziele der entstehenden IR-Dienste betreffen. Dazu gehört auch die Entscheidung darüber in welcher Weise die Facettierung des Informationsraums im Zuge der horizontalen Evolution weiter vollzogen wird. Die zuvor erwähnten Optimierungen in Form einer Zusammenfassung mehrerer Klassen und Beziehungen in einen einzigen Informationsraum-Dienst (siehe hierzu Abschnitt 4.3.1) lassen sich vom Informationsraum-Architekten an diesem Punkt als Empfehlungen für den Generierungsvorgang formulieren. Dies kann gegebenenfalls zur Aufnahme von weiteren kanonischen Schnittstellen führen. Die Entscheidung dazu trifft er aufgrund der Semantik der Daten und des daraus resultierenden erwarteten Abfrageverhaltens.

Wiederverwendung

Die Rolle *Wiederverwendungs-Bibliothekar* hat in dieser Phase die Aufgabe, sowohl für die neu hinzugefügten Teile des Informationsraums wiederverwendbare IR-Dienste zu finden, als auch die Wiederverwendung der neu erzeugten IR-Dienste zu erleichtern.

---

[43] Das SRS (aktuell in Version Std 830-1998) stellt einen von der IEEE (Institute of Electrical and Electronic Engineers) veröffentlichten Standard zur Spezifikation von Software dar.

[44] http://www.ontoprise.com

[45] http://www.omg.org/ontology/

Für den organisationsinternen Fall geschieht Letzteres durch das Anlegen einer Sammlung von für die Wiederverwendung relevanten Informationen über die neu zu erstellenden Dienste. Neben einer genauen Beschreibung der Klassen und Beziehungen und insbesondere ihrer Daten-Attribute können das je nach Bedarf Meta-Informationen sein, die Aussagen oder Zusicherungen über die Art und Qualität der Daten betreffen. Beispiele dafür sind Zuverlässigkeit und Antwortzeit der jeweiligen IR-Dienste, Aktualität und Korrektheit der Daten sowie Pläne für die langfristige Aufrechterhaltung des Betriebs. Diese Informationen erhält der Wiederverwendungs-Bibliothekar vom Informationsraum-Architekten.

Art und Umfang dieser Informationen sind stark von den Bedürfnissen des Einzelfalls abhängig und werden deshalb – genau wie ihr Format – hier nicht weiter festgelegt. Als Beispiel wäre die Annotation der zugrunde liegenden Ontologie denkbar oder die Einführung eines organisationsweiten Informationsmodells wie im KIM-Projekt (Majer, Meinecke und Freudenstein, 2007). Diese Informationen sollten nach Bedarf auch nach der Inbetriebnahme des Systems weiter gepflegt und aktualisiert werden, um mit verlässlichen Informationen die Voraussetzung für eine erfolgreiche Wiederverwendung zu schaffen. Dienstzugangspunkte zur Verwaltung und Suche stellen Registrierungsdatenbanken wie UDDI (siehe Abschnitt 3.1.4) zur Verfügung. Hier lassen sich auch im Falle einer organisationsübergreifenden Wiederverwendung zusätzliche Merkmale auswerten, wie beispielsweise Dienstzugangsabkommen fremdgenutzter Dienste, aber auch Fragestellungen zum Digital Rights Management (DRM)[46]. In (Schwall, 2005) wird ein Verfahren vorgestellt, das dieser Problematik auf Basis von speziellen Informationsraum-Diensten Rechnung trägt.

Damit wird die Rolle des Wiederverwendungs-Experten phasenübergreifend und existiert auch nach Durchlaufen aller Aktivitäten des Vorgehensmodells weiter. Die Wiederverwendung wird vollzogen, indem der Wiederverwendungsexperte den Klassen- oder Beziehungsnamen mit dem Informationsraum-Identifikator (ISID) des Informationsraum-Dienstes assoziiert. Dadurch kann in der späteren Phase der *Generierung* der modellierte Dienst-Verbund aufgelöst und mit dem konkreten Dienst verknüpft werden. Durch die kanonischen Schnittstellen der IR-Dienste wird das durch die Wiederverwendung unentbehrliche Untersuchen potenziell wiederverwendbarer Komponenten und gegebenenfalls ihrer Anpassung zur Integration erleichtert.

Ontologieerstellung

Die Rolle *Ontologie Experte* hat in dieser Phase die Aufgabe, das Datenmodell in eine OWL-Ontologie (siehe Abschnitt 3.1.5.1.4) zu überführen und die Wiederverwendungs-Informationen aus der vorherigen Phase zu integrieren. Die Ontologie in Form eines OWL Dokuments stellt das Ergebnis dieser Phase dar. Wurde bereits in der Modellierungsphase durch den Informationsraum-Architekt eine in OWL formulierte Ontologie erstellt, so beschränkt sich die Erstellung auf die Einarbeitung der Wiederverwendungsinformationen. In den übrigen Fällen muss das Datenmodell nach OWL transformiert werden. Im Falle von in-

[46] Aufgrund der Unabhängigkeit eines physikalischen Mediums können digitale Daten leicht ohne Qualitätseinbußen weitergegeben werden. Da keine Nutzungsregeln zu solch digitalen Inhalten definiert werden können, die eine etwaige Weiterverwendung im Sinne eines Urhebers steuern, bleibt diese in aller Regel unberücksichtigt. Die Definition und Gewährleistung von Nutzungsregeln für digitale Inhalte wird in seiner Gesamtheit als Digitales Rechtemanagement (DRM) bezeichnet.

kompatiblen Datenmodellen muss dies manuell erfolgen, beispielsweise durch grafische Werkzeuge wie Protégé[47]. Wünschenswert sind allerdings automatische oder semi-automatische Verfahren. In (Gasevic, Djuric, Devedzic und Damjanovi, 2004) wird ein Verfahren entwickelt, das eine Transformation von Daten, die in UML modelliert wurden durch Anwendung von XSL(T)[48] in OWL umwandeln lässt.

Die Integration der Wiederverwendungs-Informationen erfolgt durch Annotation der entsprechenden Klassen und Beziehungen in der Ontologie. Auch eventuelle zusätzliche Annotationshinweise wie Optimierungen aus der Modellierungsphase werden in dieser Phase in die Ontologie integriert.

Generierung

Die Phase der Generierung beschäftigt sich mit Erzeugung von IR-Diensten. Die Aufgabe der Rolle *IR-Dienst Entwickler* ist es, die zuvor erstellte Ontologie in eine Menge ausführbarer Informationsraum-Dienste zu überführen. Die Verwendung des Globalen Typ Systems mit der Einführung kanonischer Schnittstellen wird eine automatische Erzeugung der Dienste erleichtert. In (Kunz, 2004) wird ein Informationsraum Compiler entworfen, der auf Basis einer Informationsraum-Ontologie den entsprechenden dienstorientierten Informationsraum erzeugt. Ergebnis dieser Phase sind einerseits die ausführbaren Informationsraum-Dienste, die in Form des Entwurfsmusters »Abstrakte Fabrik« (Gamma, Helm, Johnson und Vlissides, 1995) realisiert werden. Andererseits produzieren diese Fabriken auch die plattformspezifischen Quelltexte, die zur späteren Anpassung und Realisierung der vertikalen Evolution benötigt werden.

Anpassung und Test

Die Rolle *IR-Dienst-Entwickler* hat in dieser optionalen Phase die Aufgabe, eventuell benötigte Modifikationen an den IR-Diensten durchzuführen. Die Vorgaben hierzu werden vom Informationsraum-Architekten zur Verfügung gestellt und können im Zuge neuer oder veränderter Anforderungen entstanden sein. Die Modifikationen vollziehen sich im Sinne der vertikalen Evolution und stellen daher keine substanzielle Änderung der kanonischen Schnittstelle dar. Daher ist eine Änderung der Ontologie zunächst nicht nötig.

Sind substanzielle Änderungen derart vonnöten, dass die kanonische Schnittstelle geändert werden muss, führt dies zur Ausbildung eines eigenen IR-Diensttyps und einer Erweiterung des Schnittstellenraums des Globalen Typ Systems. Der sich vollziehende Vorgang der horizontalen Evolution führt dann zur Erweiterung der Generierungsphase um eine weitere Fabrik[49].

Wurde eine Anpassung durchgeführt, so wird eine Testphase notwendig, in welcher die Rolle *Tester* die vom *IR-Dienst Entwickler* zur Verfügung gestellten IR-Dienste auf Korrektheit der

[47] http://protege.stanford.edu/

[48] http://www.w3.org/TR/xslt

[49] In Analogie mit der biologischen Evolution findet hier eine *Variation* statt, die zur Ausbildung weiterer Facetten einer Spezies des substanziell zu ändernden IR-Dienstes führt. Die vertikale Evolution hingegen stellt eine Spezialisierung eines IR-Dienstes dar.

Modifikationen und Verträglichkeit mit der zugrunde liegenden kanonischen Schnittstelle testet. Aufgrund der kanonischen Schnittstellen lassen sich a priori Testfälle, beispielsweise in Form von Unit-Tests, für die generischen und bereits bekannten Teile der Schnittstelle wiederverwenden. Dadurch kann der Aufwand beim Testen auf die tatsächlich durchgeführten Änderungen beschränkt werden.

Verteilung

Die Rolle Informationsraum-Verwalter legt die erzeugten IR-Dienste in dieser Phase an ihrem jeweiligen Bestimmungsort auf den Zielrechnern ab und führt alle für die uneingeschränkte Nutzbarkeit des Systems notwendigen Aktionen durch. Dazu gehören unter anderem das Einrichten der IR-Dienste in ihren zugehörigen Hostsystemen wie einem Web-Server und eventuell zusätzlich benötigter Informationssysteme, die Verwaltung von zusätzlich benötigten Komponenten und Objekten wie Zertifikaten. Eine wichtige Aufgabe umfasst das Registrieren der Dienste in einem Verzeichnisdienst. Gerade hier bietet sich auch ein großes Potenzial für Automatisierungen an. So ließen sich beispielsweise im Fall einer UDDI Registrierungsdatenbank die kanonischen Schnittstellen mit entsprechenden semantischen Auszeichnungsmerkmalen aus der Datenmodellierung verbinden. Dies kann durch das Konzept der tModels in Form von *categoryBags* zu den Einträgen der IR-Dienste erreicht werden (siehe Abschnitt 3.1.4). Das Resultat dieser Phase sind funktionsbereite und identifizierbare IR-Dienste.

4.4 Zusammenfassung

Die Einführung des Globalen Typ Systems (GTS) erlaubt, basierend auf den grundlegenden Axiomen des World Wide Web, eine vollständige Beschreibung des Informationsraums einer Web-Anwendung. Die Konzeptualisierung des Informationsraums durch Ontologien bildet die Voraussetzung für ein formal definiertes System von Konzepten und Relationen (Gruber, 1993). Dadurch bildet der Informationsraum die Fähigkeit aus, sich wandeln und anpassen zu können. Diese Fähigkeit wird in Abschnitt 4.2.3 mit den Konzepten der horizontalen und vertikalen Evolution systematisiert.

Das Globale Typ System bildet die Grundlage für die Partitionierung des Informationsraums durch eine Menge von Diensten. Diese Informationsraum-Dienste basieren auf wohldefinierten und homogenisierten Schnittstellen, die den universalen Zugriff auf die zugrunde liegenden Informationen gestatten. Die Universalität der Schnittstelle ermöglicht die effiziente Wiederverwendung in einem dynamischen, schwach typisierten Umfeld. Durch den Vollzug der vertikalen Evolution hin zu einer sukzessiv anwendungsnäheren Spezifität kann ein statisches, stark typisiertes Umfeld unterstützt werden.

Das ISID Konzept erlaubt eine eindeutige Zuordnung von Informationen durch eine explizite Verknüpfung des erbringenden Dienstes und seiner Informationsressource. Auf diese Weise sind nicht nur Dienste auf Ebene einer eindeutigen Registrierung wie beispielsweise durch einen Verzeichnisdienst wie UDDI (Universal Description and Discovery and Integreation) (Bellwood, Clément, Ehnebuske, Hately et al., 2002) eindeutig identifizierbar. Die Informationseinheiten selbst, durch den ISID gekennzeichnet, erlauben einen eindeutigen Rückschluss auf den Erbringer der Information (bijektive Informationsabbildung). Dadurch lassen sich

mannigfache Ausprägungen der zugrunde liegenden Daten in Form unterschiedlicher Schemata als Informationseinheiten dediziert anfordern. Die Trennung schafft aber auch ideale Voraussetzung für die Skalierbarkeit von IR-Diensten. Durch Entkopplung der physischen Adresse des IR-Dienstes durch seine Dienstidentifikation lassen sich auch fortgeschrittene Beschreibungsverfahren zur Dokumentation und dem Management ganzer Landschaften dienstorientierter Web-Anwendungen realisieren (Meinecke, Gaedke und Nussbaumer, 2005). So kann beispielsweise der dynamische Austausch von Diensten durch die zusätzliche Indirektion der Dienstidentifikation erleichtert werden, da nur die zentralen Teile einer Registrierungsdatenbank aktualisiert werden müssen. Fortgeschrittene Konzepte wie die »Selbstheilung von Diensten« können auf Basis einer geeigneten Zuordnung von Dienstidentifikatoren erreicht werden. Auch eine systematische Zusammenführung unterschiedlicher Informationsräume hin zur Ausbildung von *Informationsraum-Föderationen* lassen sich durch die Entkopplung leichter arrangieren, da a priori von einem verteilten Adressraum ausgegangen wird.

Die Informationseinheit *ContentObject* bildet eine grundlegende Basis für die Verwendung von Metadaten nach dem DublinCore Standard. Sie bietet den Vorteil, dass das Maß an zusätzlichem Aufwand für die Metadaten gering ist. Außerdem ergibt sich dadurch der Vorteil eines definierten kleinsten gemeinsamen Nenners, den jede Informationseinheit besitzt. Dadurch können Web-Anwendungen in die Lage versetzt werden einen Teil der Geschäftsobjekte zu verstehen, ohne diese im Vorfeld zu kennen. So können Suchanfragen oder Verknüpfungen von Informationen auf Grund des gemeinsamen Verständnisses dieser Metadaten formuliert werden. Das ContentObject bildet die Grundlage für Erweiterungen und Anpassungen in unterschiedlichen Anwendungsdomänen. In (Razumna, 2003) wird eine auf dem ContentObject basierende Erweiterung um den LOM Metadaten Standard für den Bereich *eLearning* vorgestellt. Eine konsequente Generalisierung des ContentObject führt zu einer fortwährenden Segmentierung des Informationsraums in die unterschiedlichen Geschäftsobjekte einer Anwendungsdomäne. Dabei folgt die Generalisierung den Prinzipien der Objektorientierung, also einer datenorientierten Sichtweise auf den Informationsraum und nicht einer funktional-fachlich geprägten.

Unterstützung von Ontologien

Der Informationsraum-Entwurf stellt einen neuartigen Ansatz dar, der aufzeigt wie Informationssysteme unter Nutzung von IR-Diensten, im vorliegenden Fall durch Web Services, in einer Weise realisiert werden können, welche die Wiederverwendung und das Zusammenfassen von Datenbeständen vereinfacht. Zusätzlich bringt er noch eine semantische Komponente mit ein, die üblicherweise fehlt und dadurch – gerade in Informationsgetriebenen Projekten – auch die Nützlichkeit der Gesamtlösung beeinträchtigt.

> *»Although it could be said that every database designer has an ontology in his or her mind, most of it does not make it to the database model.« (Rojas, Bernardi, Ratsch, Kania et al., 2002)*

Diesem Problem wird durch die Rolle des Informationsraum-Architekten begegnet, indem eine Ontologie als Grundlage für die Definition des Informationsraums und der darauffolgenden Generierung der zugehörigen IR-Dienste herangezogen wird. Die Programmierung von Web Services oder nur die Einrichtung konkreter Datenschemas in der Datenbank wird damit aus Sicht der Semantik verschattet; überhaupt lassen sich durch den hohen Grad an

Automatisierbarkeit Produktionszyklen verkürzen und damit die Anpassbarkeit und Evolution deutlich besser unterstützen, als dies sonst der Fall wäre.

Unterstützung der Evolution

Die Evolution vollzieht sich in mehreren Iterationen und spiegelt die Veränderung des zu Grunde liegenden Informationsraums wider. Sie tritt nicht als eigene Phase auf, sondern als Iterationsschritt, der zum wiederholten Durchlaufen eines Teils oder aller Phasen führt. Damit wird explizit der Tatsache Rechnung getragen, dass Softwarelösungen immer wieder nachträglichen Änderungen der Anforderungen unterworfen sind und von in Betrieb befindlichen Produkten oftmals eine Erweiterung ihrer Funktionalität verlangt wird. Die in Abschnitt 4.3.5 vorgestellte Methodik unterstützt explizit die Informationsraum Evolution. So wird die Facettierung des Informationsraums durch die Ausbildung weiterer kanonischer Schnittstellen in Form der horizontalen Evolution vollzogen. Die dedizierte Spezialisierung von IR-Diensten, die eine Konkretisierung in Richtung spezialisierter Anwendungen unterstützt, wird durch die vertikale Evolution beschrieben.

5 Das WSLS-Rahmenwerk

In diesem Kapitel wird das WSLS-Rahmenwerk entwickelt, das den Ansprüchen an die zuvor aufgestellten Anforderungen entspricht. Der WSLS-Ansatz (WebComposition Service Linking System) stellt den funktionalen und integrativen Rahmen für die im vorigen Kapitel entwickelten Informationsraum-Dienste bereit.

5.1 Ausgangspunkt

Das Rahmenwerk basiert auf dem WebComposition Ansatz (Gaedke, 1998), der seit 1997 am Institut für Telematik der Universität Karlsruhe (TH) kontinuierlich fortentwickelt wird. Dabei wird eine Web-Anwendung in anwendungsspezifische Domänen unterteilt, die wiederum als adaptive Konfigurationseinheiten den Ausgangspunkt zur dienstorientierten Evolution darstellen. Eine Domäne wird dabei nach dem Baukastenprinzip durch Hinzufügen, Konfigurieren und Austauschen existierender Dienstkomponenten den der Domäne zugrunde liegenden Anforderungen angepasst. Dadurch wird eine Domäne befähigt, sich entlang den sich ändernden Anforderungen zu spezialisieren, und kann so optimal in ihre Umgebung integriert werden.

Das im Rahmen der vorliegenden Arbeit entstandene WebComposition Service Linking System (WSLS) Rahmenwerk (Gaedke, Nussbaumer und Meinecke, 2005) definiert eine Referenzarchitektur, die aufzeigt, wie der Informationsraum einer Web-Anwendung durch Dienste systematisch und effektiv nutzbar gemacht werden kann. Dazu wurde im vorigen Kapitel eine Methode entwickelt, die aufzeigt wie der Entwurf des Informationsraums auf Basis von Informationsraum-Diensten methodisch vorgenommen wird. Die dadurch entstandenen Schnittstellennormen in Form kanonischer Schnittstellen bilden eine wichtige Voraussetzung für die Realisierung eines offenen und erweiterbaren Systems, wie es durch die WSLS-Referenzarchitektur beschrieben wird.

Ein wesentlicher Fokus liegt auf der Aspekt-orientierten Sichtweise von Komponenten wie sie in WSLS zum Einsatz kommen. Die Trennung in die singulären Bereiche Inhalt, Navigation, Präsentation, Dialog, Prozess und Kommunikation grenzen dabei elementare, in Web-Anwendungen inhärent gegebene Aspekte von einander ab.

Das folgende Kapitel definiert die WSLS-Referenzarchitektur, die eine Web-Anwendung als eine sich über die Zeit ändernde Instanz dieser Architektur betrachtet. Die Architektur ist

offen gehalten und gestattet so, dass sie sukzessive um Modelle und Methoden angereichert wird, die eine systematische Konstruktion und Evolution von dienstorientierten Web-Anwendungen gewährleisten. Um eine effiziente Konfiguration der Aspekte, sowie deren nahtlose Komposition bzw. Verwebung während des Betriebs, zu ermöglichen, kommen unterschiedliche Strategien wie adaptive Algorithmen, Quantifizierungsmechanismen sowie domänenspezifische Konfigurationssprachen zum Einsatz, die eine nachhaltige Anpassung und Evolution gewährleisten.

5.1.1 WSLS Suite – technologische Realisierung methodischer Konzepte

Die WSLS Suite bildet eine im Zuge der vorliegenden Arbeit entwickelte Laufzeitumgebung, die zum Ziel hat die Modelle, Methoden und Rahmen-Architektur in produktiver Form als einsetzbare Werkzeuge umzusetzen. Dadurch lassen sich die Ergebnisse kontinuierlich überprüfen, da sie in unterschiedlichen Szenarien eingesetzt und evaluiert werden können. Die Entwicklung des WebComposition Service Linking System (WSLS) (Gaedke, Nussbaumer und Meinecke, 2005) wurde in verschiedenen Projekten mit unterschiedlichen Ausrichtungen und Schwerpunkten vorangetrieben und bildet das technologische und methodische Rückgrat für Weiterentwicklungen im Rahmen der dienstorientierten Entwicklung und Evolution von Web-Anwendungen.

Dazu gehören beispielsweise international ausgerichtete Forschungs-Kooperationen mit der Universität Madrid sowie der Aufbau des Forschungs-Portals webengineering.org. Außerdem wurden und werden zahlreiche Forschungs- und Entwicklungsprojekte mit WSLS umgesetzt und evaluiert. Dazu gehören unter anderem:

- Das Ziel des Projekts »Notebook-Universität Karlsruhe (TH)« (NUKATH) umfasste die Entwicklung und Evolution von anwendungsspezifischen Basisdiensten, die zur Unterstützung bei der Konstruktion ubiquitärer E-Learning Anwendungen dienen. Im Zentrum stand hierbei die orts- und zeitunabhängige Nutzung dieser Dienste, die durch unterschiedliche Einrichtungen sowie von Studierenden und Mitarbeitern der Universität zur Verfügung gestellt (Entwicklung) und im Zuge sich wechselnder Anforderungen dediziert angepasst werden können (Evolution) (2002-2003).
- Das von Microsoft Research, Cambridge, UK geförderte Projekt »Living, Learning, and Teaching in Mobile Universities« (M-University) setzte konsequent die erfolgreiche Entwicklung des Konzeptes der Anwendungsspezifischen Basisdienste fort, die im Rahmen des vom BMBF geförderten Projektes Notebook-Universität entstanden sind. Hierbei lag der Fokus auf der Evolution von Basisdiensten in einer föderalgeprägten interorganisatorischen Wechselwirkung. Ein wichtiges Ziel stellte dabei die Fähigkeit einer M-University dar, diese Wechselwirkungen durch dedizierte Dienste und Protokolle in Form eines Globalen Typ Systems ganzheitlich zu verbinden (2003-2004).
- Im Rahmen des universitätsweiten KIM-Projektes »Karlsruher Integriertes InformationsManagement« wird den Anforderungen an ein modernes geschäftsprozessorientiertes Informationsmanagement begegnet. Dabei stehen dienstorientierte Web-Anwendungen, die einen massiven Einsatz von Web Services basierend auf den in

Kapitel 4 entwickelten kanonischen Schnittstellen fokussieren im Zentrum (seit 2005).

- Das durch Microsoft Research, Cambridge, UK geförderte Projekt »Software Engineering for Information Appliances at Home« befasste sich mit der Anwendung von Konzepten aus dem Web Engineering zum Bau von Web-Anwendungen für aus Familien und Freunden bestehende Netzwerke in Privathaushalten. Aufbauend auf dem Web-Composition Service Linking System (WSLS) entstanden eine Reihe von Web-Anwendungen zur Visualisierung und Interaktion mit heterogenen Datenquellen im Web. Ein wesentliches Ergebnis stellte dabei die Gewährleistung der Adaptivität für unterschiedliche Benutzungsschnittstellen, wie beispielsweise dem Windows Media Center dar (2006).

5.1.2 WebComposition Prozess Modell

Das Komponenten-basierte Web Engineering (Component-based Web Engineering, CBWE) befasst sich mit systematischen Ansätzen zum Entwurf und Entwicklung von Web-Anwendung mit Hilfe von Komponenten (Gaedke und Rehse, 2000). Dabei steht das Reuse Repositorium (Gaedke, Rehse und Graef, 1999) im Vordergrund, das sämtliche Entwurfsartefakte durch die unterschiedlichen Phasen hindurch verfügbar macht. Das zugrunde gelegte Vorgehensmodell zeichnet sich durch seinen *evolutionsorientierten* Charakter aus. Dadurch ist es für den Einsatz im Web Engineering äußerst geeignet. Das Modell ist *offen*, d.h. es ermöglicht eine Integration beliebiger Entwicklungsmodelle. Ziel dabei ist, dass eine Unabhängigkeit zwischen der Entwicklung und Wiederverwendung von Komponenten erreicht werden kann. So ließen sich Komponenten innerhalb eines Entwicklungsteams beispielsweise mit Hilfe des Wasserfallmodells realisieren, während ein anderes Entwicklungsteam ein anderes Vorgehensmodell favorisiert.

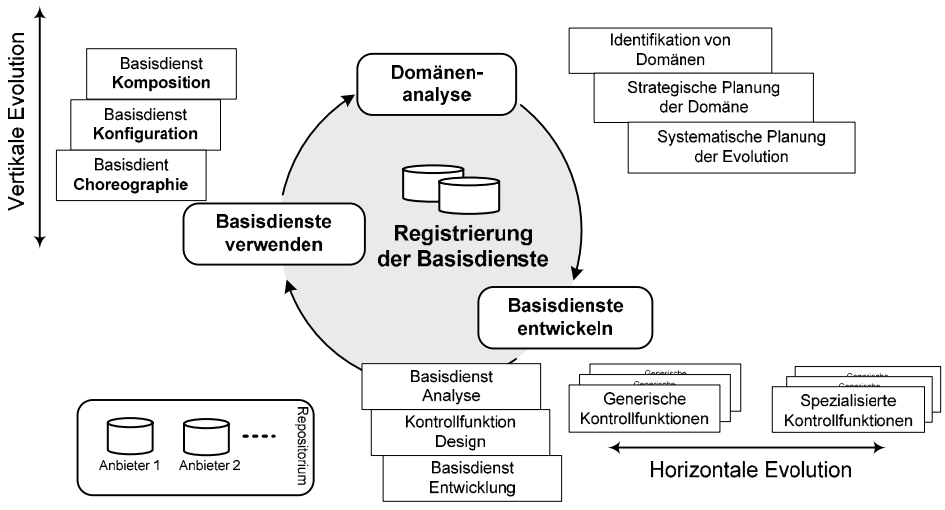


Abbildung 5-1: Das WebComposition Vorgehensmodell als Basis für eine effiziente Entwurfsmethodik für den Einsatz im Notebook-Universität Karlsruhe (TH) Projekt.

Im Innovationsprojekt »Notebook-Universität Karlsruhe« wurde dieses offene Entwicklungsmodell entsprechend der Verwendung von a*nwendungsbezogenen Basisdiensten* (Gaedke, Juling und Nussbaumer, 2004) weiterentwickelt (siehe Abbildung 5-1). Eine Konkretisierung des Basisdienst-Konzepts wird im Rahmen der WSLS-Referenzarchitektur in Abschnitt 5.4 vorgenommen.

In Kombination mit der im Notebook-University Projekt (NUKATH) (Deussen, Juling und Thum, 2004) entwickelten Referenzarchitektur einer Laufzeitumgebung wird eine allgemeine Basis geschaffen, um den konkreten Anforderungen an Entwicklung, Betrieb und Nutzung im Kontext multimedialer Lehr- und Lernsysteme Rechnung zu tragen. Diese Anforderungen ergeben sich zum einen aus der mediengestützten Ausbildung, die durch die Wiederverwendung lose gekoppelter *Dienste* effizient gestaltet werden kann. Zum anderen ergeben sie sich aus den neuen Möglichkeiten kollaborativen, personalisierten und ubiquitären Lernens, also der kontinuierlichen Anpassbarkeit der identifizierten Dienste auf bestimmte vorherrschende Begebenheiten.

Demnach wurden zwei wichtige Bereiche für den Aufbau, Betrieb und die Nutzung dienstorientierter Web-Anwendungen identifiziert:

- Die Konkretisierung eines evolutionsorientierten Vorgehensmodells zur Entwicklung allgemeiner dienstorientierter Web-Anwendungen, mit der Fähigkeit, die dedizierten Anforderungen, die sich für eine Notebook Universität ergeben, umzusetzen.
- Die Konzeption einer Laufzeitumgebung für das Computer-Aided Web Engineering (CAWE) zur effektiven und effizienten Integration von Basisdiensten in einer Web-Anwendung, die für den speziellen Anwendungsfall einer Notebook Universität ausgeprägt werden können.

Der dem WebComposition Ansatz zugrunde liegende *Dienstleistungscharakter* wird durch Entkopplung der konzeptionellen Ebene, des Dienstes, und seiner Konkretisierung, einer Komposition mehrerer Komponenten erreicht (Gaedke, 2000). Ein Dienst stellt damit eine zentrale Einheit dar, die der kontinuierlichen Evolution unterworfen ist. Das in der Notebook-Universität angepasste WebComposition Entwicklungsmodell gibt daher – basierend auf dem evolutionsorientierten Charakter dieser Dienste – drei wesentliche Phasenabschnitte vor (siehe Abbildung 5-1). Diese Abschnitte beschrieben den Lebenszyklus eines Dienstes aus Sicht seiner Evolution.

- **Analyse der Anwendungsdomänen**:
  Die zentrale Aufgabe der Domänenanalyse besteht in der Identifikation und Beschreibung von Anwendungsdomänen. Im konkreten Fall der Notebook-Universität wurden diese Anwendungsdomänen durch die Einflussgrößen der Lehre und des Lernens an einer Universität bestimmt. Entsprechende Anwendungsszenarien umfassten dabei konsequenterweise die Erforschung und Beschreibung diverser Lehr- und Lernszenarien, wie dem Einsatz Mobiler Lehr- und Lernszenarien (Andre, Barthelmeß, Brauch und Deussen, 2004), der Anwendung mobiler Systeme (Bonn, Dieter und Schmeck, 2004) oder der Syndikation und Verteilung lehrbezogener Inhalte (Gaedke, Juling und Nussbaumer, 2004) sowie ihrer evolutionsbezogenen Abhängigkeiten untereinander (horizontale Evolution). Aufgrund der kontinuierlichen Weiterentwicklung der universitären Lehre wird eine gezielte und systematische Evolution der Anwendungsdomänen erwartet. Die Bestimmung dieser Domänen, welche unter dem Einfluss der gezielten Weiterentwicklung – nämlich die Anwendung und Durchfüh-

rung der Evolution – stehen, bildet somit einen unabdingbaren Vorgang innerhalb der Domänenanalyse.

- **Entwicklung von Basisdiensten zur Unterstützung und Realisierung von Anwendungsdomänen**:
  Die Phase der Basisdienstentwicklung umfasst die Analyse einer zuvor identifizierten Anwendungsdomäne hinsichtlich ihrer speziellen Ausprägung und Verwendung von Basisdiensten. Hierzu werden einerseits bereits existierende und wieder verwendbare Basisdienste bestimmt. Zusätzlich werden die neu zu entwickelnden Basisdienste oder Teile dieser Dienste, die *Dienstelemente*, bestimmt. Dienste (*Control Function*) und Dienstelemente (*Service Element*) stellen Abstraktionen für Anwendungsteile in Form von Komponenten dar. Hierbei können auch gezielt Dienstelemente für besonders zu berücksichtigende Eigenschaften von Anwendungsdomänen, wie beispielsweise Unterstützung der Darstellung für Blinde oder sehgeschädigte Menschen (Klaus und Scherwitz-Gallegos, 2004), angepasst werden. Eine Konkretisierung dieser Konzepte findet sich im Abschnitt 5.3.1. Bereits existierende Prozessfunktionalität kann durch das Dienstverzeichnis gefunden und für die Entwicklung neuer Eigenschaften wieder verwendet werden.
  Eine wichtige Facette bei der Entwicklung von Basisdiensten stellt die Möglichkeit der Kommunikation zwischen den einzelnen Beteiligten dar. Im Idealfall werden alle Beteiligten (Stakeholder) eines Projekts in die Lage versetzt direkt etwas zur Gesamtlösung einer bereits in der Domänenanalyse identifizierten Domäne beizutragen. Ein entsprechendes Vorgehen in Form von Domänenspezifischen Sprachen (*Domain Specific Language, DSL*) findet sich im anschließenden Abschnitt 5.2. Es wird ein Rahmenwerk basierend auf DSLs vorgestellt, das die Kommunikation zwischen den einbezogenen Stakeholdern unterstützt und es gestattet, Lösungsstrategien in Form Domänenspezifischer Sprachen zu entwerfen.

- **Verwendung von Basisdiensten nach dem *Baukastenprinzip* zur Realisierung der Web-Anwendung**:
  Um eine einheitliche Integration der Basisdienste gewährleisten zu können, bedarf es einer Laufzeitumgebung, die die speziellen Eigenschaften und Ausprägungen der entwickelten Basisdienste berücksichtigt. Auf diese Weise können die entstandenen Basisdienste durch Verwebung zu dienstorientierten Web-Anwendungen nach dem Baukastenprinzip zusammengefügt werden. Im Zuge einer solchen Komposition müssen unter anderem Aspekte wie das Zusammenspiel und die funktionalen Abhängigkeiten von Basisdiensten genauso Berücksichtigung finden wie Aspekte Sicherheit, Qualität und unterschiedlicher Konfigurationsmöglichkeiten. Bestimmte Komponenten und Komponentenverbünde (Kontrollfunktion und Dienstelemente) können als »maturierter Verbund« wiederum als eigenständige Komponente angeboten werden. Neben der Fähigkeit der Maturität (Gaedke, 2000) durch Analyse der *Verwendung* von Komponenten wird zusätzlich auch die Maturität ganzer Anwendungsgruppen betrachtet.

Basierend auf der durch das WebComposition Prozess Modell gegebenen Grundlage wird im Folgenden der WSLS-Ansatz schrittweise entwickelt. Dazu wird im nächsten Abschnitt zunächst das Paradigma der Konfiguration vorgestellt, das eine wesentliche Grundlage für die WSLS Laufzeitumgebung darstellt. Dabei wird basierend auf der Fähigkeit der Konfiguration eine geeignete Klasse von Beschreibungssprache, den Domänenspezifischen Sprachen, ent-

wickelt. Ein Vorgehensmodell sichert deren systematischen Einsatz; im Besonderen hinsichtlich der Kommunikation zwischen Projektbeteiligten und ihrer Wiederverwendung.

Im Anschluss daran wird die WSLS-Referenzarchitektur beschrieben, deren dienstorientierter Charakter sich in der Definition fachlicher und konzeptueller Komponenten ausdrückt. Um eine ideale Unterstützung der Evolution ihrer Dienste und Komponenten zu gewährleisten werden Konzepte aus dem Bereich der Aspekt-orientierten Programmierung auf Web-spezifische Problemcharakteristika übertragen. Diese Konzepte werden dann in Form der Definition des *Anwendungsspezifischen Basisdienstes* modellhaft synthetisiert.

5.2 Konfigurieren anstatt Programmieren

5.2.1 Ausgangspunkt Domänenspezifische Modellierungssprachen

Klar und zutreffend spezifizierte Anforderungen sowie die kontinuierliche und intensive Zusammenarbeit mit allen betroffenen Stakeholdern gehören zu den fünf bedeutendsten Erfolgsfaktoren in Softwareentwicklungs-Projekten (The Standish Group International, 1994-2005; Cutter Consortium, 2000). Eine funktionierende und effektive Kommunikation stellt dabei die Grundlage zur Umsetzung dieser Faktoren dar. Besonders im Bereich der Anforderungsanalyse und des konzeptionellen Entwurfs ist das Vermeiden von Missverständnissen zwischen allen Projektbeteiligten unabdingbar.

In den letzten Jahren sind im Bereich des Web Engineering eine Vielfalt an Methoden entstanden, die eine möglichst umfassende, systematische und teilweise formale Spezifikation der verschiedenen Aspekte einer verteilten webbasierten Lösung anstreben (siehe Abschnitt 3.2). Diese Ansätze zielen mit ihren Modellen meist die Eigenschaften und Problemstellungen des Web Engineering so umfassend wie möglich abzudecken und bieten hierfür teilweise eine sehr umfangreiche Menge an Methoden und Notationen. Aufgrund der damit verbundenen Mächtigkeit und Ausdruckskraft der Modelle eignen sich diese jedoch allenfalls zur Spezifikation und als Mittel zur Kommunikation innerhalb eines geschulten Entwicklungsteams. Allerdings sind sie weniger geeignet in Entwicklungsprojekten, bei denen sehr unterschiedliche Stakeholder mit unterschiedlichen technischen Hintergründen involviert sind. Diese Mannigfaltigkeit an unterschiedlichen Nutzergruppen findet sich nachgerade in Projekten zur Integration von Anwendungen oder Informationen mit dem Ziel eine Service-orientierte Architektur umzusetzen. In Abschnitt 3.3 wurde die Notwendigkeit für leichtgewichtige Modellansätze mit visuell geprägter Entwicklungsunterstützung als eine wichtige Voraussetzung für die Forderung einer Flexibilisierung des Entwicklungsprozesses gewonnen.

Domänenspezifische Sprachen als Mittler zwischen Modell und Wirklichkeit

Um diese Anforderung zu erfüllen, bedarf es allerdings einer Umorientierung bezüglich der Granularität von zu modellierenden Elementen durch die explizite Berücksichtigung und Einbeziehung der entsprechenden Nutzergruppen. Anstatt einen allumfassenden Ansatz zu verfolgen, wie es durch die rein modellgetriebenen Ansätze geschieht (MDA, Modell-Driven Architecture) geschieht, bedarf es – gerade aufgrund der starken Einbeziehung von Men-

schen – geeigneter Problemlösungsstrategien, die auf bestimmte, in sich abgeschlossene Bereiche einer Problemdomäne fokussieren. Diese Fokussierung ermöglicht einen guten Ausgangspunkt zu einer weiteren deduzierten Beschreibung unterschiedlicher Gruppen von Menschen, die an der Lösung einer Problemdomäne beteiligt sein können. In (Aoyama, 2005) wird dazu beispielsweise das Persona-Konzept entwickelt, um verschiedene Gruppen und ihre Basiseigenschaften zu definieren.

Eine Alternative zu den gängigen rein modellgetriebenen Ansätzen stellen Domänenspezifische Sprachen (Domain-specific Languages, DSL), manchmal auch als »Little Languages« (Bentley, 1986) bezeichnet, dar. Deursen, Klint und Visser definieren sie als Programmiersprachen oder ausführbare Spezifikationssprachen, die ihre Ausdrucksstärke durch die Verwendung geeigneter Notationen und Abstraktionen unter Fokussierung und Beschränkung auf eine klar definierte Problemdomäne erlangen (Deursen, Klint und Visser, 2000). Aufgrund ihrer starken Fokussierung auf einen lediglich kleinen Aspekt der zu entwickelnden Lösung – kombiniert mit den Möglichkeiten grafischer Notationen aus der Problemdomäne – sind DSLs *einfach zu erlernen*, *einfach zu verstehen* und *einfach einzusetzen*.

Dies gilt für die Entwickler gleichermaßen wie für Kunden oder sogar Endanwender. Durch die Nutzergruppen-spezifische Integration dedizierter grafischer Notationen und begleitender Editoren kann die Verständlichkeit und damit die »Benutzerfreundlichkeit« einer DSL noch weiter verbessert werden. DSL-Programme können ähnlich wie bei herkömmlichen Programmiersprachen C++, Java oder C# mittels eines speziellen DSL-Compilers oder Interpreters in ausführbaren Code transformiert werden.

Abgrenzung zu Modellgetriebenen Ansätzen

Wichtig ist dabei, dass eine DSL nicht als Ersatz oder gar Konkurrenz zu Modellgetriebenen Ansätze (*Model Driven Software Development*, MDSD) angesehen werden sollte, sondern vielmehr als eine sinnvolle Ergänzung. So prognostiziert die Gartner Group, dass der Einsatz von DSLs bis ins Jahr 2012 auf 30% im Vergleich zu ca. 15% im Jahr 2006 zunehmen wird (Norton, 2006). In (Schwinger und Koch, 2003) wird auf die zu einem gewissen Grade *fehlende* Agilität eines modellgetriebenen Ansatzes für das Web Engineering in Hinsicht auf kurze Entwicklungszyklen verwiesen. Dieser Problematik können DSLs durch ein dediziertes Rahmenwerk, das ein Vorgehen zur Entwicklung und Verwendung unter Berücksichtigung der beteiligten Entwickler- und Nutzergruppen vorschreibt, besser entgegentreten (Nussbaumer, Freudenstein und Gaedke, 2006b).

Ziel dieses Ansatzes ist es, Stakeholder und Domänenexperten durch den Einsatz von DSLs in die Lage zu versetzen, Aspekte eines verteilten, webbasierten Systems eigenständig zu verstehen, zu validieren, zu ändern und sogar selbst zu entwickeln. Regelmäßig auftretenden Missverständnissen und Schwierigkeiten bei der Kommunikation kann durch eine dadurch ermöglichte, intensive Integration von Stakeholdern in den Entwicklungsprozess begegnet werden.

5.2.2 Das WebComposition DSL-Rahmenwerk

Abbildung 5-2 stellt das evolutionäre, auf dem WebComposition Ansatz basierende Prozessmodell für die DSL-basierte Entwicklung von dienstorientierten Web-Anwendungen vor. Dabei werden drei Phasen im Zuge der Entwicklung von DSLs unterschieden:

- **DSL Analyse***:* Die semantische Konzeption umfasst das Design und den Entwurf einer Sprache als Ausgangspunkt für zugehörige Abstraktionsebenen und Bearbeitungswerkzeuge. Sie dient ferner der Wiederverwendungsanalyse, ob eine passende Sprache existiert und wiederverwendet werden kann.
- **DSL Entwicklung***:* Die Entwicklung für die Wiederverwendung umfasst die syntaktische Konzeption der einzelnen Bestandteile, die zur Konstruktion der DSL benötigt werden. Eine effektive Anpassung und Bereitstellung dieser Anwendungsbestandteile wird durch dediziertes Konfigurieren von Experten (Domänenexperten, Stakeholder) durchgeführt.
- **DSL Verwendung**: Die Entwicklung für Wiederverwendung umfasst die effiziente Verknüpfung existierender Bausteine (*Solution Building Blocks*) zur Generierung von Web-Anwendungen. Bausteine werden mittels DSL Programmen konfiguriert, die durch Verwendung der entsprechenden Interaktionswerkzeuge erstellt wurden.

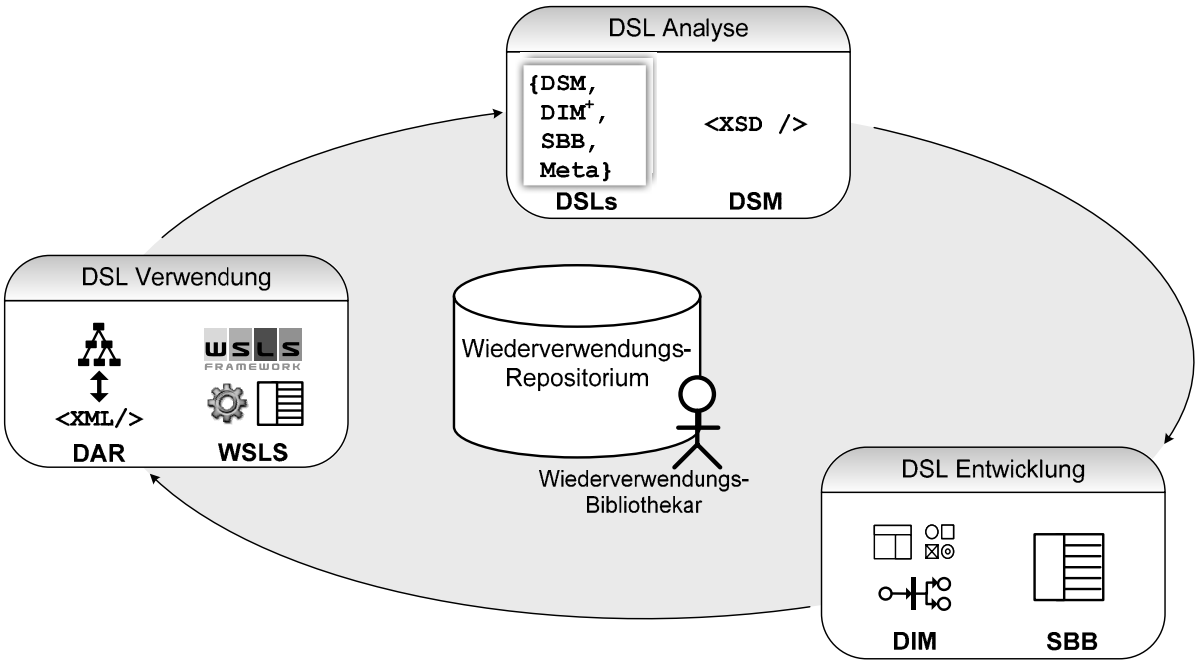


Abbildung 5-2: Das WebComposition DSL-Rahmenwerk als Ausgangspunkt für eine DSL-basierte Entwicklung von Web-Anwendungen.

DSL Analyse

In der *Analyse*-Phase wird im Bestand der verfügbaren DSLs nach einer für den gegebenen (Problem-)Kontext geeigneten DSL recherchiert. Dabei können verschiedene Kontext-Parameter wie beispielsweise der zu spezifizierende Aspekt der Web-Anwendung, die Charakteristika der mitarbeitenden Stakeholder oder die Art des zu konzipierenden Systems in die Suche mit einfließen. Zusätzlich bietet sich hier die Möglichkeit auf das bereits existie-

rende Wissen der Analyse der Anwendungsdomänen aus Abschnitt 5.1.2 zurückgegriffen werden. Zur Konzeption eines derartigen Repositoriums sowie der Realisierung der Suche, Identifikation und Selektion von Elementen sei auf (Gaedke, Rehse und Graef, 1999) verwiesen.

Bleibt die Suche erfolglos, d.h. wurde keine passende DSL gefunden, bedarf es einer Neuentwicklung, respektive Anpassung, einer bereits existierenden DSL. Ist dies der Fall, wird in enger Zusammenarbeit mit den am Projekt beteiligten Stakeholdern die Problemdomäne analysiert und geeignete Abstraktionen und Konzepte zur Beschreibung von Lösungen innerhalb der Domäne identifiziert. Das Ergebnis dieser Phase bildet das *Domain-Specific Model (DSM)*[50] kurz DSL-Modell. Es ist dabei unerheblich ob es im Repositorium gefunden, angepasst bzw. neu entwickelt wurde.

Beim DSL-Modell handelt es sich um eine strukturell wohldefinierte Schematisierung aller Lösungen, die mit der DSL innerhalb ihrer Problemdomäne spezifiziert werden können. Als Ausgangspunkt für die Modellierung werden XML Schema Definitionen (XSD) verwendet. Die Methode ist offen bezüglich der Findung und Konkretisierung eines spezifischen Modells. Es können unterschiedliche weitere Ansätze verfolgt werden, die in Einklang mit entsprechendem Vorwissen und Befähigungen des Projektteams stehen müssen. Beispiele für unterschiedliche Ansätze finden sich in (Wada und Suzuki, 2005).

Entwicklung Domänenspezifischer Sprachen

Die Phase »DSL Entwicklung« setzt auf der »Analyse«-Phase auf und wird im Falle von Änderungen bzw. Neuentwicklung einer DSL durchlaufen. Wurde hingegen in der vorhergehenden Phase eine geeignete existierende DSL gefunden, kann die Phase übersprungen werden. Wird eine ähnliche Sprache gefunden, kann diese angepasst und dediziert weiterentwickelt werden. Die daraus resultierende *kontinuierliche Evolution* von DSLs (Nussbaumer, Freudenstein und Gaedke, 2006c) führt zu einer beständigen Spezialisierung der DSL und dadurch ihrer Verbesserung hinsichtlich ihrer konkreten Aufgabe.

Basierend auf dem DSM werden zunächst ein oder mehrere *Domain Interaction Model (DIM)* entwickelt. Ein DIM definiert zu den im DSM spezifizierten Abstraktionen und Konzepten eine dedizierte Notation, die für eine gegebene Stakeholder-Gruppe so intuitiv wie möglich sein sollte. Dabei muss das DIM nicht notwendigerweise alle Aspekte des DSM abdecken. In manchen Fällen ist für eine bestimmte Zielgruppe nur ein Ausschnitt des Modells relevant. Dann kann ein für diesen Bereich und die entsprechende Zielgruppe zugeschnittenes Interaktionswerkzeug bereitgestellt werden.

Mit Hilfe dieser Notationen werden Stakeholder in die Lage versetzt die DSL entsprechend ihren Möglichkeiten und Voraussetzungen anzuwenden. Das bedeutet sie werden in die Lage versetzt auf dem DSM basierende DSL-Programme besser verstehen, validieren und sogar selbst entwickeln zu können, ohne dabei mit komplizierterem Quellcode konfrontiert zu werden. Die Konzepte und Notationen eines DIM müssen deshalb in Zusammenarbeit mit den Stakeholdern aus der Problemdomäne abgeleitet werden, so dass sie für diese möglichst

[50] Der Begriff DSM wird in der einschlägigen Literatur bisweilen auch mit Vorgang des Modellierens mit Hilfe einer DSL, dem Domain Specific Modeling, verbunden. Im Kontext des WebComposition DSL-Rahmenwerks bildet er nur die Basis für die Modellierung, indem lediglich das sprachliche Modell beschrieben wird.

einfach zu erlernen und anzuwenden sind. Bei der Erstellung der Notationen sollte grundsätzlich Wert auf Einfachheit gelegt werden, beispielsweise der Gestalt, dass darauf basierende Modelle mit Papier und Bleistift gezeichnet werden können. Dadurch kann im Vergleich zu Modellen, die aufgrund ihrer Komplexität eher für den Entwurf am Computer geeignet sind, eine höhere Akzeptanz und schnellere Erlernbarkeit realisiert werden (Nielsen, 1993). Um den verschiedenen Anforderungen und Charakteristika verschiedener Stakeholder-Gruppen entsprechen zu können, kann innerhalb einer DSL für jede Gruppe ein eigenes DIM bereitgestellt werden. Durch begleitende, auf dem DIM basierende grafische Editoren kann deren Anwendbarkeit und Effektivität noch weiter erhöht werden.

Neben der Entwicklung eines oder mehrerer DIMs wird in der Phase »Entwicklung für Wiederverwendung« ein sog. *Solution Building Block (SBB)* für die DSL entwickelt. Ein SBB ist eine Fachkomponente, die für die Ausführung der mit einer DSL erstellten Programme zuständig ist. Dabei fungiert der SBB nicht zwangsläufig als Code-Generator, sondern als eine Komponente, deren Verhalten mittels eines DSL-Programms konfiguriert werden kann und so das gewünschte Verhalten effektiv umsetzt. Die daraus entstehende Nähe zu einer konkreten Implementierungsplattform findet vor allem in jüngerer Zeit mehr und mehr Aufmerksamkeit (Imbusch, Langhammer und Walter, 2005). Die Attraktivität liegt in der effizienten Lösbarkeit existierender Probleme; weniger im modelltheoretischen Ansatz zunächst alle Probleme zu lösen. Im Zuge dessen ist das *WebComposition Service Linking System (WSLS)* (Gaedke, Nussbaumer und Meinecke, 2005) entstanden, das als Referenzarchitektur fungiert für eine technische Plattform, die in der Lage ist SBBs auszuführen. WSLS setzt auf den Grundprinzipien »Wiederverwendung« und »Evolution« des WebComposition Ansatzes auf und ermöglicht die systematische Evolution von Web-Anwendungen durch explizite Wiederverwendung hochgradig konfigurierbarer Software-Komponenten. Die Realisierung des Prinzips »Konfigurieren statt Programmieren« stellt dabei eines der Hauptziele von WSLS dar. Im Hinblick auf die SBBs erleichtert das WSLS-Rahmenwerk deren systematische Komposition und Konfiguration zu vollständigen Web-Anwendungen.

Verwendung Domänenspezifischer Sprachen

In der Phase »*Entwicklung mit Wiederverwendung*« wird mit Hilfe der gefundenen oder angepassten bzw. neu entwickelten DSL Komponenten (dem Domain-Specific Model, den Domain Interaction Models und dem Solution Building Block) ein DSL-Programm zur Spezifikation eines Aspekts einer Web Anwendung entwickelt. Wir bezeichnen ein solches auf dem DSM basierendes und mit Hilfe der Notationen aus einem oder mehrerer DIMs entwickeltes DSL- Programm auch *Domain Abstract Representation (DAR)*. Ein DAR stellt die Spezifikation einer konkreten Lösung innerhalb der Problemdomäne der DSL dar. Da ein DSM in der Regel in Form eines XML Schema Dokuments vorliegt, wird dementsprechend ein DAR in einem darauf basierenden XML Dokument serialisiert und gespeichert.

Das entwickelte DAR wird anschließend in seiner XML Repräsentation an den Solution Building Block der DSL übergeben. Dieser adaptiert sein Verhalten entsprechend dem DSL-Programm und bringt es dadurch (indirekt) zur Ausführung. Die Entwicklung von Web-Anwendungen kann somit im Zuge ihrer Evolution durch die Komposition von SBBs und deren Konfiguration mit DARs durchgeführt und entsprechend angepasst werden.

Wiederverwendungs-Repositorium

Die entstehende Vielzahl an DSLs für die unterschiedlichen Aspekte dienstorientierter Web-Anwendungen, aber auch die DSLs selbst, unterliegen der ständigen Evolution durch Variation und Selektion. So können die Erfahrungen aus dem Einsatz von DSLs mit unterschiedlichen Stakeholder-Gruppen zu Verbesserungen bzw. Änderungen (Variation), Neuentwicklungen oder gar dem Verwerfen (Selektion) von DSLs führen.

Um den Einsatz und die Verwaltung der DSLs in einem solch evolutionären Umfeld möglichst systematisch und effizient zu gestalten, sieht der Ansatz die Einrichtung eines zentralen *Wiederverwendungs-Repositorium* (Gaedke, Rehse und Graef, 1999) vor. Dieses Repositorium dient dabei als zentraler Ort zur Speicherung, Verwaltung und Suche von DSLs und zugehöriger Meta-Information. Während des gesamten Prozesses werden neu entwickelte oder angepasste Artefakte sowie Meta-Information wie zum Beispiel Erfahrungen im Umgang mit einer DSL systematisch abgelegt. Darüber hinaus werden beispielsweise in der Analyse-Phase bestehende DSLs abhängig von durch den Anwendungs- und Projektkontext vorgegebenen Kriterien im Repositorium gesucht. Versionierungs-Funktionalitäten, ein geeignetes Klassifikationsschema sowie entsprechende Suchfunktionalitäten sind daher für das Wiederverwendungs-Repositorium unabdingbar.

Unterstützung des effizienten Findens von DSLs

Ein guter Ausgangspunkt findet sich in einem von Prieto-Diaz vorgeschlagenen Ansatz zur Aspektklassifikation (*Facetted Classification*) (Prieto-Díaz, 1991). Ursprünglich wird dieser Ansatz zur Bestimmung der Ähnlichkeit von Softwarekomponenten verwendet. Er beruht auf einer vordefinierten Menge an Schlüsselworten, welche von den beteiligten Domänenexperten aus den zugehörigen Modellen einer DSL (DSM und DIM) konsequent extrahiert werden können. Das ursprüngliche Vorgehen von Prieto-Diaz sieht dabei drei Phasen vor:

- *Accessing*: Zugriff auf existierende Modelle und Notationen,
- *Understanding*: Verstehen der Modelle und
- *Adapting*: Anpassen der Modelle.

Die Schlüsselworte werden nach verschiedenen Aspekten in ein Klassifizierungsschema eingeordnet und als Beschreibung für die DSL verwendet. Als Ausgangspunkt für die Aspektklassifikation dienen die als grundlegende Einflussgrößen erkannten Aspekte *Daten*, *Darstellung*, *Navigation*, *Dialog*, *Prozess* und *Kommunikation*. Ein Thesaurus[51] wird für jeden Aspekt abgeleitet, um synonyme Bezeichnungen für die Schlüsselworte zu berücksichtigen. Dieser Ansatz ist gerade hinsichtlich der großen fachlichen Diversifikation und unterschiedlicher Hintergründe einzelner Stakeholder im Falle von DSLs vielversprechend, da die späteren Nutzer gleichzeitig an der Entwicklung dieser Thesauri beteiligt werden können. Die Menge initialer Schlüsselwörter kann so direkt während des Entwurfs der DSL unter Beteiligung der Stakeholder befüllt werden. Die zu erwartenden begrifflichen Uneindeutigkeiten während des DSL Entwurfs können direkt zur Pflege des Thesaurus verwendet werden.

[51] Ein Thesaurus repräsentiert ein Modell, das es anstrebt, ein Themengebiet möglichst genau zu beschreiben. Es besteht aus einer systematisch geordneten Sammlung von Begriffen, die thematisch miteinander assoziiert sind.

Schlüsselworte können nur innerhalb des Kontexts ihres zugehörigen Aspekts benutzt werden; Uneindeutigkeiten werden durch den Thesaurus aufgelöst. Ein wichtiger formaler Bestandteil des Ansatzes stellt die Ausnutzung des Konzeptabstandsgraphen. Diese Abstände zwischen den Schlüsselworten der Aspekte können herangezogen werden, um ein Ähnlichkeitsmaß zu definieren (Ostertag, Hendler, Prieto-Díaz und Braun, 1992). Das wiederum kann zur Ähnlichkeitsbestimmung zwischen DSLs oder gegebenenfalls auch ihren Bestandteilen herangezogen werden.

In der Spezifikation des WebComposition Wiederverwendungs-Repositorium (Gaedke, 2000) wird der Algorithmus von Prieto-Díaz verwendet und dahingehend verallgemeinert, dass keine konkrete Menge an Schlüsselwörtern vorausgesetzt wird. Vielmehr wird unter Einbeziehung eines Benutzers die Anfrage sukzessive spezifiziert. In der Realisierung des WebComposition Wiederverwendungs-Repositorium geschieht dies durch eine grafisch gestützte hypermediale Anwendung. Dadurch werden die Benutzer in die Lage versetzt ihre Anfragespezifikationen auf Basis der dieser Anfrage passenden DSLs sukzessive zu verfeinern, um den Anfrage-Suchraum weiter einzugrenzen. Diese können dann ausgehend von den im vorigen Zwischenschritt gewonnenen Erkenntnissen ihre Anfrage entsprechend ändern.

Die Verwaltung eines komplexen Wiederverwendungs-Repositoriums muss als prozessübergreifende Aufgabe auch personelle Berücksichtigung finden (Gaedke, Rehse und Graef, 1999). Daher wird zur Koordination die Team-Rolle des *Wiederverwendungs-Bibliothekars* eingeführt. Sie ist für die systematische Verwaltung und den optimalen Einsatz der DSLs zuständig. Sie begleitet das Projekt-Team durch den gesamten Entwicklungsprozess und achtet bei der Entwicklung und der Auswahl von DSLs auf effiziente Wiederverwendung. Darüber hinaus steht sie bei der Suche und Anwendung von DSLs dem Team beratend zur Seite und ist für die Administration des Wiederverwendungs-Repositoriums sowie die Konservierung und Bereitstellung gewonnenen Wissens verantwortlich.

5.3 Die WSLS-Referenzarchitektur

5.3.1 Component-Based Web Engineering (CBWE)

Eine Lösung für die effiziente Wiederverwendung ist der Einsatz von Komponententechnik. Der Begriff (Software-) Komponente wird mit teilweise sehr stark abweichenden Definitionen verwendet (Braun, 1994; Sametinger, 1997). Eine mögliche und weitverbreitete Definition gibt Szyperski (Clemens Szyperski, Gruntz und Murer, 2002):

»A software component is a unit of composition with contextually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition.«

Die dabei verfolgte Idee ist es, neue Anwendungen zu möglichst großen Teilen aus schon existierenden eigenständigen Software-Artefakten nach dem Baukasten-Prinzip zusammenzusetzen. Dies Zusammensetzung sollte idealerweise möglichst einfach und schnell funktionieren, so dass Entwicklungskosten gespart werden können. Zur Laufzeit wird über standardisierte Schnittstellen auf die Komponenten zugegriffen, welche ihre Implementierung nach

dem Black-Box-Prinzip verstecken. Neben der Wiederverwendung von Programmcode ergibt sich als weiterer Vorteil die Möglichkeit, feingranulare Konzepte des Entwurfs in der Implementierung zu repräsentieren (Gaedke, Turowski und Rehse, 1999). Eine derartige Repräsentation wirkt sich besonders positiv bei Änderungen am Entwurf aus, die sich dann mit wesentlich geringerem Aufwand auf Änderungen an der Implementierung übertragen lassen.

Die Idee, die hinter dem Entwurf von Software aus Komponenten, dem Component-based Software Engineering (CBSE) (Alan W. Brown, 1996), steckt wird seit mehreren Jahrzenten diskutiert. Die Idee der Komponente ist dabei so alt wie die Softwaretechnik selbst; so finden sich grundlegende Aussagen zu Komponenten bereits in einem Beitrag von McIlroy auf der Software Engineering Konferenz von 1968 (McIlroy, 1968). Der Trend hin zu komponentenorientierten Ansätzen zieht sich von da an wie ein roter Faden durch die Weiterentwicklung der Softwaretechnik. In den 90er Jahren schuf Microsoft mit seinem Komponentenmodell COM (Microsoft, 1995) und den Nachfolgemodellen COM+ und DCOM durch die Integration in die Windows-Betriebssysteme einen kommerziellen Markt für diesen Bereich und ebnete damit den Weg für einen Siegeszug der Komponententechnologie.

Auch im Web Engineering lassen sich die Erkenntnisse dieser Disziplin Gewinn bringend einsetzen. In (Gaedke und Rehse, 2000) wird dafür der Begriff *Component-Based Web Engineering* (CBWE) definiert:

»The cost-effective production of (high-quality) Web applications using a defined process that includes systematic reuse of components and of domain knowledge.«

5.3.1.1 Das WSLS Komponentenmodell

Das WebComposition Service Linking System (WSLS) basiert auf der Idee des Component-Based Web Engineering. Es definiert eine Referenzarchitektur für den Entwurf und die Realisierung von dienstorientierten Web-Anwendungen auf Basis von Komponenten. Die Ziele, die dieser Architektur zu Grunde liegen, sind:

- Entwicklung von Web-Anwendungen durch Komposition lose gekoppelter Dienste.
- Unterstützung der Evolution der Web-Anwendungen durch Konfigurierbare Konvertibilität.
- Erleichterung der Wiederverwendung von Komponenten und Lösungsbausteinen.
- Offenes Anwendungsmodell, das die Hinzunahme zusätzlicher Entwicklungsphasen ermöglicht.

Komponenten symbolisieren den Übergang eines abstrakten Anwendungskonzeptes – wie dem eingangs vorgestellten Basisdienst – hin zu einer ausführbaren Anwendungsplattform. Dabei ist wichtig, dass kein Bruch zwischen der abstrakten Modellebene und ihrer Repräsentation im konkreten plattformspezifischen Modell besteht. Dies bedeutet insbesondere, dass eine Web-Anwendung zu jedem Zeitpunkt eine Entsprechung ihres konzeptuellen abstrakten Modells bildet.

Die Konkretisierung einer identifizierten *Anwendungsdomäne* wird durch die Modellierung durch Komponenten erreicht. Die WSLS-Referenzarchitektur unterscheidet dabei zwei grundsätzliche Typen dieser Komponenten:

- fachliche Komponenten (Anwendungsspezifischer Basisdienst) *und*
- konzeptuelle Komponente (Domäne).

Eine *konzeptuelle Komponente* beschreibt einen Teilbereich des abstrakten Konzepts einer Web-Anwendung. Sie eignet sich als Modellierungseinheit und Kommunikationsgegenstand für die beteiligten Nutzergruppen. Aufgrund ihrer abstrakten Natur eignen sich konzeptuelle Komponenten ideal als Integrationspunkte des in Abschnitt 0 beschriebenen DSL-Ansatzes. Sie ermöglichen die Aufnahme eines funktional-fachlichen Konzepts durch Hinzukonfigurieren lose gekoppelter Grundbausteine. Die im Rahmen dieses Ansatzes beschriebenen Lösungsbausteine (Solution Building Blocks, SBB) bilden dabei eine spezialisierte Untermenge der fachlichen Komponenten, die in WSLS-Referenzarchitektur existieren. Sie sind in der Lage ausführbaren Programmcode aus konfigurierbaren DSL Programmen zu erzeugen.

Eine *fachliche Komponente* bildet als konkrete funktionale Einheit einen Grundbaustein zur Erbringung der gesamten Web-Anwendung. Sie löst eine bestimmte Aufgabe innerhalb einer Anwendungsdomäne. Eine fachliche Komponente ist plattformspezifisch und liegt als ausführbare und abgeschlossene Einheit vor. Sie lehnt sich an die Definition des im Rahmen der Arbeitsgruppe 5.10.3 erarbeiteten Memorandums für Fachkomponenten an (Turowski, 2002). Die darin beschriebenen Merkmale bilden einen guten Ausgangspunkt für eine vereinheitlichende Spezifikation von fachlichen Komponenten. Allerdings finden sich die Ergebnisse dort eher auf einer sprachlichen Ebene angesiedelt. Ein wesentliches Hindernis stellen nach (Fettke, Loos und Tann, 2001) fehlende methodische Hilfestellungen dar. Diese sollten in der Art entwickelt werden, dass ein Vorgehensmodell zur Verfügung gestellt wird, aus dem hervorgeht, in welcher Reihenfolge konkrete Spezifikationsschritte durchzuführen sind. Ein solches Vorgehensmodell sollte darüber hinaus ebenso inhaltliche Aussagen treffen, nach welchen Kriterien die gesamte Funktionalität eines Anwendungssystems im Hinblick auf einzelne Komponenten aufgeteilt werden könnte.

5.3.1.2 Funktionale Prägung durch fachliche Komponenten

Der Vorgang der funktionalen Prägung einer Domäne entspricht der Bereitstellung eines *funktionalen Konzepts*, während die Domäne selbst den nicht-funktionalen Charakter beschreibt. Diese Trennung ermöglicht eine verbesserte Wiederverwendbarkeit von Fachkomponenten und Domänen und erlaubt eine anwendungsnahe konzeptuelle Modellierung, die eine rasche Anwendungskonstruktion durch Aufprägen funktionaler Konzepte in Form von Komponenten ermöglicht.

Die Wiederverwendung der Domänen erlaubt eine flexible (Re-)Organisieren von Web-Anwendungen auf Basis dieser nicht-funktionalen Anwendungskonzepte, die durch die Domänen ausgedrückt werden. Außerdem unterstützt die funktionale Trennung eine bessere Verteilung unterschiedlicher Aufgaben, wodurch Domänenexperten besser in den Entwicklungsprozess der Web-Anwendung einbezogen werden können. Dadurch ergibt sich beispielsweise die Möglichkeit, Teile des Entwicklungsprozesses zu parallelisieren, was sich vorteilhaft auf die gesamte Produktionszeit auswirkt.

Außerdem kann die grundlegende Funktionalität der Komponenten weiterentwickelt werden. Dazu können bereits etablierte Verfahren wie die Anwendung des Objekt-orientierten

Paradigmas oder Aspekt-orientierte Modellierung zur funktionalen Weiterentwicklung eingesetzt werden.

Durch eine weitere fachliche Trennung der funktionalen Aspekte in Gruppen relevanter Einflussgrößen wie Informationen, Darstellung, Dialog und Navigation kann der diese Parallelisierung weiter ausgeweitet werden. Um eine fachliche Trennung dieser Aspekte zu gewährleisten, bedarf es geeigneter Beschreibungsformen, die dann – in einer entsprechenden Umsetzung als Komponenten – die funktionalen Aufgaben erledigen. Zusätzlich muss gewährleistet werden, dass diese elementaren Einflussgrößen in einer sinnvollen Weise miteinander zusammenarbeiten, bzw. verwoben werden. Dazu bedarf es einer geeigneten *Mediation*, die es gestattet, die individuellen Charakteristika der konkreten Elemente miteinander in Beziehung zu setzen und so einen nach außen hin einheitlichen Zustand zu verkörpern.

Die daraus resultierenden Definitionen zur Lösung der aufgeführten Problematik wird in (Gaedke, Nussbaumer und Meinecke, 2004) als *Anwendungsspezifischer Basisdienst* definiert und im Abschnitt 5.4 vorgestellt.

5.3.1.3 Wiederverwendungsorientierte Konstruktion von Web-Anwendungen

In Abschnitt 5.1.2 wurde das Zusammenwirken der einzelnen Phasen des WebComposition Prozess Modells *Identifizieren*, *Entwickeln* und *Verwenden* auf Basisdienste vorgestellt. Im Anschluss daran wurde ein Verfahren entwickelt, das die Beschreibung von fokussierten Problembereichen mit Hilfe Domänenspezifischer Sprachen gestattet. Eine zentrale Rolle spielen dabei die Komponenten zur Ausführung dieser DSLs.

Die folgenden Abschnitte beschäftigen sich mit der Frage, wie diese Bausteine grundsätzlich beschaffen sein müssen, um eine effiziente Konstruktion von Web-Anwendungen durch Komponenten zu ermöglichen. Die Web-Anwendungskonstruktion mit Hilfe der fachlichen und konzeptuellen Komponenten vollzieht sich dabei in zwei verschiedenen Stufen.

1. **Realisierung domänenspezifischer Baugruppen**: Ein Anwendungskonzept wird auf Basis von konzeptuellen Komponenten (Domänen) erstellt und dann durch fachliche Komponenten geprägt. Fachliche Komponenten sind entweder spezielle Lösungsbausteine (SBB, Solution Building Blocks), die durch DSL konfigurierbar sind oder bestimmte anwendungsbezogene Grundbausteine, die eine dedizierte Aufgabe lösen (Anwendungsspezifische Basisdienste). Existierende Grundbausteine werden wiederverwendet und angepasst; nicht existierende Bausteine werden neu entwickelt. Die so entstehende Web-Anwendung wird damit sukzessive durch maturierte Komponentenverbände, den *Anwendungsbausteine (ABB, application building blocks)*, konstruiert. Die damit erreichte Form der Wiederverwendung entspricht der Strategie *Entwicklung für Wiederverwendung*.
2. **Serienfertige Realisierung von Web-Anwendungen**: Bereits konstruierte Bausteine in Form von *Anwendungsbausteinen* werden zu einer Web-Anwendung kombiniert. Diese serienfertigen Komponenten bilden das Rückgrat, um eine Web-Anwendung aus fertig konstruierten Elementen zusammenzusetzen. Auf diese Weise lassen sich Web-Anwendungen schnell und effizient fertigen und gegebenenfalls nachträglich noch anpassen. Das Prinzip, das hinter dieser Form der Realisierung steckt, wird auch als »off-the-shelf«-Produktion bezeichnet und findet sich in unterschiedlichen Varianten wieder. Die damit erreichte Wiederverwendungsstrategie entspricht der *Entwicklung mit Wiederverwendung.*

Die Konstruktion der Benutzungsschnittstelle der Web-Anwendung mit Hilfe der konzeptuellen Komponenten wird unter Berücksichtigung der dafür wichtigen Aspekte Darstellung, Dialog und Navigation im anschließenden Kapitel 6 vorgestellt.

Im folgenden Abschnitt wird zunächst der technisch-funktionale Rahmen für die fachlichen Komponenten entwickelt. Dieser umfasst die Definition der Anwendungsspezifischen Basisdienste und deren Aufbau aus den atomaren Elementen Kontrollfunktion und Dienstelement.

5.4 Anwendungsspezifische Basisdienste

Ein Anwendungsspezifischer Basisdienst stellt einen dienstorientierten Zugriffspunkt auf eine Web-Anwendung dar. Dabei stehen vor allem Eigenschaften im Vordergrund, die es einem Basisdienst gestatten sich an der Anwendung zu orientieren. Dahinter steht die Einsicht, dass analog zur realen Welt, für bestimmte Aufgaben spezialisierte Werkzeuge zum Einsatz kommen. Ein Anwendungsspezifischer Basisdienst stellt eine abstrakte, aber wohldefinierte Vorschrift für die Generierung eines spezialisierten Werkzeuges dar.

Er umfasst dabei sowohl die konzeptionelle Ebene, die sich mit Fragen nach Brauchbarkeit und Beschaffenheit beschäftigt. Dazu zählen unter anderem die Identifikation der Anwendungsdomäne sowie entscheidende Eigenschaften dieser Domäne (siehe Abschnitt 5.2).

Er umfasst aber auch technische Fragestellungen, die Anforderungen nach einer effizienten und qualitativ hochwertigen Umsetzung Rechnung tragen müssen. Das Prinzip, das sich dahinter verbirgt wird auch als »fitness to purpose« bezeichnet (Budgen, 2003) und stellt ein Qualitätsmerkmal dar, das bezeichnet wie geeignet ein Werkzeug (Basisdienst) für den ihm gedachten Zweck (Anwendungsdomäne) ist (siehe Abschnitt 5.1.2).

Anwendungsspezifische Basisdienste beschreiben eine dedizierte Dienst- bzw. Prozessfunktionalität innerhalb einer Anwendungsdomäne. Sie bilden wieder verwendbare Grundeinheiten, die entsprechend den Anforderungen unterschiedlicher Szenarien angepasst werden können. Beispiele dafür sind der lokale oder zeitliche Kontext, benutzerdefinierte Einstellungen oder Sicherheit.

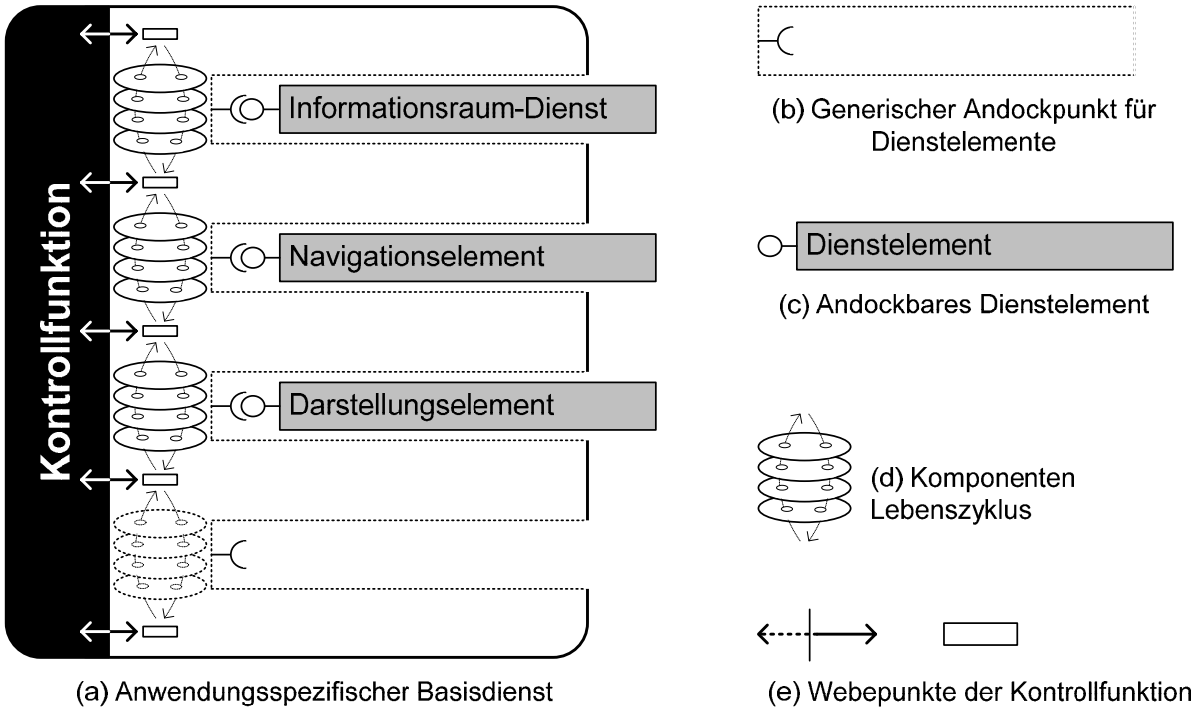


Abbildung 5-3: Entwurfsvorschrift eines Anwendungsspezifischen Basisdienstes wie er im WSLS-Ansatz verwendet wird.

Abbildung 5-3 stellt das zugrunde gelegte Klassifikationsschema eines Basisdienstes dar. Dabei werden insgesamt zwei wesentliche Elemente definiert:

- **Kontrollfunktionen** (engl. *control function*) haben die Aufgabe, einzelne Dienstelemente miteinander zu verbinden. Sie delegieren und organisieren den Datenfluss, die Integration und die Datenverarbeitung der beteiligten Dienstelemente. Eine Kontrollfunktion ist auch für die Umsetzung der Dienstfunktionalität eines potenziellen Geschäftsprozesses verantwortlich.
  In ihrer softwaretechnischen Ausprägung entspricht eine Kontrollfunktion dem Entwurfsmuster *Mediator* (Gamma, Helm, Johnson und Vlissides, 1995) und strebt an, eine möglichst *lose* Kopplung zu erzielen. Dadurch wird nicht nur die Möglichkeit zur Wiederverwendung von Dienstelementen gesteigert, sondern auch deren voneinander unabhängige Entwicklung gefördert. Dienstelemente können somit für Basisdienste hinzugefügt und entlang der den sich kontinuierlich ändernden Anforderungen angepasst werden.
- **Dienstelemente** (engl. *service elements*) werden über standardisierte Schnittstellen integriert (siehe Abbildung 5-3 (b) und (c)). Die hierbei fokussierten Standards umfassen insbesondere webbasierte Schnittstellenstandards, wie beispielsweise der Web Service Description Language (WSDL) (Chinnici, Gudgin, Moreau und Weerawarana, 2003) und Regeln zur Unterstützung der Interoperabilität, beispielsweise mittels der Standardisierungsvorgaben der Web Services Interoperability Organization (WS-I). Zusätzlich können auch softwaretechnische Komponenten als Dienstelemente vermittelt werden. Um die lose Kopplung zu ermöglichen bedarf es eines geordneten Durchlaufens unterschiedlicher fest vorgegebener Stationen innerhalb dieser Dienstelemente. Gleichzeitig stellen sie Methoden zur Verfügung, die es gestatten, auch solche Funktionalitäten zu beschreiben, die ein zugrunde liegendes Modell »quer

schneiden« (engl. »cross-cutt«). Diese Funktionalitäten werden in der Aspekt-orientierten Software Entwicklung (Filman, Elrad, Clarke und Aksit, 2004) auch als »Aspekt« bezeichnet. Ein Aspekt-orientierter Ansatz ist gut geeignet für den Einsatz in Anwendungen mit sich stark ändernden Anforderungen. Das liegt in der prinzipiellen Offenheit bezüglich des konkreten Komponentensystems begründet, welches ein nachträgliches Hinzuzufügen von Funktionalität ermöglicht.

Diese Konkretisierungen stellen eine Weiterentwicklung der in (Gaedke, 2000) vorgestellten Einflussgrößen, die auf eine Web-Anwendung wirken, dar. Hierbei werden Anwendungsspezifische Basisdienste als eine Menge von Dienstelementen und einer zwischen ihnen vermittelnden Kontrollfunktionen aufgefasst. Diese Dienstelemente erfüllen dabei unterschiedliche Aufgaben, die zur Erbringung der Basisdienstfunktionalität notwendig sind. Die Menge der beteiligten Dienstelemente ist abhängig von definierbaren Einflussgrößen und somit nicht abgeschlossen. Um eine systematische Konstruktion von Basisdiensten zu ermöglichen bedarf es einer Ordnung und Vorschrift, die eine nahtlose Verwebung dieser Dienstelemente mit existierenden Kontrollfunktionen ermöglicht.

Abbildung 5-3 stellt einen Anwendungsspezifischen Basisdienst (a) dar, der über verschiedene *Webepunkte* (e) verfügt. Mit Hilfe dieser Webepunkte können die spezialisierten Dienstelemente (c) an durch die Kontrollfunktion kontrollierten andockbaren Schnittstellen (b) angebunden werden.

Im Sinne des eingangs eingeführten komponentenbasierten Web Engineering (CBWE) werden Kontrollfunktionen und Dienstelemente im Folgenden als Komponenten aufgefasst. Diese Komponenten werden in WSLS als *WSLS-Elemente* bezeichnet und dienen als eine gemeinsame abstrakte Oberklasse für Kontrollfunktionen und Dienstelemente.

Im Folgenden werden ihre speziellen Eigenschaften und Charakteristika beschrieben, die eine gezielte wiederverwendungsorientierte Entwicklung gewährleisten (vgl. Abschnitt 5.4.1.1). Außerdem wird eine Evolutionsstrategie entwickelt, die es gestattet das allgemein anerkannte Objekt-orientierte Paradigma anzuwenden und dabei gleichzeitig dessen immanente Nachteile abzumildern (vgl. Abschnitt 5.5). Abschließend werden die erarbeiteten Konzepte exemplarisch am Beispiel eines WSLS-Elements – einem Informationsraum-Dienstelement – aufgezeigt (vgl. Abschnitt 5.6).

5.4.1 Lebenszyklus von WSLS Elementen

Dienstelemente und Kontrollfunktionen werden in der WSLS-Referenzarchitektur als Komponenten realisiert. Eine wichtige Eigenschaft von Komponenten bildet dabei die Abgeschlossenheit. Das bedeutet, dass sämtliche Interaktionen, die von einer Komponente initiiert werden, auch von dieser Komponente verarbeitet werden müssen. Es ist nicht ausgeschlossen und sogar wünschenswert, dass eine Komponente auch auf Interaktionssignale von anderen Komponenten reagieren kann. Das standardmäßige Verhalten muss einer Komponente allerdings immer gestatten, ihre eigenen Signale, auch identifizieren zu können. Dies ist vor allem dann wichtig, wenn eine Komponente an unterschiedlichen Stellen innerhalb einer Web-Anwendung wiederverwendet wird.

Jede Komponente in der WSLS-Referenzarchitektur wird durch die abstrakte Informationseinheit *WSLSElement* repräsentiert, die vom *ContentObject* abgeleitet ist. Dadurch lassen

sich Informationsraum-Dienste definieren, die den Zugriff auf diese Komponentenrepräsentationen organisieren. Die Evolutionsbetrachtungen aus Kapitel 4 können somit direkt auf die Verwaltung und Organisation der verwendeten Komponenten einer Web-Anwendung übertragen werden.

Damit die unterschiedlichen Dienstelemente mit verschiedenen Kontrollfunktionen kombiniert werden können, bedarf es eines geordneten und systematischen Ablaufs, wie die Komposition dieser Komponenten zur Laufzeit erreicht wird. Die verschiedenen Phasen sind in Abbildung 5-4 dargestellt. Der generische Andockpunkt eines Dienstelements besteht aus einer Menge von Phasen, die jeweils unterschiedliche Aufgaben haben. Eine Kontrollfunktion dirigiert die ihr zugeordneten Dienstelemente und ruft deren Phasen in einem geordneten Ablauf auf.

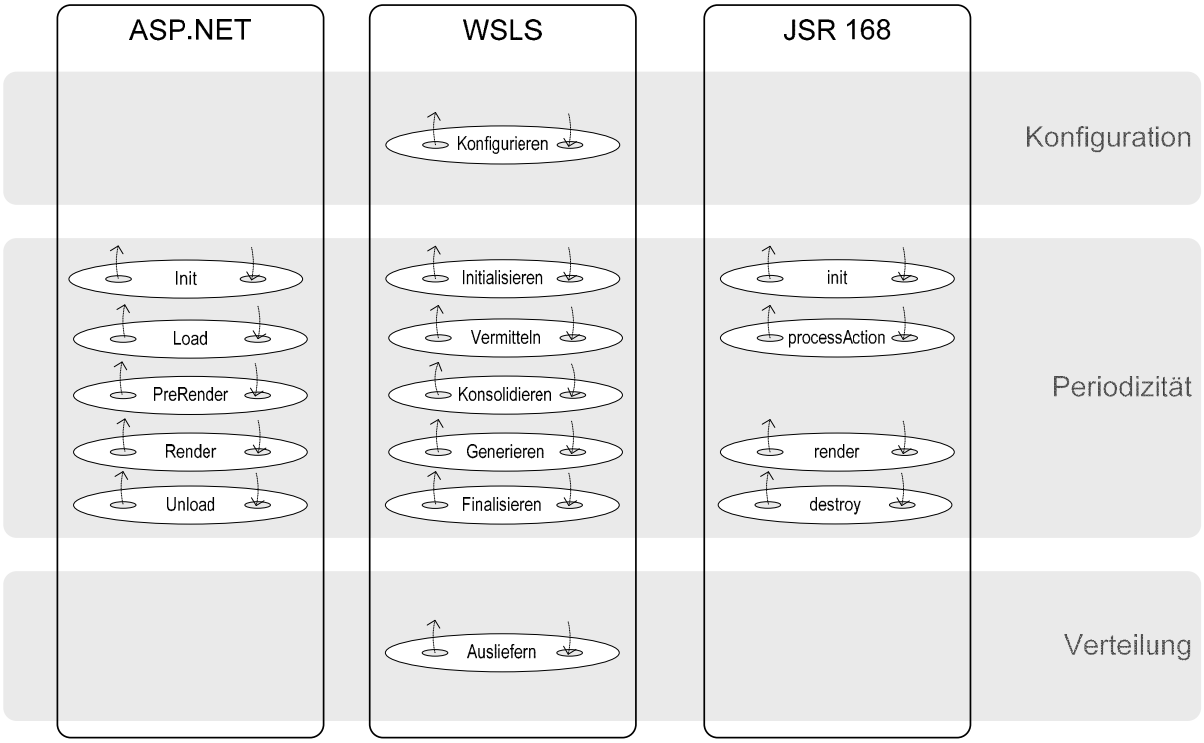


Abbildung 5-4: Die einzelnen Phasen im Lebenszyklus eines WSLS Elements ausgeprägt als Dienstelement, die durchlaufen werden.

Die Anzahl der Phasen ist dabei nicht abgeschlossen. So können für dedizierte WSLS-Elemente zusätzliche Phasen eingeführt werden. Dies erhöht die Flexibilität und Anpassbarkeit an unterschiedliche konkrete Realisierungsplattformen. Auf diese Weise kann beispielsweise der Lebenszyklus der ASP.NET Controls auf den WSLS Lebenszyklus übertragen werden. Genauso ließen sich JSR 168 basierte Portlets via WSRP durch dedizierte Dienstelemente in die WSLS-Referenzarchitektur integrieren. Abbildung 5-4 zeigt eine mögliche Zuordnung der Lebenszyklen von ASP.NET Controls sowie des JSR 168 Lebenszyklus. In beiden Fällen lassen sich die plattformspezifischen Phasen auf existierende WSLS-Phasen abbilden. Die übrigen, nicht abgebildeten Phasen, können mit Hilfe softwaretechnischer Maßnahmen wie etwa dem Einsatz des Entwurfsmusters Schablone (Gamma, Helm, Johnson und Vlissides, 1995) hinzufügt werden.

Die Phasen unterteilen sich in drei von einander unabhängige Bereiche. Diese Bereiche folgen nicht zwangsläufig sequenziell aufeinander, sondern bilden Einheiten zusammengehö-

render Phasen. In diesem Sinne bezeichnen sie Aktivitäten mit bestimmten zuordenbaren Aufgaben.

- **Konfiguration**: Das Konfigurieren von WSLS-Elementen stellt einen wichtigen Abschnitt im Lebenszyklus dar. Dieser Bereich wird immer dann aktiviert, wenn ein WSLS-Element konfiguriert werden muss. Dies kann durch Hinzufügen neuer Elemente (Standardkonfiguration), einen Kontextwechsel (beispielsweise Nutzer oder Lokation) oder manuell initiiert werden. Ein konkretes Beispiel für die Phase Konfiguration wird in Abschnitt 6.2.2 vorgestellt.
- **Periodizität**: Phasenabschnitte in diesem Bereich treten periodisch auf. Das bedeutet, für jede Verwendung eines WSLS-Elements werden die zugeordneten Phasen durchlaufen.
- **Verteilung**: Dieser Bereich beschreibt die Auslieferung von WSLS-Elementen. In (Gaedke, Meinecke und Nussbaumer, 2004b) wird detailliert ausgeführt, wie eine Verteilung von WSLS-Elementen unter Berücksichtigung von Sicherheit und Lizenzierungsmaßgaben erfolgen kann.

Die einzeln auftretenden Phasen des Bereichs Periodizität und ihre Aufgaben werden im Folgenden skizziert:

Phase Initialisieren

Die Initialisierphase übernimmt alle wesentlichen Voreinstellungen, die für ein Dienstelement wichtig sind. Die genauen Aufgaben innerhalb dieser Phase hängen von der jeweiligen Art des Elements ab. Im Falle eines Elements, das über Interaktion verfügt, müssen in dieser Phase die entsprechenden Oberflächenelemente (z.B. Verweise oder Formularelemente) mit den auszuführenden Aktionen verknüpft werden. Im Falle eines IR-Dienstelements werden in dieser Phase die Daten geladen.

Phase Vermitteln

In der Vermittlungsphase werden auftretende Interaktionen behandelt. Dabei treten unter Umständen Zustandsänderungen ein, die einen Ausgabeeffekt hervorrufen. Solche Effekte können beispielsweise durch benutzerdefinierte Interaktion mittels Dialogen herbeigeführt werden. Die Auswirkungen dieser Interaktionen können wiederum Auswirkungen auf andere Dienstelemente nach sich ziehen.

Phase Konsolidieren

Die Konsolidierungsphase dient der Verfestigung aller durch Interaktionen hervorgerufener Zustandsänderungen. In der Phase der Konsolidierung wird gewährleistet, dass sämtliche innerhalb des gegebenen Zyklus durchführbare Benutzerinteraktionen abgeschlossen sind. Die Konsolidierung schließt die plattformneutrale Konstruktion und Verwebung aller beteiligten Elemente ab. Nach Durchlaufen der Konsolidierungsphase befindet sich ein Basisdienst im Zustand der Auslieferbarkeit. Die Auslieferung an eine dedizierte Plattform wird dann in der abschließenden Generierung vorgenommen.

Phase Generieren

Die Generierungsphase nimmt einen konsolidierten Basisdienst und erzeugt den Programmcode, des zur Laufzeit benötigten Implementierungsmodells. Im Falle einer Web-Anwendung ist dies HTML oder (X)HTML konformer Markup. Ein Vorteil der Loslösung dieser Phase von der Konsolidierung bildet die Fähigkeit, plattformspezifischen Code zu generieren. So wäre es durchaus möglich verschiedene Ausgabegeräte wie PDAs (Personal Digital Assistent) oder Smartphones durch eine dedizierte Anpassung in dieser Phase zu erreichen. Im Falle eines Informationsraum-Dienstelements wird in dieser Phase eine SOAP kodierte Nachricht erzeugt.

Phase Finalisieren

Diese üblicherweise sehr plattformspezifische Phase erlaubt notwendige Aufräumarbeiten durchzuführen, wie beispielsweise das Beenden von Strömen auf ein lokales Dateisystem oder das explizite Schließen von Datenbankverbindungen.

5.4.1.1 Lebenszyklus am Beispiel des Informationsraum-Dienstelements

In der *Initialisierungsphase* wird ein Proxyobjekt konstruiert, das mit dem entsprechenden Informationsraum-Dienst kommunizieren kann. Hier kommt der Vorteil der kanonischen Schnittstellen zum Tragen, da für jeweils einen atomaren Diensttyp auch nur ein Proxy benötigt wird. Diese Proxy Objekte fungieren im Zusammenhang mit der horizontalen Evolution des Informationsraums als funktionales Gegenstück, um die Interaktion mit einem bestimmten Typus von IR-Dienst zu ermöglichen. Bestimmte wesentliche Parameter, wie der konkrete Endpunkt des Dienstes oder sicherheitsrelevante Eigenschaften lassen sich durch Konfiguration der entsprechenden Anwendungsdomäne erreichen.

In der Vermittlungsphase wird die *ServiceCard* des IR-Dienstes ausgelesen, um dienstabhängige Eigenschaften wie beispielsweise Cachezeiten, unterstützte Ausgabeschemas oder verwendete Übergabekontexte zu ermitteln.

Die *Konsolidierungsphase* verfestigt die ermittelten Werte und konfiguriert den in der Initialisierung erzeugten Dienstproxy. So kann auf Basis ermittelter Cachezeiten die Installation und Verwaltung eines vorgelagerten Caches im entsprechenden Proxy konfiguriert werden. Falls der Informationsraum-Dienst verschiedene Ausgabeschemas für die von ihm verwalteten Geschäftsobjekte verfügt, werden die entsprechenden Übergabekontexte erzeugt.

In der *Generierungsphase* wird ausgehend von den verfestigten Werten eine entsprechende Nachricht erzeugt. Im Falle eines SOAP-basierten Web Services kann dies mit Hilfe einer SOAP Nachricht erreicht werden. Im Falle eines REST-basierten Web Services würde diese Phase eine entsprechend parametrisierte URL erzeugen.

5.5 Komponentenevolution

Der kontinuierliche Wandel, dem Web-Anwendungen unterliegen, hat auch Auswirkungen auf die Komponenten, die einer Domäne zugeordnet sind. Durch die Entkopplung des Anwendungskonzepts (Domäne) von der erbringenden Funktion (fachliche Komponente) ist eine feingranulare Unterstützung der Evolution von Komponenten hin zur Maturität (Gaedke, 2000) möglich.

Eine Möglichkeit der sukzessiven Weiterentwicklung von Komponenten besteht darin, ihre Funktionen zu generalisieren. Diese Form der Weiterentwicklung ist sehr geeignet, eine Funktion, die durch eine Komponente erbracht wird, schrittweise weiter zu verfeinern. Durch Anwenden des Objekt-orientierten Paradigmas lassen sich so funktionale Spezialisierungen durch Wiederverwendung existierender Klassen beschreiben. Der Vorgang der schrittweisen Verfeinerung ist besonders gut geeignet, um eine zufriedenstellende Struktur der Klassen gemäß ihrer eigentlichen Funktionalität zu erlangen. Auf diese Weise wird die dominante Dekomposition der Aufgaben und den damit verbundenen Funktionen erreicht.

Dies ist einerseits wünschenswert, da so eine klare funktionale Hierarchie aufgebaut werden kann. Andererseits führt dies unweigerlich zum Problem der »Tyrannei der Dominanten Dekomposition« (Tarr, Ossher, Harrison und Sutton, 1999). Dahinter verbirgt sich die Unfähigkeit, das Prinzip des »Separation of Concerns« Paradigmas hinreichend auf *alle möglichen* Belange einer Klasse anzuwenden. Gerade Web-Anwendungen, die dem Einfluss der Evolution unterliegen, sind dieser Problematik im Besonderen ausgesetzt, da zukünftige Anforderungen in aktuellen Umsetzungen in der Regel nicht berücksichtigt werden können. In Web-Anwendungen tritt dieser Sachverhalt besonders stark auf, wenn die eingesetzten Modelle auf dem Objekt-orientierten Paradigma basieren (Cristina Cachero und Koch, 2002).

Die WSLS-Referenzarchitektur unterstützt die Komponentenevolution, indem sie die positiven Eigenschaften der Dominanten Dekomposition mit Ansätzen aus der Aspekt-orientierten Modellierung verbindet. Ein wichtiger Punkt, der dabei Berücksichtigung finden muss, ist die Frage nach der Abhängigkeit von Typen untereinander. Aufgrund ihrer Beschaffenheit gestatten es WSLS Elemente grundsätzlich miteinander verwoben zu werden; nicht jede Kombination ist allerdings auch sinnvoll. Das so entstehende Problem wird auch als Komponentendilemma bezeichnet, das das Paradoxon zwischen Anzahl der wiederverwendbaren Komponenten und dem Aufwand zum Finden der richtigen Komponente beschreibt (Henninger, 1994).

Die beiden Ansätze zur Komponentenevolution – Generalisierung und Aspekt-orientierte Verfahren innerhalb eines offenen Systems - werden in den nächsten Abschnitten vorgestellt.

5.5.1 Komponentenevolution durch Generalisierung

Aufgrund der zunehmenden Spezialisierung von Komponenten für eine bestimmte Aufgabe, nimmt der generische Nutzen ihrer ursprünglichen Aufgabe immer mehr ab. Wegen des Hinzufügens neuer spezialisierter Komponenten durch Generalisierung ihrer Typen, vererben sich auch deren Abhängigkeiten. In abgeschlossenen Typsystemen sind zum Entwicklungszeitpunkt alle Typen bekannt; in offenen Systemen ist dies nicht möglich und auch nicht wünschenswert. Da neue Typen in Form von Komponenten zu jeder hinzugefügt werden können, lassen sich Anwendungen je nach Art und Beschaffenheit ihrer Domäne gezielt entwickeln und anpassen. Allerdings ist es dann wichtig, dass die Komponenten, die zu einander passen, also von einander abhängen, auch entsprechend gefunden werden können.

Eine Milderung des Komponentendilemmas kann also durch das Vorhalten von Typabhängigkeiten in einem offenen System erreicht werden.

Eine mögliche Lösung dazu bilden die im Web entwickelten Ansätze zum »Tagging«. Dabei werden Schlüsselwörter – die so genannten *tags* – zu Entitäten gespeichert, die diese charakterisieren. Es existieren unterschiedliche Ansätze, wie das Anbringen der Tags umgesetzt wird. Zumeist kommen Menschen zum Einsatz, die diese Entitäten klassifizieren und ihre Interpretation in Form eines oder mehrerer Tags hinzufügen. Diese hauptsächlich nutzerzentrierten Verfahren werden überwiegend in Web 2.0-Anwendungen wie Flickr[52] oder Delicious[53] angewendet. Die Zahl der Tags ist dabei nicht eingeschränkt und wird auch nicht durch ein festes Vokabular vorgegeben.

Zusätzlich können solche Tags auch automatisch oder semi-automatisch angebracht werden, beispielsweise durch Analyse ihrer Verwendung in unterschiedlichen Kontexten oder durch automatische Programmcode-Analysen. Dadurch lassen sich zu einander passende Komponenten finden, die sich durch eine erfolgreiche Verwendung bewährt haben.

Die WSLS-Referenzarchitektur gibt vor, dass jedes WSLS-Element über einen universal eindeutigen Identifikator verfügt. Solche Identifikatoren können den in (Leach, Mealling und Salz, 2005) standardisierten UUIDs (Universally Unique Identifier) entsprechen. Dadurch kann gewährleistet werden, dass jede in WSLS eingesetzte fachliche Komponente weltweit eindeutig bestimmt werden kann. Daher eignen sich diese Identifikatoren ideal zur Auszeichnung von Typabhängigkeiten.

Initiale Tagmenge

Eine technische Umsetzung zum Tagging für Komponenten besteht in Form der deklarativen Auszeichnung, die bereits auf Ebene des Quelltextes vorgenommen wird. Im Falle von bereits zur Entwicklungszeit bekannten Typabhängigkeiten lassen sich diese direkt im Quelltext modellieren. Sprachen und Rahmenwerke wie *.NET* oder *Java ONE* erlauben fortschrittliche Introspektionsmöglichkeiten auf der Metaebene von Programmcode. So können Typabhän-

[52] http://www.flickr.com

[53] http://del.ioc.us

gigkeiten als Eigenschaften und Attribute deklarativ zu existierendem Code hinzugefügt werden. Diese können herangezogen werden, um eine initiale Startmenge von grundsätzlich gültigen Kompatibilitätsregeln auf bereits existierenden und zum Auszeichnungszeitpunkt bekannten Typen zu definieren.

Listing 5-1 zeigt wie eine solche Auszeichnung am Beispiel der .NET Plattform. Hier wird eine Menu-Komponente mit Hilfe eines benutzerdefinierten Attributs (Zeile 1, *UniqueIdentifier*) ausgezeichnet. Zusätzlich werden zwei weitere Deskriptoren verwendet *Convenient* (Zeile 2) und *Inconvenient* (Zeile 3), die eine bereits zur Entwicklungszeit bekannte Typabhängigkeit definieren. Im vorliegenden Fall können die eindeutigen Identifikatoren der übergebenen Typen der Navigations-Dienstelemente *MenuNavigator*, *HybridCollection* und *PreviousNext-Navigation* mit Hilfe von Introspektion herausgefunden werden.

```
01. [UniqueIdentifier("{761E5D50 − 5EC4 − 468d − B978 − ECC57E9F4B60}")]
02. [Convenient (typeof(MenuNavigator))]
03. [Inconvenient (new Type[] {typeof(HybridCollection), typeof(PreviousNextNavigation)} )]
04. public abstract class Menu : WSLS.CF.DomainCF
```

Listing 5-1: Deklarative Auszeichnung einer Kontrollfunktion in WSLS mit Hilfe eines standardisierten universal eindeutigen Identifikators.

Die hierbei verwendeten Tags umfassen:

- **UniqueIdentifier**: Ordnet einem WSLS-Element einen weltweit eindeutigen Identifikator zu. Dieser kann als UUID auf Basis des in RFC 4122 verabschiedeten Algorithmus konstruiert werden.
- **ConvenientType**: Beschreibt die Kompatibilitätsstufe *passend* zwischen der auszeichnenden (Menu) und ausgezeichneten (MenuNavigator) Komponente. Diese Information kann beispielsweise während der Konfiguration einer Menu-Komponente verwendet werden, um die Auswahl möglicher und passender Dienstelemente zu priorisieren.
- **InconvenientType**: Beschreibt die Kompatibilitätsstufe *unpassend* zwischen der auszeichnenden (Menu) und ausgezeichneten (HybridCollection) Komponente. Diese Information kann beispielsweise während der Konfiguration einer Menu-Komponente dazu verwendet werden, die Auswahl möglicher und passender Dienstelemente entsprechend zu reduzieren.

Erzeugen der Taghülle

Die initiale Tagmenge kann nur auf Basis des Quellcodes vorgenommen werden und eignet sich daher lediglich als ein erster aber wichtiger Schritt. Durch das Hinzufügen neuer Komponenten, die durch Anwenden des Objekt-orientierten Paradigmas bestehende Funktionalität erweitern, werden bestehende Typabhängigkeiten weitervererbt. Im Falle von neu entwickelten Komponenten verfügen bereits bestehende Komponenten offensichtlich noch über keine Typabhängigkeiten. In diesem Fall lassen sich Tags manuell zu einer Komponente hinzufügen, um zu einander passende Komponenten zu kennzeichnen. Neue Komponenten können auch durch Föderation hinzugewonnen werden. In diesem Fall werden die dedizierten Informationsraum-Dienste zur Manipulation einer WSLS Komponenten Registrierung

herangezogen, um Komponentenrepräsentationen auszutauschen. In diesem Fall ist es sehr wahrscheinlich, dass verschiedene Komponenten unabhängig von einander entwickelt wurden. Deren Kompatibilitätsstufen zueinander lassen sich dann durch manuelles Anbringen der entsprechenden Tags regulieren.

Die Tagging-Architektur am Beispiel

In der WSLS Suite existieren beispielsweise Komponenten, die eine Lösung des Navigationsmusters[54] »Set-based Navigation« anbieten. Durch Generalisierung können spezifischere Varianten dieses Musters erzeugt werden, wie etwa ein Vorwärts-Rückwärts-Element oder ein Element, welches das »Hybrid-Collection« Muster realisiert.

Umgekehrt lassen sich solch systematischen Navigationsmuster zumeist recht gut auf gleichartig strukturierten Content anwenden. Sollen aber beispielsweise bestimmte Domänenbaugruppen zusammengefasst und durch Navigationsregeln zugreifbar gemacht werden, ist dies oftmals ein willkürlicher Prozess, der sich an der Anwendungsfunktion ausrichtet. Typische Beispiele für solche Navigationsstrategien sind Landmarks oder Menüs. In (Nussbaumer, Freudenstein und Gaedke, 2006a) wird eine Navigationsstrategie entwickelt, die auf dem in Abschnitt 0 entwickelten DSL-Rahmenwerk basiert. Sie stellt Komponenten zur Verfügung, die mit einer DSL konfiguriert werden können, die die Navigation unter den beteiligten Domänenbaugruppen beschreibt.

Abbildung 5-5 stellt das Zusammenspiel zwischen der beteiligten Kontrollfunktion *Menu* und einer Menge von Navigations-Dienstelementen bei der Komponentenevolution dar. Durch die Föderierbarkeit von WSLS Elementen als *Globale Typen* bedarf es geeigneter Mechanismen zur Auszeichnung der Kompatibilität von Komponenten. In WSLS sind die Kompabilitätsstufen *ConvenientType* (C) und *InconvenientType*(IC)[55] dafür vorgesehen. Die Auszeichnungsebene (E_2) verbindet die konkrete Komponentenebene (E_1) mit der Menge an verfügbaren Komponenten (E_3) des Informationsraums.

[54] Eine Auswahl existierender Hypermedia Design Pattern, wozu auch das Set-based Muster gehört, wird von wird von der ACM-SIGWEB zur Verfügung gestellt und kann unter http://www.designpattern.lu.unisi.ch eingesehen werden.

[55] Convenient (passend, geeignet) und Inconvenient (unpassend, ungeeignet) beziehen sich lediglich darauf auszuzeichnen, wie gut die entsprechend Komponenten zu einander passen oder nicht. Es ist zwar denkbar, dass Komponenten exklusiv für einander entwickelt werden, aber nicht der Standardfall.

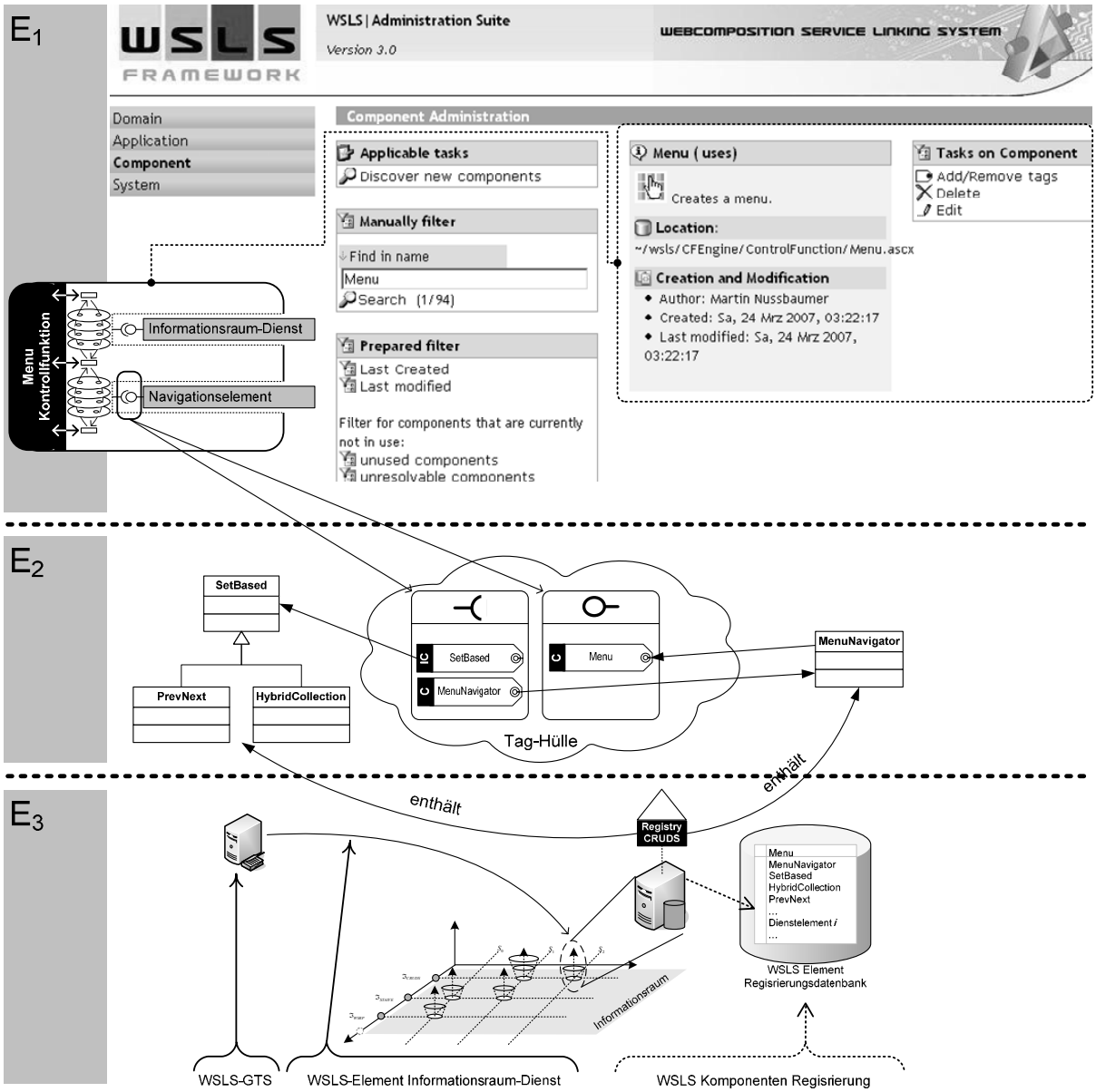


Abbildung 5-5: Zusammenspiel zwischen Kontrollfunktionen und Dienstelementen bei der Komponentenevolution am Beispiel von Navigations-Dienstelementen. Durch die Föderierbarkeit von Globalen Typen bedarf es geeigneter Mechanismen zur Auszeichnung der Kompatibilität von Komponenten.

Mit der Auszeichnung *C* lassen sich passende Komponenten nach ihrer Wichtigkeit in Bezug zu einer Komponente ordnen. Eine mögliche Art der Visualisierung sind sogenannte Begriffswolken (engl. *tag cloud*), die sich im Web 2.0 etabliert haben. Das Tag C(Menu), das durch das Navigations-Dienstelement *MenuNavigator* definiert wird, besagt, dass die Menu-Kontrollfunktion mit dem Navigations-Dienstelement *MenuNavigator* kompatibel ist.

Mit der Auszeichnung *IC* lassen sich inkompatible, also nicht zueinander passende Komponenten, auszeichnen. Dadurch lässt sich der mögliche Raum an verfügbaren Elementen einschränken. Das Tag IC(Set-based), das durch die Kontrollfunktion *Menu* definiert wird, sagt aus, dass sie inkompatibel zum Navigations-Dienstelement *SetBased* ist. Zusätzlich lässt sich folgern, dass sämtliche von *SetBased* abgeleiteten Navigationselemente inkompatibel zu dieser Kontrollfunktion sind. Diese Eigenschaft überträgt sich durch die Generalisierung von Set-based auf PrevNext und HybridCollection.

Das Hinzufügen zusätzlicher Tags, die eine initiale Menge erweitern, können als administrative Aufgabe verstanden werden. Diese Aufgabe wird dann an den konkreten WSLS-Elementen vorgenommen. Solche administrativen Aufgaben, die eine Komponente betreffen können in der Administrativen Ebene (E_1) durchgeführt werden. Im Abschnitt 6.2 der Anwendungsmodellierung wird die Tagging-Architektur eingesetzt, um die Schnittstelle zur Konfiguration von Web-Anwendung zur Evolutionszeit zu realisieren.

5.5.2 Komponentenevolution mit Aspekt-orientierten Modelleigenschaften

Die Aspekt-orientierte Programmierung (AOP) tritt als ein neues Programmierparadigma an, welches zum Ziel hat, »Separation of Concerns« nicht nur auf konzeptioneller Ebene, sondern auch auf Implementationsniveau durchzusetzen. Sie ist, ebenso wie die strukturierte, logische, funktionale oder Objekt-orientierte Programmierung, ein für sich eigenständiges Programmierparadigma. Jedes dieser Paradigmen bietet dabei seinen Anwendern bestimmte Möglichkeiten, Anwendungsdomänen in Module zu dekomponieren.

Die Dekomposition komplexer Systeme stößt bei den vier Letztgenannten regelmäßig auf das Problem, dass eine eindeutige Modularisierung bestimmter Belange des Systems *nicht* möglich ist. Die Idee zur Aspekt-orientierten Programmierung wird erstmals von Kiczales in (Kiczales, Lamping, Mendhekar, Maeda et al., 1997) beschrieben. Dabei werden Code-Fragmente identifiziert, die nicht zu der auf die Kernfunktionalitäten ausgerichtete Zerlegung des Programmes in Code-Module oder Klassen passen. Diese Funktionalitäten werden Aspekte genannt. Ihre wesentliche Eigenschaft ist, dass ihre Code-Fragemente sich durch die Implementierung der Kernfunktionalitäten eines Programms ziehen und es »querschneiden« (cross-cutting). Daraus resultiert eine Verteilung des zu einem Aspekt gehörenden Codes über die Code-Komponenten des Programms, die zudem untereinander verbunden sein müssen, um den Aspekt-Code korrekt ausführen zu können. Dieser Sachverhalt wird auch als »scattered« (engl. verstreut) und »tangled« (engl. verworren) code bezeichnet. Dahinter verbirgt sich die beobachtbare funktionale Defragmentierung, die einen ursprünglich – im Sinne einer dominanten Dekomposition – sauberen Entwurf nach und nach durch die sich ergebenden Änderungen aufweicht und zerstört. Eine Anwendung Aspekt-orientierter Vorgehensweisen verspricht jedoch gerade diesen Problemen entgegenzutreten und durch die Integration in den Entwicklungsprozess die Wiederverwendung der Aspekte zu steigern (Elrad, Aldawud und Bader, 2002). Außerdem ermöglicht sie die Erhöhung der Konsistenz zwischen den Anforderungen dem realisierten Produkt (Clarke, 2002).

5.5.2.1 Der Lösungsansatz durch Aspekt-orientierte Modellierung

Ein ***Aspekt*** beschreibt eine Funktionalität, die nicht zur Kernfunktionalität eines Programmes gehört und sich daher orthogonal zur funktionalen Zerlegung des Programmes in Komponenten (functional decomposition) über diese Komponenten verteilt und sie damit quasi »durchschneidet«. Daher werden Aspekte auch als »crosscutting concerns« bezeichnet; im Kontrast zu den Kernbelangen des Basis-Codes, den »core concerns«. Im Gegensatz zur Objekt-orientierten Modellierung beschreiben sie »Eigenschaften eines Systems, die eine Auswirkung auf dessen Ausführung haben, die sich nicht notwendigerweise an den funktionalen Modulen (dieses Systems) ausrichtet« (Lopes, 1997).

Joinpoints bezeichnen die Verknüpfungspunkte im Basis-Code, an denen es zur Ausführung von Aspekt-Code kommt. Sie können beliebige Punkte im Kontrollfluss eines Programms sein. Mögliche Verknüpfungspunkte stellen beispielsweise Methodenaufrufe oder Zugriffe auf Variablen dar. ***Pointcut Designatoren*** beschreiben eine Menge von Verknüpfungspunkten, auf die eine Eigenschaft angewendet werden soll. Pointcuts werden dabei in einer speziellen Aspekt-Sprache formuliert. So können beispielsweise alle Methodenaufrufe – in Form einzelner Verknüpfungspunkte – von Methoden mit einer bestimmten Signatur erfasst werden.

Ein ***Advice*** stellt schließlich ein Code-Fragment eines dedizierten Aspekts dar, die funktionale Eigenschaft, die bei Erreichen eines Verknüpfungspunkts ausgeführt werden soll. Ihre zusätzlich zu erbringende Funktion wird dann mit Hilfe eines Verwebers (engl. *weaver*) zum ausgeführten Programm hinzugewoben. Advices werden meist in der Aspekt-Sprache formuliert, enthalten aber Programm-Code in derselben Sprache, in der das Basisprogramm realisiert wurde. Das Zusammenfügen dieser Funktionalitäten wird als *Verwebung* bezeichnet.

Trotz seiner verhältnismäßig langen Bestehenszeit seit Anfang der 90er Jahre steckt das AOP Konzept hinsichtlich seiner Umsetzung und industriellen Anwendung noch weitestgehend in den Kinderschuhen. Die Gründe hierfür sind vor allem auf das Fehlen von Designrichtlinien und Modellen, die eine systematische Anwendung der Konzepte ermöglichen, zurückzuführen. Außerdem behindert die Nichtfokussierung auf konkrete Anwendungsdomänen, wie die Entwicklung von Web-Anwendungen, die Erstellung solcher Modelle, da dadurch ein zu allgemeiner und generischer Bezug hergestellt wird. Eine Anwendung von Konzepten aus der AOP kombiniert mit der Anwendungsdomäne des Web Engineering stellt daher einen viel versprechenden synergetischen Ansatz dar.

5.5.2.2 Wohldefinierte Verknüpfungspunkte in der WSLS-Referenzarchitektur

Zur systematischen Unterstützung der Evolution durch Aspekte wird das Konzept der wohldefinierten Verknüpfungspunkte eingeführt. Sie werden mit dem Komponenten Lebenszyklus (siehe Abschnitt 5.4) verwoben, so dass das Etablieren beliebiger zusätzlicher funktionaler Eigenschaften entlang dieser Punkte ermöglicht wird. Die Unterstützung wohldefinierter Verknüpfungspunkte ist zudem gerade in Verbindung mit heterogenen verteilten Systemen und den per se vordefinierten Kommunikationspunkten wie Request und Response im Falle einer Client/Server Architektur äußerst geeignet. Zusätzlich wird zugunsten der Systematik die Flexibilität des Programmiermodells eingeschränkt, so dass das Anwendungsmodell für die Verknüpfungspunkte maßgeblich ist (Janus, 2006).

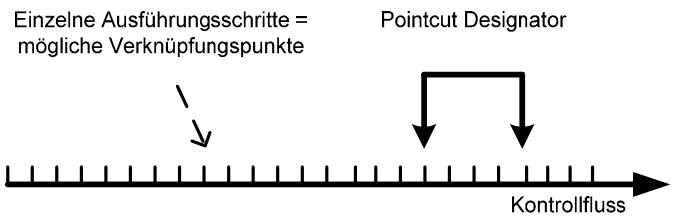


Abbildung 5-6: Frei wählbare Verknüpfungspunkte.

In der Abbildung 5-6 ist ein Pointcut Designator einer beliebigen AOP Unterstützung dargestellt. Veranschaulichend gesprochen kann der Pointcut im flexibelsten Fall beliebig viele

Verwebungsstellen enthalten und die Pointcuts – repräsentiert durch Pfeile – können auf beliebige Ausführungsschritte gelegt werden. Die Ausführungsschritte sind die atomaren Operationen des unterstützenden Verknüpfungspunkt-Modells.

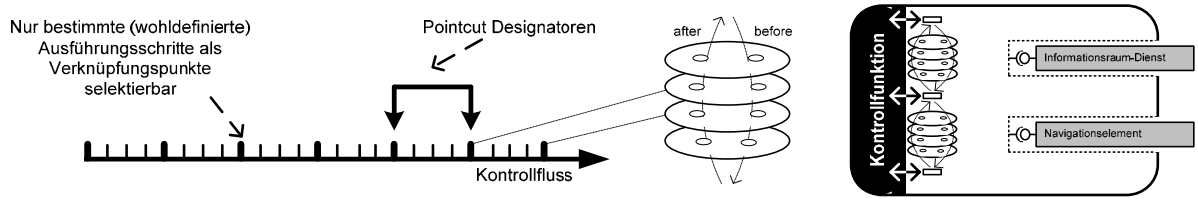


Abbildung 5-7: Wohldefinierte Verknüpfungspunkte, die mit dem Lebenszyklus von WSLS-Elementen assoziiert werden.

Abbildung 5-7 stellt das Modell der wohldefinierten Verknüpfungspunkte dar. Danach werden nicht beliebige Verknüpfungspunkte gestattet, sondern lediglich jene, die a priori bekannt sind. Die Abbildung stellt die Assoziation der Verknüpfungspunkte mit dem Lebenszyklus der WSLS Elemente dar. Pointcut Designatoren können zur Quantifizierung auf die Metadaten dieser Objekte zugreifen, also beispielsweise alle Elemente eines Typs oder mit einem bestimmten Merkmal versehen. Zusätzlich lassen sich Kontrollfunktionen zusammenfassen, die mit bestimmten Dienstelementen ausgestattet sind. Auf diese Weise könnten Aspekte allen Kontrollfunktionen zugeordnet werden, die mit einem Informationsraum-Dienstelement eines bestimmten Typs verbunden sind.

Verwebungs-Unterstützung

Ein Verweber (engl. *weaver*) verbindet den Aspekt-Code mit dem Basis-Code zu einem einzigen Programmablauf. Er stellt einen eigenen Teil eines AOP Softwaresystems dar, der entweder zur Übersetzungszeit (engl. *compile-time weaving*) oder erst zur Laufzeit (engl. *runtime weaving*) eingesetzt wird. Seine wesentliche Aufgabe besteht darin, dass die zur Entwicklungszeit getrennten Komponenten zur Laufzeit zu einem Kontrollfluss zusammengefasst werden (Kiczales et al., 1997)

Bei beiden Weaving-Varianten müssen die wohldefinierten Verknüpfungspunkte Zugriff auf Aspekt bzw. Advice-Komponenten bieten. *Compile-time weaving* hat hier den Vorteil, dass die AOP-Unterstützung außerhalb der eigentlichen Web-Anwendung umgesetzt werden kann, während *runtime weaving* einen Eingriff bis hin zum Refactoring der Web-Anwendung erfordert. *Compile-time weaving* ist hingegen wenig flexibel im Vergleich zum *runtime weaving*, da bei Änderungen die gesamte Web-Anwendung neu kompiliert werden muss, wobei das *runtime weaving* ein Hinzufügen von Aspekten durch Konfiguration, also auch bei einer laufenden Anwendung erlaubt.

Das WSLS-Element *Kontrollfunktion* bietet sich aufgrund der Funktion als Mediator zwischen einzelnen Dienstelementen ideal an, um die Verwebung zu implementieren. So können beispielsweise mit Hilfe des Entwurfsmusters Schablone (Gamma, Helm, Johnson und Vlissides, 1995) die Einwebestrategien der *advices* umgesetzt werden.

Die wohldefinierten Verknüpfungspunkte reduzieren zwar die Flexibilität der AOP Implementierungstechnik, ermöglichen aber eine systematische Modellierung in Richtung der Modell-

gesteuerten Software-Entwicklung. Durch die Reduktion der Zahl der Verknüpfungspunkte auf eine vordefinierte Menge, lassen sich Aspekte auch leichter wieder verwenden.

In (Janus, 2006) wird eine Sprachunterstützung basierend auf XML entworfen, die eine Formulierung von Aspekten auf Basis der wohldefinierten Verknüpfungspunkte für Web-Anwendungen beschreibt. Zusätzlich gestattet sie eine Quantifizierung dieser Aspekte entlang der in WSLS relevanten Größen fachliche Komponente, konzeptuelle Komponente und der Web-Anwendung selbst. Dadurch können Aspekte auf beliebige Teile oder sogar komplette Web-Anwendungen definiert werden. Auf diese Weise werden bestehende Komponenten wie die Entwickler selbst mit einem hohen Maß an »Vergesslichkeit« (engl. *obliviousness*) ausgestattet, die ein Black-box Verhältnis zwischen Aspekt und zu verwebendender Komponente erlaubt.

Nach Filman und Friedman gilt: »Better AOP systems are more oblivious. […] “Just program like you always do, and we’ll be able to add the aspects later.” (Filman und Friedman, 2005).

Diese Unterstützung wurde in der WSLS Suite realisiert und deren Effektivität anhand eines Beispiels demonstriert. Im Anhang findet sich die komplette Sprachbeschreibung in Form eines XML Schemas wieder.

5.5.2.3 Schema Validierung als Aspekt

Ein Beispiel für einen Aspekt stellt das folgende Szenario dar: Ein Informationsraum-Dienst vom Typ CRUDS wird verwendet, um web-konforme Texte zu verwalten. Diese Texte lassen sich von Redakteuren einer Web-Anwendung online editieren, um deren Inhalte zu aktualisieren. Die Texte dürfen zu einem Schema konforme Auszeichnungselemente enthalten, um beispielsweise Formatangaben oder Strukturierungselemente zu integrieren.

Im Folgenden wird demonstriert, wie das gewünschte Verhalten mit Hilfe eines Aspekts realisiert werden kann. Abbildung 5-8 stellt das Verhalten und die Wirkungsweise des *ValidateAspekt* in Form eines Sequenzdiagramms dar.

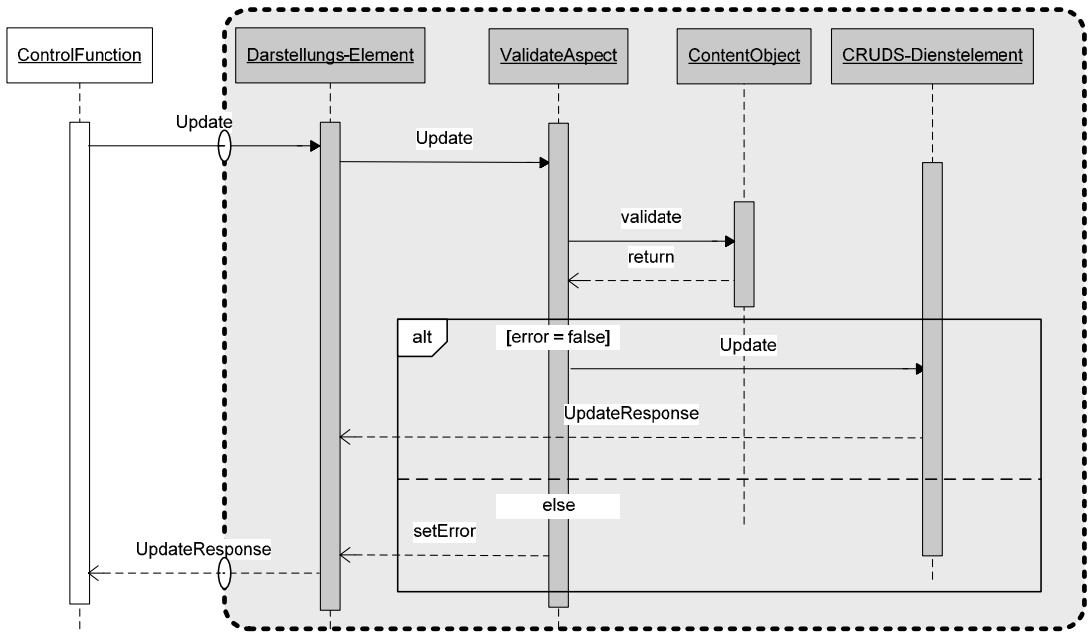


Abbildung 5-8: Verhalten und Kontrollfluss des ValidateAspekts als UML Sequenzdiagramm dargestellt.

Abbildung 5-9 stellt den Vorgang nach Eingabe eines Textes und anschließendem Aktivieren des Update Knopfes dar. Aufgrund des falsch konstruierten Markup Textes wird eine entsprechende Fehlermeldung angegeben, die hilft, den Fehler zu beheben. Die Überprüfung der Validität wurde mit Hilfe eines Aspekts realisiert.

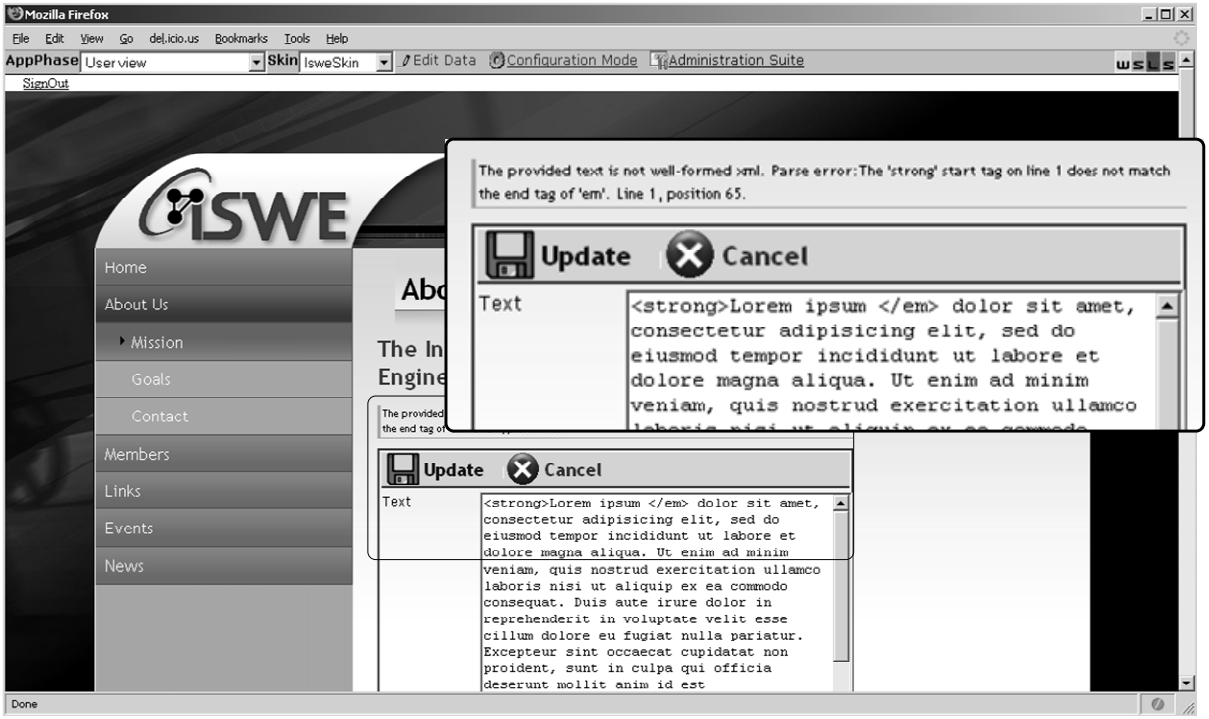


Abbildung 5-9: Szenario „online editieren“ mit einer Schema Überprüfung

Komponentenmaturität durch Aspekte

Durch Aspekt-orientierte Ansätze kann auch eine funktionale Maturierung einer Anwendungsdomäne in ihrer Ausprägung aus fachlichen Komponenten stattfinden. Das wird durch das wohldefinierte Verknüpfungspunkt-Modell erreicht. Aspekte können mittels des in den Kontrollfunktionen realisierten Verwebers zur Laufzeit zum Komponenten Code »hinzugewoben« werden.

Dies führt zunächst zu einer *transienten Maturierung*. Aufgrund der losen Bindung der hinzugewobenen Komponenten lässt sich das entstandene neue Verhalten jederzeit durch Hinzufügen oder Austauschen von Komponenten bzw. den beteiligten Aspekten ändern. Soll das entstandene neue Verhalten allerdings konserviert werden, um es wie im Falle der Maturität einer Komponente in WebComposition (Gaedke, 2000) als eigenständiges Verhalten anzubieten, bedarf es zusätzlicher Anforderungen an die Komponente bzw. der Verfügbarkeit ihres Quellcodes. Streng genommen wird dadurch die durch die Aspekt-orientierte Herangehensweise hinzugewonnene Flexibilität hinsichtlich der dominanten Dekomposition eingeschränkt. Allerdings wird so eine Erweiterung und Flexibilisierung der Evolution von Web-Anwendungen erreicht: Durch Aspekte maturierte Komponenten können wiederum als Ausgangpunkt für zukünftige Entwicklungen verwendet werden.

Im Zuge dieser Konservierung lassen sich zwei Maturitätsvarianten unterscheiden: »Komponente mit transparenter Verfestigung« oder »Komponente mit semi-transparenten Verfestigung«.

Im Fall der Variante »Komponente mit semi-transparenten Verfestigung« wird basierend auf der AML Spezifikation entlang der wohldefinierten Verknüpfungspunkte die existierende Komponente generalisiert. Die Verwebung findet mit Hilfe von gängigen Konzepten aus der Objektorientierung, nämlich Generalisierung und Methodenüberschreibung, statt. Dadurch entsteht eine Komponente, die nicht den kompletten Quellcode des realisierten funktionalen Verhaltens umfasst, sondern lediglich den Teil, der durch die Aspekt Spezifikation zur Verfügung gestellt wurde.

Die Variante »Komponente mit transparenter Verfestigung« ähnelt der Neuentwicklung einer Komponente. Ihr bestehender Quellcode wird mit Hilfe der AML Spezifikation an den entsprechenden wohldefinierten Verknüpfungspunkten verwoben. Dadurch entsteht eine neue Komponente, die um das Verhalten des Aspekts erweitert wurde. Der Vorteil dieser Variante besteht in der Vollständigkeit des zur Komponente zugehörigen Quellcodes, wodurch die Übersetzung auf verschiedene Plattformen und Rechnerarchitekturen gewahrt bleibt[56]. Sie wird ab dem Verwebungszeitpunkt als eigenständige neue Komponente betrachtet.

Die Verfestigung stellt somit eine Funktion der unterschiedlichen Ausprägungen und Konfigurationen des Verhaltens einer Komponente mit Aspekten über verschiedene Konfigurationszeitpunkte dar.

5.6 Beispiel eines Informationsraum-Dienst-Elements

Die informationstragenden Inhalte bilden die grundlegende Verarbeitungseinheit eines Basisdienstes. Die in Kapitel 4 entwickelte Systematik erlaubt den evolutionsorientierten Entwurf von Diensten zum segmentierten Zugriff auf den Informationsraum. Am Beispiel der kanonischen Dienstschnittstelle CRUDS (siehe Abschnitt 4.2.4) wurde aufgezeigt wie ein Dienst zur Manipulation, Verwaltung und Suche von Informationen entworfen wird.

Der enorme Vorteil der einfachen Wiederverwendbarkeit solcher kanonischer Schnittstellen lässt sich auf die Ebene der Dienstelemente übertragen. So bedarf es lediglich eines dedizierten Dienstelements für je eine kanonische Schnittstelle, um beliebige Informationsraum-Dienste dieses Schnittstellentyps einzubinden.

[56] Hierbei bleibt die Einschränkung auf das Komponentensystem, in dem die Komponente realisiert wurde, nach wie vor erhalten.

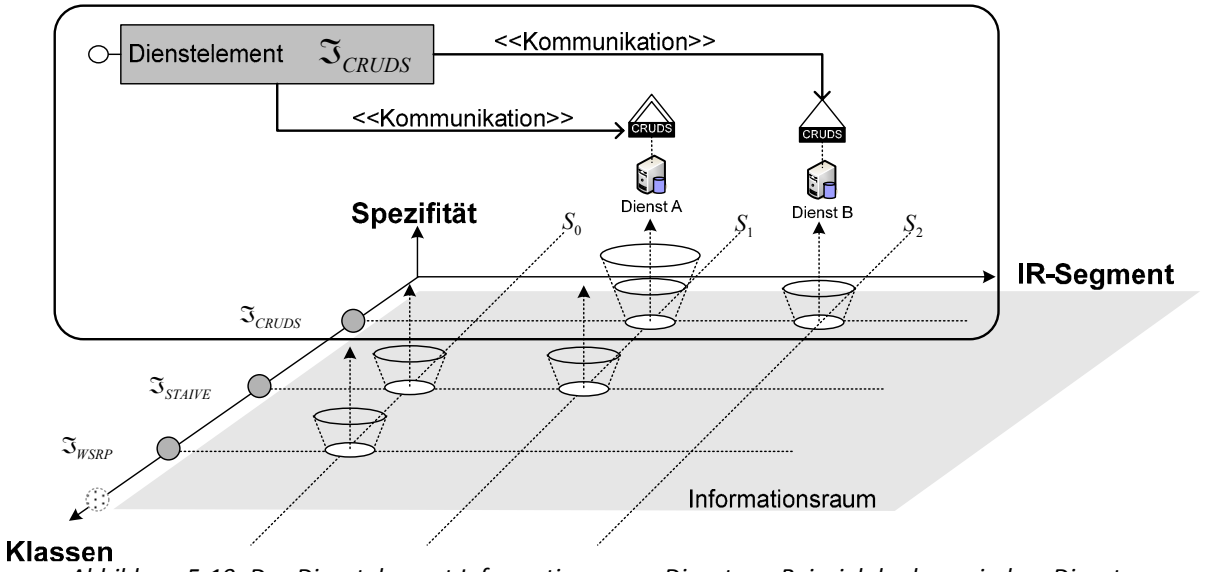


Abbildung 5-10: Das Dienstelement Informationsraum-Dienst am Beispiel der kanonischen Dienstschnittstelle CRUDS.

Abbildung 5-10 erläutert, wie sich der durch die Einführung kanonischer Dienstschnittstellen gewonnene Wiederverwendungseffekt auf die Ebene der Dienstelemente durchschlägt. Aufgrund der systematischen Durchführung der vertikalen Evolution bleibt diese Eigenschaft auch bei spezifischeren Informationsraum-Diensten weiterhin erhalten (in der Abbildung Dienst A). Selbstverständlich können durch spezifische Weiterentwicklungen des IR-Dienstelements auch spezifischere Funktionen – im Sinne der vertikalen Evolution von Informationsraum-Diensten – verwendet werden. Die Grundfunktionalität *aller* entsprechend typisierten Informationsraum-Dienste lässt sich allerdings jederzeit mit *einem* Dienstelement abrufen (in der Abbildung werden Dienst A und Dienst B vom gleichen Dienstelement genutzt). Das ist gerade hinsichtlich kurzer Entwicklungszyklen vorteilhaft, da so spezifischere Funktionalität in verschiedenen Iterationen auf einem bereits voll lauffähigen Prototypen hinzugefügt werden kann. Auf diese Weise lassen sich beliebige Geschäftsprozesse in WSLS integrieren.

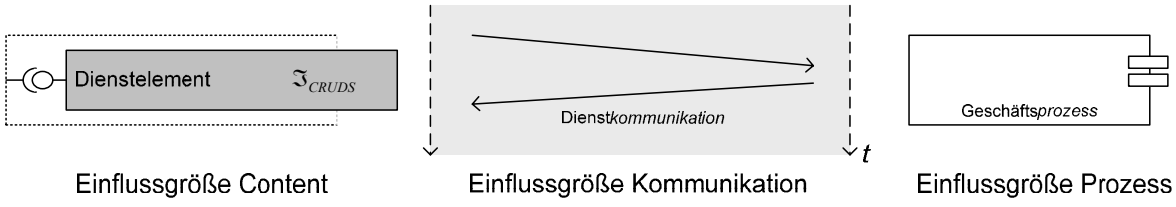


Abbildung 5-11: Zusammenspiel der Einflussgrößen Inhalt, Kommunikation und Prozess im Informationsraum-Dienstelement.

Abbildung 5-11 stellt den Zusammenhang zwischen Einflussgrößen Content, Kommunikation und Prozess auf. Diese werden durch den Informationsraum-Dienst erbracht und unterliegen der Kontrolle des jeweiligen Dienstanbieters.

Der **Content** des IR-Dienstelements basiert auf einer Konkretisierung des ContentObject innerhalb des Globalen Typ Systems. Der Informationsraum-Entwurf setzt die Definition eines

ContentObject für ein Globales Typ System voraus. Die WSLS-Referenzarchitektur definiert ein solches Basisobjekt, das konform zu den in 4.2 eingeführten Regeln des Globalen Typ Systems ist.

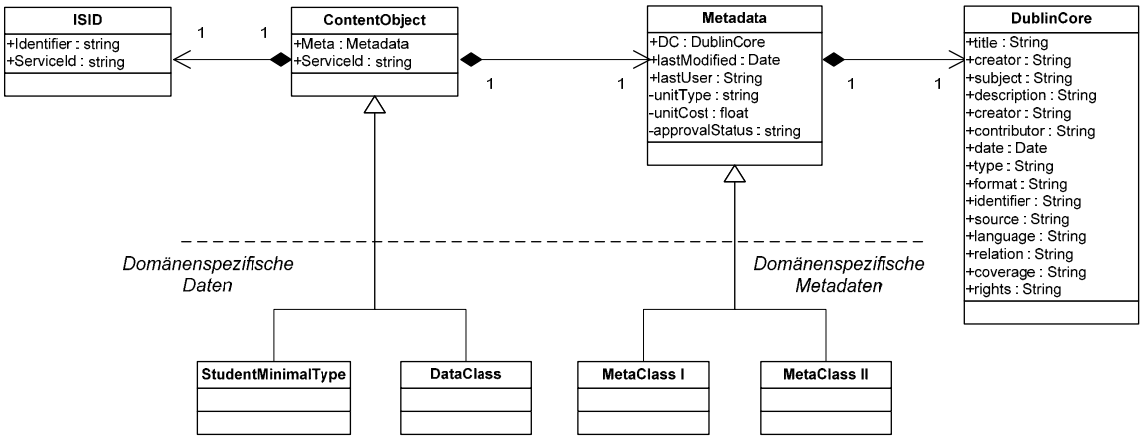


Abbildung 5-12: Grundlegende Definition des ContentObject im Globalen Typ System der WSLS-Referenzarchitektur.

Abbildung 5-12 stellt diese Definition in Form eines UML Klassendiagramms schematisch dar. Das ContentObject bildet dabei konform zu den Voraussetzungen des Globalen Typ Systems einen Zusammenschluss aus Informationsraum Identifikator (ISID) und zugehöriger Metadaten. Die WSLS-Referenzarchitektur verwendet dazu die standardisierten Dublin Core Metadaten (Andresen, 2003). Das ContentObject, wie es in WSLS verankert wurde, bildet den Ausgangspunkt für die Modellierung domänenspezifischer Erweiterungen. Das gilt einerseits für die Anwendungsdaten, die sich beispielsweise an Geschäftsobjekten ausrichten können. Eine solche Erweiterung von Geschäftsobjekten wurde in Abschnitt 4.2.4 im Zuge der CRUDS Schnittstelle vorgestellt. Andererseits lassen sich auch Erweiterungen in der Dimension der Metadaten realisieren. Dadurch können dediziert zusätzliche Beschreibungen und Klassifikationen des Contents vorgenommen werden, die entsprechend dem zu Grunde liegenden Anwendungsbereich angepasst werden können. In (Razumna, 2003) wird dies am LOM Metadaten Standard (Hodgins und Duval, 2005) vorgeführt. Auf diese Weise können sowohl Metadata als auch Anwendungsdaten abhängig vom Anwendungskontext dediziert angepasst werden, ohne die gemeinsame Basis – das ContentObject – zu verändern.

Die Einflussgröße **Kommunikation** bezeichnet grundlegende Kommunikationsprotokolle, die einem Basisdienst zur Verfügung stehen. Um ein nahtloses Einfügen eines Dienstes in unterschiedliche Umgebungen zu ermöglichen, muss die Kommunikation mit den heterogenen Systemen besondere Berücksichtigung finden. So lassen sich neben proprietären Kommunikationsprotokollen wie dem lokalen Dateisystem insbesondere auch standardisierte Protokolle wie etwa HTTP, SOAP oder WebDAV (Web Distributed Authoring and Versioning, W3C Empfehlung) (Goland, Whitehead, Faizi, Carter et al., 1999) unterstützen. In seiner Ausprägung als XML Web Service verwendet ein Informationsraum-Dienst immer das SOAP Protokoll. Daneben sind auch Szenarien denkbar, in denen REST-basierte Web Services eingesetzt werden. Gerade durch die stetige Zunahme der im Zuge des Web 2.0 Phänomens öffentlich verfügbaren Informationsquellen, erhalten leichtgewichtigere Protokolle oftmals den Vorzug vor dem aufwändigeren SOAP Pendant.

Die Einflussgröße **Prozess** beschreibt die anschließende Weiterverarbeitung der Daten eines Basisdienstes. So sind hier Anbindungen an externe, weiterverarbeitende Workflows zur Integration in weitere Geschäftsprozesse denkbar. Informationsraum-Dienste ausgeprägt als Web Services bieten sich als sichere, plattformübergreifende Technologie für eine Anbindung externer Prozesse an. Die unterschiedlichen Anbindungsszenarien durch Informationsraum-Dienste wurden in Abschnitt 4.3 vorgestellt. In diesem Zusammenhang ist vor allen Dingen die Anbindung von Alt-Systemen (Legacy Systeme) über eine Web Service Schnittstelle als Verarbeitungselement innerhalb eines Basisdienstes interessant. Dadurch kann ein Investitionsschutz für bereits existierende Systeme unterstützt werden. Durch die Verwendung offener bzw. standardisierter Schnittstellen stellen diese aber auch gleichzeitig eine Investion in die Zukunft dar. So kann zukünftig die Systemfunktionalität noch einfacher in Basisdienste integriert werden, so dass sich hier weitere Sparpotenziale eröffnen.

Zusammenspiel der Prozess- und Kommunikationsschnittstelle am Beispiel CRUDS

Eine Trennung der Informationsraum-Dienste hinsichtlich der Schnittstelle und des Verarbeitungsprozesses eröffnet dabei neben der Möglichkeit von kanonischen Schnittstellen noch weitere Vorteile. Die Wiederverwendbarkeit der Prozesskomponenten als solche kann dadurch erhöht werden, dass sie ebenso wie die kanonischen Schnittstellen auch als kanonische Prozesskomponenten verfügbar gemacht werden. Durch Vererbung können diese Prozesse dann sukzessive spezifischer werden. Die Prozessebene stellt einen universalen Komponenten-basierten Zugang zur Verfügung.

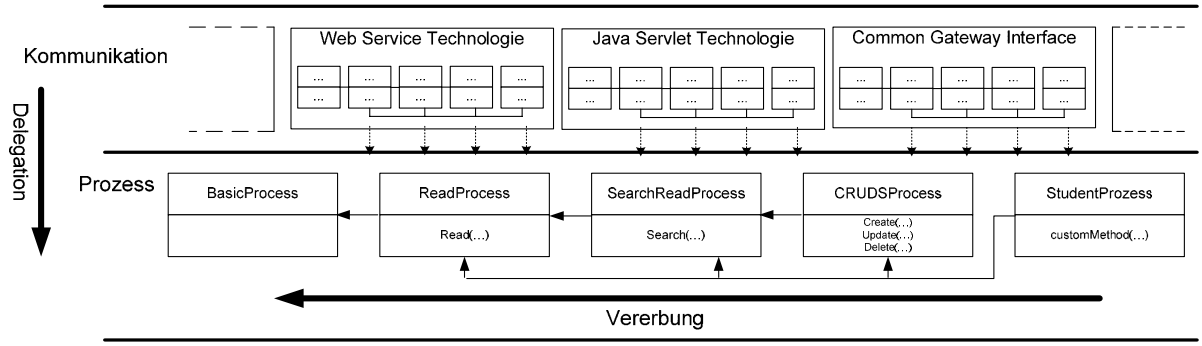


Abbildung 5-13: Kopplung der Aspekte Kommunikation und Prozess am Beispiel von CRUDS.

Während die Kernfunktionalität von CRUDS innerhalb der Prozessebene einen universalen Zugangspunkt zur Geschäftslogik definiert, übernimmt die Kommunikationsschicht deren Anbindung aus Sicht eines dedizierten Dienstnehmers. Dies wird durch Delegation an die entsprechenden konkreten Prozessimplementierungen erreicht (vgl. Abbildung 5-13).

Damit wird einerseits eine Maximierung der Flexibilität der Prozesse erreicht, da diese unabhängig von Kommunikationseinflüssen realisiert werden können. Umgekehrt wird eine Maximierung der Flexibilität der Kommunikationsebene erreicht, da diese lediglich durch Anpassen der Konfiguration an entsprechende Prozesse *delegiert* werden kann. Das Delegationsprinzip unterstützt auch die Verwendung plattformspezifischer Eigenschaften, wie das durch die Microsoft .NET Plattform unterstützte Provider Pattern (Howard, 2004).

Die WSLS Suite umfasst konkrete Prozesskomponenten zur automatischen Anbindung von CRUDS Informationsraum-Diensten an unterschiedliche Datenspeicher.

5.7 Zusammenfassung

Existierende Modellierungstechniken und -modelle beinhalten in der Regel eine äußerst umfangreiche Menge an Konzepten und Notationen, um dadurch die zugehörige Problemdomäne so erschöpfend wie möglich behandeln zu können. Im Hinblick auf die Zusammenarbeit mit Stakeholdern, d.h. Kunden und Anwendern, die meist über keinerlei Erfahrung im Bereich der Software-Entwicklung verfügen, erfordern diese jedoch einen zu hohen Lernaufwand. Zur Reduktion der diskutierten Schwierigkeiten im Bereich der Kommunikation strebt der WSLS-Ansatz eine sehr intensive Einbindung der Stakeholder in den Entwicklungsprozess an. Diese sollen eigenständig Spezifikationen von Aspekten einer Web-Anwendung verstehen, validieren, ändern und sogar entwickeln können. Um dies zu erreichen, stellt das Prinzip der Einfachheit eine Schlüsselrolle dar, so dass Domänenspezifische Sprachen (DSLs) eine ideale Alternative zu existierenden schwergewichtigen Ansätzen darstellen.

Da die Menge an DSLs einer ständigen Evolution durch die Variation und Selektion unterliegt, wird ein evolutionärer wiederverwendungsorientierter Ansatz entwickelt, der eine systematische Entwicklung, Verwaltung und Anwendung von DSLs gestattet. Um der Forderung nach kurzen Entwicklungszyklen nachzukommen wurde die Spezifikation einer Laufzeitumgebung am Muster der WSLS-Referenzarchitektur vorgestellt. Diese dient hierbei als technische Plattform, um Anwendungsbausteine (Solution Building Blocks) zu Web-Anwendungen zusammenzufügen, die dann wiederum den Erfordernissen entsprechend durch DSL-Programme konfiguriert werden können. Dadurch lassen sich diese Anwendungsbausteine beiliebig an Applikationsänderungen anpassen und erfüllen so die Anforderung der *konfigurierbaren Konvertibilität*.

Die Einführung von fachlichen und konzeptuellen Komponenten erhöht die Wiederverwendbarkeit und gestattet den Übergang des *kommunikationsorientierten* DSL Vorgehens hin zur *funktionalen Prägung* durch Komponenten. Anwendungsspezifische Basisdienste unterstützen das Prinzip des »Separation-of-Concerns« durch eine klare Auftrennung in die wesentlichen Aspekte einer Web-Anwendung in Form unterstützender Dienstelemente. Dadurch kann die Wiederverwendbarkeit erheblich gesteigert werden. Außerdem ermöglicht die Spezialisierung auf unterschiedliche Dienstelemente eine feingranulare Modellierung und Anwendung dedizierter, auf einen speziellen Aspekt fokussierte, Entwicklungsmodelle. Durch die Einführung eines Lebenszyklus werden Kontrollfunktionen befähigt, Dienstelemente miteinander zu verweben und fortschrittliche Verwebestrategien von Aspekten zu unterstützen.

Die funktionale Anpassung an sich ändernde Anforderungen wird durch eine zweifältige Komponentenevolution erreicht. Einerseits wird die sinnvolle Anwendung des Objektorientierten Paradigmas durch Typ-Generalisierung vorgenommen, um weitestgehend die Wiederverwendung existierenden Programmcodes und vorhandener Bibliotheken zu gewährleisten. Dem dabei unumgänglich entstehenden Phänomen der dominanten Dekomposition wird durch Aspekt-orientierte Modelleigenschaften begegnet. Die wohldefinierten Verknüpfungspunkte stellen den Schnittpunkt zwischen dem Lebenszyklus des WSLS-Komponentenmodells und der Aspekt-orientierten Modellierung her. Dadurch lassen sich auch nicht-funktionale Eigenschaften, wie beispielsweise Sicherheit, domänenübergreifend modellieren (Meinecke, Nussbaumer und Gaedke, 2005).

6 Modellierung der Benutzungsschnittstelle

»To some, Web design focuses on visual look and feel [..] To others, Web design is about structuring information and navigation through the document space. Others might even consider Web design to be about the technology used to build interactive Web applications. In reality, design includes all of these things and maybe more.«

(Thomas Powell)

6.1 Ausgangspunkt

Die Benutzungsschnittstelle stellt einen Dienstzugangspunkt zwischen menschlichen Benutzern und der Web-Anwendung mit ihren zugrunde liegenden Diensten dar. Sie stellt eine visuell geprägte Integrationsschicht dar, die es Benutzern gestattet – in Form von Diensten – mit der Web-Anwendung zu interagieren. Hierbei müssen Fragestellungen bezogen auf die relevanten Aspekte der Benutzungsschnittstelle betrachtet werden. Im Einzelnen sind dies:

- die *Navigation* über einen dienstorientierten Informationsraum,
- die adäquate *Darstellung* von Informationen und
- die *Nutzerinteraktion* mit dem dienstorientierten Informationsraum.

Die Separierung dieser Aspekte in möglichst orthogonal zu einander stehende Entwurfsentitäten spielt dabei eine zentrale Rolle, da sie Wiederverwendung und Anpassbarkeit einzelner Entwurfsentitäten begünstigt. Der Wirkungsbereich dieser Trennung umfasst modellbehaftete Eigenschaften sowohl bezogen auf technisch funktionale Details einer Implementierungsplattform, als auch nicht-funktionale Eigenschaften wie regionale und kulturelle Unterschiede (Becker und Mottay, 2001)[57].

[57] Becker und Mottay stellen fünf Hypothesen auf, die die regionalen und kulturellen Unterschiede für Web-Anwendungen verdeutlichen. Durch Deduktion führen Sie an verschiedenen Beispielen vor, wie sich diese Hypothesen auf die Benutzerschnittstelle von Web-Anwendungen auswirken. Ein Beispiel dafür stellen regionale Unterschiede der Leserichtung (von rechts nach links) für Texte und Dialoge im arabischen Sprachraum dar.

Dazu kommen Moden und Trends als wesentliche Bestandteile des World Wide Web. Sie verbinden ästhetischen Vorgaben mit bestimmten aktuell vorherrschenden Trends, die oftmals nicht oder nur schwer vorhersagbar sind. Ein Beispiel dafür ist der Tod von Prinzessin Diana, als sehr viele Webseiten ihre emotionale Verbundenheit durch eine temporäre Umstellung ihres Farbschemas ausdrückten und dadurch virtuelle »Trauer« ausdrückten. Im Bereich des E-Commerce kann das Ignorieren von Trends zu Einbußen oder gar dem Aus der Anwendung führen. Durch das Aufkommen des Web 2.0 wird diese ästhetische Komponente von Web-Anwendungen noch stärker in den Vordergrund gerückt als das bisher der Fall war. Daneben existieren Forderungen an die Barrierefreiheit von Web-Anwendungen, die es allen Nutzern einer Web-Anwendung gestattet, die angebotenen Dienstleistungen ohne Barrieren zu verwenden.

Barrierefreiheit der Benutzungsschnittstelle

Die BITV ergänzt das am 1. Mai 2002 in Kraft getretene Behindertengleichstellungsgesetz (BGG) und legt fest, dass alle öffentlich zugänglichen Web-Angebote des Bundes spätestens bis zum 31. Dezember 2005 barrierefrei sein müssen.

Angelehnt an Richtlinien der Web Accessibility Initiative (WAI) des World Wide Web Consortiums (W3C) werden in einer Anlage zur BITV alle Anforderungen aufgelistet, die ein barrierefreies Web-Angebot erfüllen muss. Dabei handelt es sich weitestgehend um die Empfehlungen, die den WCAG (Web Accessibility Guidelines) (Chisholm, Vanderheiden und Jacobs, 1999) zugrunde liegen.

Die Bundesarbeitsgemeinschaft »Selbsthilfe«, der »Sozialverband Deutschland« (SoVD) sowie der »Sozialverband VdK« haben das BIK (Barrierefrei Informieren und Kommunizieren)[58] aufgefordert, kontinuierlich zu prüfen, wie weit die »Verordnung zur Schaffung barrierefreier Informationstechnik« (BITV) bereits umgesetzt wurde.

Das Ergebnis der Studie ist ernüchternd: lediglich 20% der getesteten Web-Anwendungen können als »barrierefrei« bezeichnet werden (Warnke, 2006). So nennt der Bericht eine Reihe von Problemen und Mängeln, die zu diesem Ergebnis geführt haben. Zwei besonders gravierende Probleme betreffen die Strukturierung der Inhalte sowie die Qualität des HTML-Codes:

- **Darstellungsabhängige Strukturierung der Inhalte**: 20,5% der Inhalte sind nicht darstellungsunabhängig strukturiert, 55,5% sind die Inhalte nur zum Teil strukturiert. Das bedeutet, dass für Strukturierung vorgesehene Auszeichnungselemente wie Tabellen für Layout-Maßnahmen fälschlicherweise verwendet wurden. Insgesamt 76% der geprüften Web-Anwendungen erfüllen also eine entscheidende Grundbedingung der gleichwertigen Nutzbarkeit von Web-Angeboten nicht oder nur zum Teil.
- **Invalider Code**: 70% der geprüften Web-Anwendungen waren nicht durchgängig valide. Die formale Standardkonformität bildet allerdings eine wichtige Grundbedingung für Barrierefreiheit, unter anderem, weil sich standardkonforme Seiten leichter für spezielle Ausgabegeräte anpassen lassen.

[58] Aktuelle Testergebnisse des BIK sowie das Testverfahren können unter http://www.bik-online.info nachgelesen werden. Bereits eruierte Tests können unter http://www.bitvtest.de/ eingesehen werden.

Die Gründe für diese nachlässige Betrachtung sind offensichtlich: Das Spannungsverhältnis zwischen barrierefreier Darstellung und einem ästhetisch anspruchsvollen Design liegt an unzureichend methodischer Unterstützung bei der Entwicklung von Web-Anwendungen. Durch das Aufkeimen komponentenbasierter Ansätze, die eine Wiederverwendung einzelner Teile unterstützen, wird dies durch das Fehlen übergeordneter Strukturen noch deutlich erschwert.

Auch um Kosten einzusparen wird eine Optimierung der »visuellen Erscheinung« oftmals über das »Erfahren« gestellt. Dazu gehört neben den visuellen Eigenschaften auch eine konsistente Modellierung und Umsetzung der Navigationsmöglichkeiten innerhalb der Web-Anwendung. Gerade auch aufgrund sich ändernder Anforderungen ist hier auf eine strikte Konsistenz zu achten[59]. Eine Metrik hierbei ist die »conversion rate« (Umwandlungsrate, Kaufrate). Darunter werden alle Maßnahmen zusammengefasst, die es ermöglichen, die Versorgungs-Performanz von Web-Anwendungen zu messen und zu ermitteln. Dazu gehören abgeschlossene Käufe genauso wie die Bereitstellung von Informationen, etwa im Bereich des Integrierten Informationsmanagement an Hochschulen. Garrett führt die »conversion rate« als wichtigen Indikator für den »Return on Investment« als Maß der Effektivität der Benutzungsschnittstelle, nach Garrett bestimmt durch die »User Experience«, ein (Garrett, 2003).

Um dieses »Erfahren« in ein positives Erlebnis bei den einzelnen Benutzern einer Web-Anwendung zu transformieren, bedarf es einer konsistenten, barrierefrei zugänglichen und ästhetisch ansprechenden Web-Anwendung.

Modellierung der Benutzungsschnittstelle durch Wiederverwendung

Während die meisten Bereiche bei der Anwendungsintegration (EAI) gut und ausreichend besetzt sind, finden sich im Bereich der Benutzungsschnittstellenintegration kaum Ansätze. So stellen manuelle und dialoggeführte Verfahren (Riehm und Vogler, 1996) lediglich eine »Notlösung“« (Kaib, 2002) zur Präsentationsintegration dar. Gerade in dienstorientierten Web-Anwendungen nimmt die effiziente Gestaltung der Benutzungsschnittstelle eine wichtige Rolle ein. Hierbei liegt der Fokus auf der Wiederverwendung existierender Komponenten und Dienste. Eng mit der Wiederverwendung verbunden sind dabei Fragen nach Automation einzelner Aspekte oder der Qualität der resultierenden Komposition, also, wie effizient sich bestehende Problemlösungsstrategien in Form von Komponenten oder Diensten wieder verwenden lassen. Neben Effizienzanforderungen stehen aber auch Fragen nach der Güte und Beschaffenheit einer Anwendungskomposition im Vordergrund, also wie sich die Komposition auf die Einflussgrößen der Benutzungsschnittstelle auswirkt.

Nachdem der Informationsraum durch Dienste nutzbar gemacht wurde und durch die Einführung fachlicher und konzeptueller Komponenten die Grundbausteine für die Anwendungskonstruktion gelegt wurden, wird nun vorgestellt, wie Web-Anwendungen mit deren Hilfe modelliert werden können. Dabei wird im folgenden Abschnitt zunächst der Konstruktionsrahmen auf Basis einer Verknüpfung der in Kapitel 5 eingeführten Komponenten zu Web-Anwendungen entwickelt. Danach werden sukzessive Modelle und methodische Hilfe-

[59] So stellt ein Gutachten des BIK beim Webauftritt der Bundesagentur für Arbeit fest: „Die Hauptmängel liegen im Bereich der Orientierung und Navigation, das Angebot ist unübersichtlich.“

stellungen für die dedizierten Aspekte Navigation, Präsentation und Dialog der Benutzungsschnittstelle, wie sie in der Problemcharakteristik Benutzungsschnittstelle in Abschnitt 2.3.3 gefordert wurden, vorgestellt.

6.2 Modellierung der Anwendung

Zunächst wird der Konstruktionsrahmen festgelegt, der dann eine dedizierte Hinwendung zu den Benutzungsschnittstellen-spezifischen Aspekten fördern kann. Dieser Rahmen umspannt das Grobgerüst einer dienstorientierten Web-Anwendung. Dazu werden die konzeptuellen Komponenten in Form von Domänen aufgegriffen. Diese bilden in ihrer Gesamtheit den Domänenraum als eine Menge von Konzepten, aus denen eine Anwendung zusammengebaut werden kann. Jedes dieser Konzepte wird geprägt durch die Assoziation mit den fachlichen Komponenten Kontrollfunktion und verschiedenen Dienstelementen. Diese können mit Hilfe der im vorigen Kapitel eingeführten Tagging-Architektur durch Eigenschaften (engl. *properties*) angereichert werden, die eine dedizierte Anpassung an vorherrschende Gegebenheiten gestatten. Die Domänen dienen dabei als ein Konfigurationskontext, der die Versorgung der definierenden fachlichen Komponenten mit der in diesem Kontext definierten Konfiguration versorgen.

Aus logischer Sicht wird die Architektur in die zwei Ebenen »Anwendungsebene« und »Konfigurationsebene« aufgetrennt (siehe Abbildung 6-1). Diese Auftrennung gestattet die systematische Entwicklung und Evolution von Web-Anwendungen durch die Wiederverwendung existierender Software Artefakte. Durch die Fokussierung auf das Hinzukonfigurieren einzelner Komponenten und Aspekte lassen sich Modifikationen gemäß sich ändernder Anforderungen anpassen, wodurch das »Konfigurieren anstatt Programmieren« Paradigma umgesetzt wird.

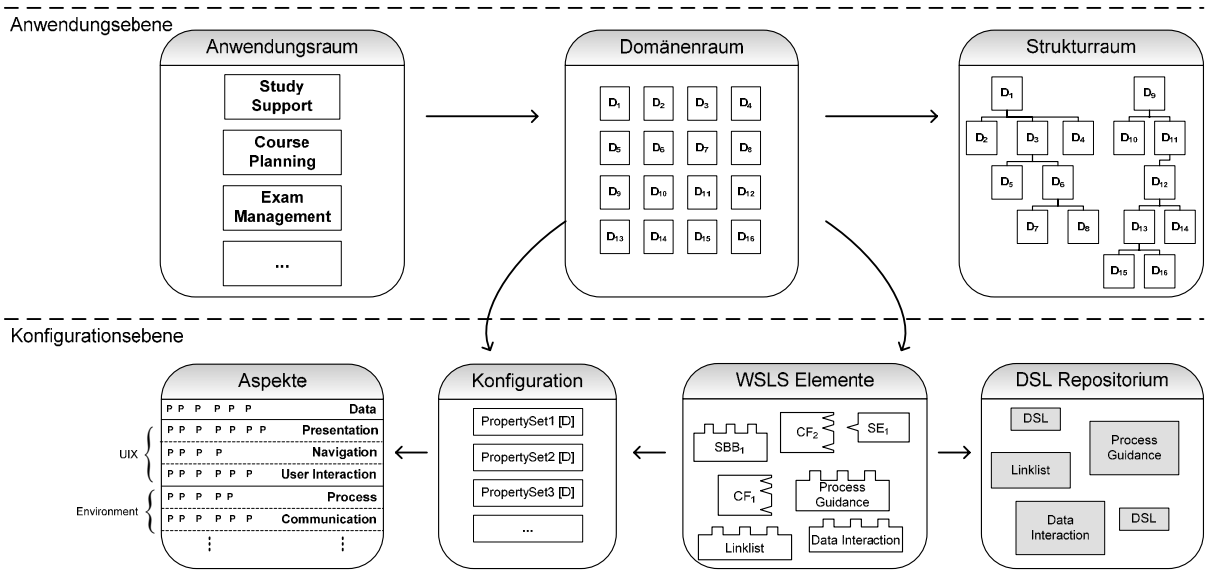


Abbildung 6-1: Logische Architektur von WSLS mit den Ebenen Anwendung und Konfiguration.

Damit können auch Anforderungen nach kürzeren Produktionszyklen erfüllt werden, indem Web-Anwendungen in prototypischer Weise erstellt werden und ihre Funktionalität sukzes-

sive durch Hinzufügen entsprechender fachlicher Komponenten erreicht wird. Darüber hinaus lassen sich auf diese Weise auch schnell lauffähige Prototypen erstellen, die als Diskussionsgrundlage mit Kunden dienen können. Dadurch können Fehlentwicklungen und Missverständnisse schon sehr früh erkannt werden und dementsprechend früh darauf reagiert werden.

Die **Anwendungsebene** besteht aus drei Datenräumen (engl. *spaces*):

- Ein **Anwendungsraum** (engl. *site space*) verwaltet alle Applikationen, die in einem WSLS System definiert werden können. Eine Anwendung stellt eine dedizierte Komposition einzelner Baugruppen von konzeptionellen Komponenten dar.
- Ein **Domänenraum** (engl. *domain space*) definiert alle konzeptuellen Komponenten aus denen Anwendungen konstruiert werden können. Die Domänen bilden den Übergang von der abstrakt-konzeptuellen Sicht einer Anwendung auf die funktional-fachliche Ausprägung.
- Ein **Strukturraum** (engl. *structure space*) kapselt die Verknüpfungen von Domänen zu Domänenbaugruppen innerhalb einer Anwendung.

Diese Trennung ermöglicht eine dienstorientierte Anwendungskonstruktion mit Domänen als den zentralen Einheiten aus denen diese Anwendungen aufgebaut werden. Sie symbolisieren abgeschlossene, wiederverwendbare Einheiten, deren Schnittstelle durch die aufgeprägten Kontrollfunktionen in dienstorientierter Art und Weise zur Verfügung gestellt werden. Eine Domäne in ihrer Gesamtheit bildet einen Verbund fachlicher Komponenten, die in dieser Gesamtheit einen dedizierten Dienst erbringt. In dieser dienstorientierten Konstruktionsweise von Web-Anwendungen liegt ein zentraler Unterschied zu den aktuell diskutierten Forschungsansätzen im Web Engineering (siehe Abschnitt 3.2).

Die **Konfigurationsebene** umfasst die physischen Eigenschaften und fachlichen Komponenten, die in WSLS verwendet werden können. Die Fachkomponenten definieren jeweils eine Menge von Eigenschaften, die sie zur Erbringung ihrer Funktion benötigen (*property set*). Jede dieser Eigenschaften entspricht einer Konfigurationsmöglichkeit zur dedizierten Anpassung des Verhaltens (*behavior*) dieser fachlichen Komponente innerhalb einer Domäne. Die physischen Eigenschaften (*properties*) werden entsprechend der Klassifizierung der ermittelten Einflussgrößen einer Web-Anwendung in die Kategorien

- Benutzungsschnittstelle (Präsentation, Navigation, Dialog) *und*
- Umgebung (Daten, Kommunikation und Prozess) eingeteilt.

Die technische Realisierung dieser Eigenschaften reicht von einfachen Name-Wert-Paaren bis hin zu umfangreichen Konfigurationssprachen. Ein Beispiel für diese umfangreichen Sprachen bilden die in Kapitel 5 eingeführten DSLs. Damit erhält jede Domäne *d* eine Menge von Eigenschaften p_{ji}, die durch die jeweils aufgeprägten fachlichen Komponenten k_j als Konfigurationsmöglichkeiten zur Verfügung gestellt werden. Um zwischen den definierten, also mit Werten besetzten, und den definierbaren, also möglichen, Eigenschaften zu unterscheiden, werden nun zwei Funktionen zu deren Bestimmung eingeführt.

Sei *H* die Hüllenfunktion, die zu jeder fachlichen Komponente k_j deren Eigenschaften p_{ji} als Konfigurationsmöglichkeiten berechnet, dann gilt:

H (d) = $\bigcup_{k_j} H(k_j)$ wobei k_j eine der Domäne *d* aufgeprägte fachliche Komponente ist.

Sei E die Eigenschaftsfunktion, die zu jeder Domäne *d* die definierten Eigenschaften *p* berechnet, dann gilt:

$E(d) = \{p \mid p \in H(d) \wedge p \; ist definiert\}$ wobei H(d) die Hülle von d bezeichnet.

Verwaisung von Eigenschaften

Je nach Implementierung der Eigenschaftsfunktion kann ein Verwaisen von Eigenschaften auftreten. Die Verwaisung einer Eigenschaft ist immer dann gegeben, wenn ein Wert für eine Eigenschaft vorhanden ist und gleichzeitig die fachliche Komponente, die diese Eigenschaft deklariert hat, *nicht mehr* zur Domäne gehört. Ein solcher Sachverhalt kann immer dann eintreten, wenn eine Domäne umkonfiguriert wird, also beispielsweise eine fachliche Komponente durch eine andere ersetzt wird.

Die obige Eigenschaftsfunktion E(d) betrachtet lediglich jene Eigenschaften, die in der Hülle von d liegen. Die Verwaisung von Eigenschaften hat aber den Vorteil, dass Werte einer Konfiguration konserviert werden können und im Falle einer erneuten Konfiguration mit der $p_{verwaist}$ definierenden Komponente reaktivierbar sind. Gerade in sich stark ändernden Umgebungen oder prototypischen Anwendungen ist eine mehrfache Rekonfiguration von Domänen zu erwarten und eine Unterstützung der Verwaisung deshalb sinnvoll. Dementsprechend wird neben der Eigenschaftsfunktion auch noch die Verwaisungsfunktion V(d) definiert.

Sei V die Verwaisungsfunktion, die zu jeder Domäne d die verwaisten Eigenschaften *p* berechnet, dann gilt:

$V(d) = \{p \mid p \notin H(d) \wedge p \; ist definiert\}$ wobei H(d) die Hülle von d bezeichnet.

Für eine technische Umsetzung der Hüllenfunktion können Mechanismen zur Reflexion und Introspektion moderner Programmiersprachen herangezogen werden. Eigenschaften lassen sich dann analog zu den in Abschnitt 5.5.1 eingeführten Auszeichnungselementen definieren. Die Hüllenfunktion iteriert dann über alle Komponenten und errechnet die mögliche Menge konfigurierbarer Werte.

6.2.1 Anwendungsebene

Eine Anwendung in WSLS stellt einen Kontext auf den Domänenraum dar. Dieser Kontext definiert einen gerichteten, azyklischen Graphen (Halin, 1989) aus lose gekoppelten, wiederverwendbaren Domänen. Die Verbindungskanten zwischen den Domänen beschreiben die strukturelle Ausprägung und definieren in ihrer Gesamtheit die Anwendung.

Dabei wird der Domänenraum *D* mit der Menge der Ecken *E* assoziiert. Die Menge der Verweise *L* im Strukturraum entspricht der Menge der Kanten *K*. Ein Anwendungskontext *A* ist ein Tupel *(D, L),* das den zugehörigen Graphen dieser Anwendung beschreibt.

Der visualisierte Teil einer Web-Anwendung stellt dabei immer einen Baum über die dargestellten Domänen dar. Das bedeutet, dass in der Visualisierung wiederverwendeter Domänen, die zunächst als Referenz im Anwendungskontext existiert haben, nun als eigenständige Einheit in Form eines Knotens in einem Baum existieren. Dieser Baum wird als *Visualisierungsbaum* bezeichnet.

In Abbildung 6-2 wird der Zusammenhang zwischen dem Visualisierungsbaum und dem Anwendungskontext verdeutlicht. Im dargestellten Beispiel wird die Domäne D_5 in den Domänen D_2 und D_6 wiederverwendet. Während sie im Anwendungskontext lediglich referenziert wird, stellt sie im Visualisierungsbaum einen eigenständigen Knoten für jede Visualisierung dar. Auf diese Weise lässt sich Domäne D_5 als eigenständige Einheit wiederverwenden, bietet aber die Möglichkeit im jeweiligen Verwendungskontext dediziert angepasst zu werden. Diese Verwendungskontexte können anhand des Pfades im zugehörigen Visualisierungsbaum unterschieden werden, im Beispiel sind dies die Pfade $D_1 \rightarrow D_2 \rightarrow D_5$ und $D_1 \rightarrow D_6 \rightarrow D_5$.

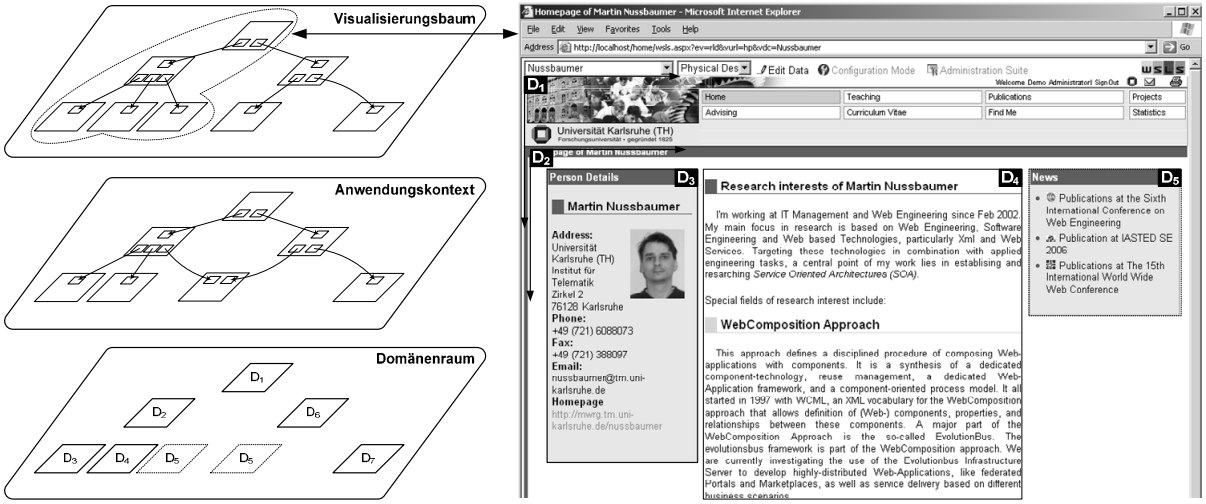


Abbildung 6-2: Wiederverwendung von Domänen im Anwendungskontext und deren Darstellung im zugehörigen Visualisierungsbaum.

Sei $d \in D$ eine Domäne des zugehörigen Anwendungskontextes *A*, dann ist |•d| die Zahl der Eingangskanten dieser Domäne, |d•| bezeichnet die Menge der ausgehenden Kanten der Domäne *d*.

Eine Domäne mit dem Eingangsgrad null (|•d| = 0) wird eine *initiale Anwendungsdomäne* genannt. Im dargestellten Beispiel ist dies Domäne D_1. Ist der Ausgangsgrad einer Domäne null (|d•| = 0), so wird sie als *Domänendienst* bezeichnet. Ein Beispiel dafür bilden die Domänen D_3, D_4 und D_5. Ist der Ausgangsgrad größer null (|d•|>0), so wird sie als *Strukturierungsdomäne* bezeichnet, deren Aufgabe in der Aufbereitung der weiteren Formation und Gestaltung der ihr zugeordneten Kinddomänen liegt. Beispiele von Strukturierungsdomänen in der Abbildung sind die Domänen D_1 und D_2.

Das Maß der Wiederverwendung einer Domäne kann über ihren Eingangsgrad bestimmt werden. Der Eingangsgrad einer Domäne |•d| wird dementsprechend auch als Wiederverwendungsgrad bezeichnet. Je höher dieser Grad, desto häufiger wird die Domäne wiederverwendet.

Durch Kombination aller Anwendungskontexte erhält man eine *Hyperanwendung*[60]. Eine Hyperanwendung stellt formal einen Multigraphen dar, der alle Verweiskanten zwischen den Domänen in sich vereinigt. Eine Hyperanwendung hat zum Ziel die semantische Ähnlichkeit zwischen den Domänenbaugruppen einer Anwendung abzubilden.

Da jede Anwendung ihren eigenen Domänenraum definiert, entspricht eine Hyperanwendung zunächst einem Wald (Goos und Zimmermann, 2005) aus diesen Anwendungskontexten. Ergänzt man die Hyperanwendung nun um eine Projektionsfunktion π, welche die definierten Eigenschaften der Konfigurationsebene einer Domäne *E(d)* auf eine beliebig wählbare Untermenge dieser Eigenschaften projiziert, so lassen sich Domänen bezüglich dieser Abbildung miteinander vergleichen. Dadurch wandelt sich der Wald aller Anwendungskontexte zu einem Hypergraphen mit Mehrfachkanten.

Der Algorithmus zur Bestimmung einer Mehrfachkante kann durch eine Funktion bereitgestellt werden, die es gestattet die projizierten Eigenschaften p miteinander zu vergleichen. Als ein Beispiel wird eine einfache Funktion vorgestellt, die die Gleichheit der Eigenschaftswerte prüfen kann. Da unterschiedliche Eigenschaften unterschiedliche Typen besitzen können, wird im Folgenden davon ausgegangen, dass ein Äquivalenzoperator ≡ für jeden dieser Typen definiert wurde.

Eine Mehrfachkante erhält man demnach genau dann, wenn gilt:

Sei $A_1 = (D_1, L_1)$ und $A_2 = (D_2, L_2)$ Anwendungskontexte und $d_{11}, d_{12} \in D_1$ und $d_{21}, d_{22} \in D_2$.

(i) $\forall p_i \in \pi(d_{11}) : \exists_1 p_j \in \pi(d_{21}), so\, dass\ p_i \equiv p_j \wedge \forall p_i \in \pi(d_{12}) : \exists_1 p_j \in \pi(d_{22}), so\, dass\ p_i \equiv p_j$

(ii) $(d_{11}, d_{12}) \in A_1 \wedge (d_{21}, d_{22}) \in A_2$

Die Forderung (i) stellt sicher, dass nach erfolgter Projektion der entsprechenden Domänen deren Eigenschaften äquivalent sind. Die Forderung (ii) stellt sicher, dass nur dann Mehrfachkanten gebildet werden, wenn diese auch in den ursprünglichen Anwendungskontexten existiert haben. Die Wirkungsweise von Forderungen (i) und (ii) wird anhand des in Abbildung 6-3 dargestellten Beispiels grafisch verdeutlicht.

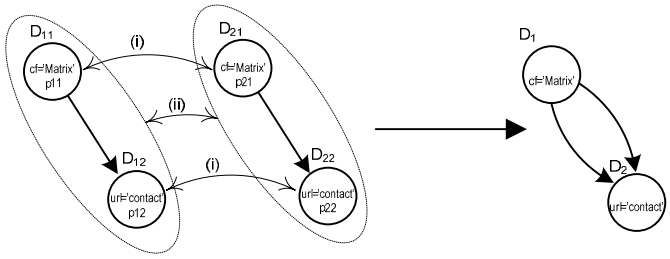


Abbildung 6-3: Zusammenspiel der Eigenschaften (i) und (ii) zur Bildung einer Mehrfachkante in der Hyperanwendung.

[60] Der Begriff Hyperanwendung wird zurückgeführt auf den formalen Begriff des *Hypergraphen*. Dieser verfügt über die Möglichkeit von Mehrfachkanten zwischen einzelnen Knoten.

Beispiele für Projektionen stellen die Abbildung auf die verwendeten fachlichen Komponenten (im Beispiel **c**ontrol**f**unction='Matrix') oder ausgewählte Eigenschaften (im Beispiel url='contact') dar.

Eine Hyperanwendung definiert sich demnach als $H_{App} = \left(\bigcup_{d \in D}, \bigcup_{l \in L}, \pi \right)$, wobei *D* die Menge aller Domänenräume aller Anwendungskontexte und L die Menge aller Verweise der Strukturräume aller Anwendungskontexte der verfügbaren Anwendungen bezeichnet.

Basierend auf dem in Abbildung 6-2 dargestellten Visualisierungsbaum soll nun exemplarisch eine Hyperanwendung auf Basis von zwei Anwendungen konstruiert werden, die durch ihre Anwendungskontexte $A_1 = (D_1, L_1)$ und $A_2 = (D_2, L_2)$ gegeben sind. Die verwendete Projektion zur Definition dieser Hyperanwendung umfasse die Abbildung auf eine Eigenschaft *url* und den Typ der Kontrollfunktion *cf*. Abbildung 6-4 zeigt linker Hand die zwei zu den Anwendungskontexten A_1 und A_2 gehörenden Visualisierungsbäume. Die resultierende Hyperanwendung mit den nach der Projektion errechneten Mehrfachkanten ist rechter Hand dargestellt.

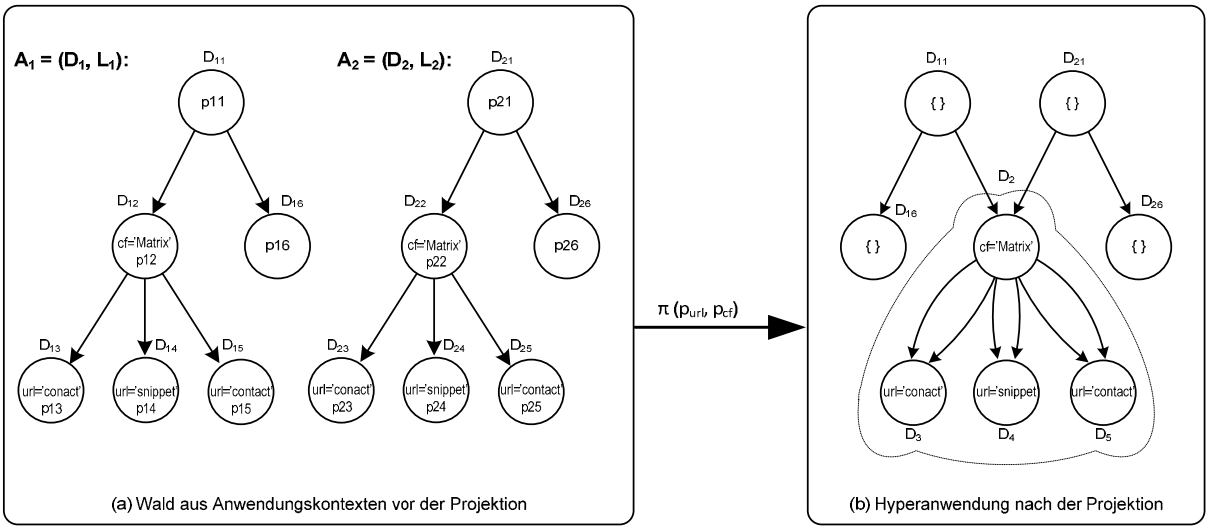


Abbildung 6-4: Konstruktion einer Hyperanwendung durch Projektion von wählbaren Eigenschaften der Konfigurationsebene.

Basierend auf den formalen Definitionen des Anwendungskontextes und der Hyperanwendung lassen sich Bewertungsmaßstäbe für den Entwicklungsprozess hinsichtlich des Maßes der Wiederverwendung ableiten. Die Zahl der Eingangskanten einer projizierten Domäne innerhalb der Hyperanwendung definiert die Anzahl der Verwendungen dieser Domänenkonfiguration. Je höher der Index, desto häufiger wird diese Domänenkonfiguration wiederverwendet.

Hyperanwendung als Indikator für die domänenbezogene Maturität

Das WebComposition Prozess Modell sieht die Phase der Maturität in der Evolutionsdurchführung (Gaedke, 2000). Damit wird der »Reifegrad« von Zusammenschlüssen einzelner Entitäten des Repositoriums bezüglich ihrer Eignung zur Wiederverwendung beschrieben. Durch die Abstraktion in Form einer Hyperanwendung kann eine systematische Unterstützung zur Bestimmung maturierter Verbünde mit Hilfe formaler Grundlagen ergänzt werden.

Die Maturität einer Teilmenge von Domänen bezeichnet hierbei die mehrfache Wiederverwendung bestimmter zusammenhängender Bereiche innerhalb des Domänenraums. Diese Bereiche bilden formal Teilgraphen der Hyperanwendung. Im abgebildeten Beispiel stellt die Domänengruppe $\{D_2, D_3, D_4, D_5\}$ der abgebildeten Hyperanwendung einen solchen Teilgraphen dar. Der gesamte Verbund bildet einen potenziellen Kandidaten, um als eigenständiger Anwendungsbaustein wiederverwendet zu werden. Die Anzahl der Mehrfachkanten kann hierbei als Indikator zur Identifikation »gereifter« Zusammenschlüsse einzelner Domänen herangezogen werden. Anders ausgedrückt lässt sich dieser Reifegrad als eine sinnvolle Kombination der entsprechenden Domänen beschreiben. Dadurch empfiehlt sich diese Kombination wiederum als Einheit zur Wiederverwendung in Form eines weiteren Anwendungsbausteins.

Die WSLS Suite unterstützt das Bilden von solch maturierten anwendungsspezifischen Domänenbaugruppen in einem dafür zuständigen Bereich, dem *Deployment*. In (Gaedke, Meinecke und Nussbaumer, 2004a) wird eine Rahmenarchitektur vorgestellt, die beschreibt, wie eine Verteilung unter den Aspekten *Sicherheit* und *Lizenzierung* von Komponenten erreicht werden kann.

6.2.2 Konfigurationsebene

Die Ebene der Konfiguration umfasst die Definition von physischen Eigenschaften und fachlichen Komponenten, die in WSLS verwendet werden können. Die durch fachliche Komponenten induzierten Eigenschaften stellen dabei eine Möglichkeit zur Adaption dar, die es gestattet, die ihnen zugedachte Funktion unter unterschiedlichen vorherrschenden Gegebenheiten zu erbringen. Jede dieser Eigenschaften entspricht somit einer Konfigurationsmöglichkeit zur dedizierten Anpassung des Verhaltens der entsprechenden Komponente innerhalb einer Anwendungsdomäne.

Die Granularität einer Eigenschaft hängt vom jeweils modellierten Typ ab. Die mögliche Palette umspannt dabei elementare Datentypen wie numerische Einheiten, Datenformate oder Zeichenkette bis hin zu umfangreichen domänenspezifischen Sprachkonstrukten zur Unterstützung und Integration von DSL Programmen basierend auf dem DSL-Ansatz. Die Eigenschaftswerte, die von fachlichen Komponenten definiert werden, können innerhalb einer Domäne dediziert verwaltet – also angelegt, gelesen, geändert oder gelöscht – werden. Dazu bedient sie sich der Phase Konfigurieren im Lebenszyklus der WSLS-Elemente (vgl. Abschnitt 5.4.1).

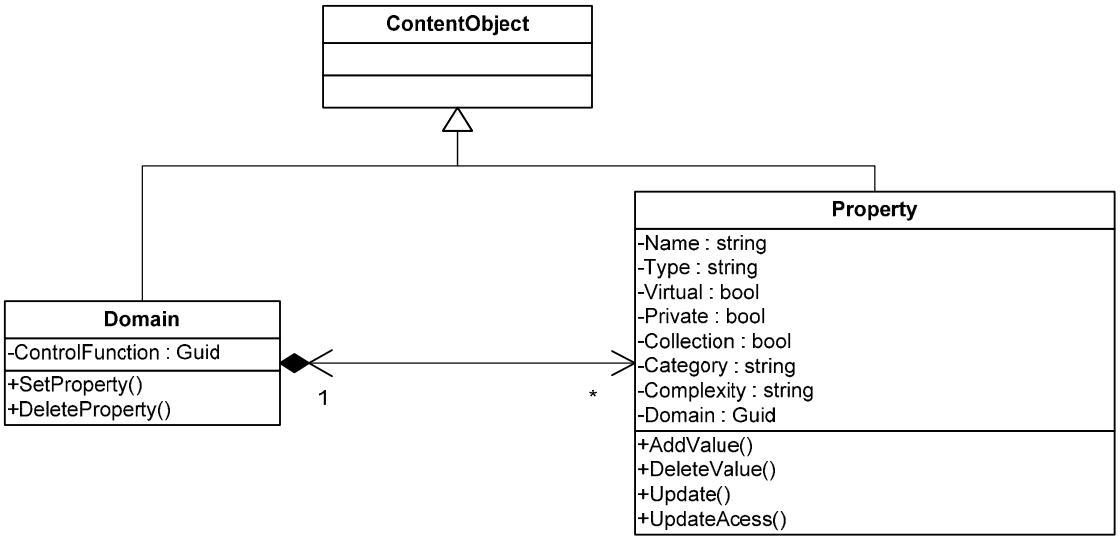


Abbildung 6-5: Zusammenspiel zwischen einer Domäne und ihren Eigenschaften (properties).

Der Zusammenhang zwischen einer Domäne und ihren Eigenschaften ist in Abbildung 6-5 in Form eines UML Diagramms dargestellt. Eine Domäne kann eine beliebige Anzahl Eigenschaften besitzen. Die Anzahl möglicher Konfigurationsmöglichkeiten hängt dabei von der aufgeprägten Kontrollfunktion ab. Die relevanten Attribute einer Eigenschaft sind in die Bereiche Klassifizierung (category, complexity, class, privat, virtual) und Wertebereich (name, type, collection) gruppiert.

Klassifizierung: Die Klassifizierungsmerkmale gestatten die Gruppierung der Eigenschaften in bestimmte vordefinierte Bereiche. Das umfasst die Klassifikation der verschiedenen durch Separation-Of-Concerns gewonnenen Einflussgrößen wie Inhalt, Darstellung oder Navigation (Attribut class). Außerdem lassen sich Kategorisierungscharakteristiken bestimmen, die den Komplexitätsgrad einer Eigenschaft beschreiben (Attribut complexity). Dadurch lassen sich Eigenschaften für bestimmte Nutzergruppen definieren oder Vorbedingungen modellieren.

Die Graph-Repräsentation als Visualisierungsbaumes über Domänen erlaubt die Anwendung des Vererbungskonzepts der den Domänen zugeordneten Eigenschaften. Die Vererbung von Eigenschaften einer Domäne auf ihre Kinder ist wünschenswert, da auf diese Weise Eigenschaftsdefinitionen einer höheren Hierarchiestufe wiederverwendet und gegebenenfalls sogar stringent propagiert werden können.

Die Vererbung von Eigenschaften wird über die zwei Zustände »privat« und »virtuell« gesteuert, die jeweils die boolschen Werte *wahr* und *falsch* annehmen können. Während der Zustand von »privat« die Sichtbarkeit der Eigenschaft beschreibt, definiert der Zustand »virtuell«, ob die Eigenschaft von einer in der Hierarchie tiefer gelegenen Domäne überschrieben werden darf. Das hat den Vorteil, dass innerhalb einer Domänenhierarchie Vorgaben definiert werden können, die von den untergeordneten Anwendungsdomänen konsequent verwendet und umgesetzt werden müssen. Dadurch ergeben sich drei sinnvolle Kombinationen, die in Tabelle 6-1 näher erläutert werden.

(privat, virtuell)	(Falsch, Falsch)	(Falsch, Wahr)
(Wahr, Falsch)	**Lokaler Einflussbereich** Damit können Eigenschaften lediglich für das Umfeld einer Domäne definiert werden, ohne eine Vererbung zu initiieren. Das ist immer dann sinnvoll, wenn es sich um eine Eigenschaft handelt, die von einem Dienst dediziert zur Verfügung gestellt werden soll. Die Eigenschaft erhält dann lediglich lokale Sichtbarkeit.	**Stringente Propagierung** Diese Kombination ermöglicht es, Eigenschaften vererben zu können, die von Nachkommen *nicht* überschrieben werden können. Diese Einstellung ist immer dann sinnvoll, wenn bestimmte Werte von Eigenschaften als Richtlinie durchgesetzt werden sollen. Dadurch lassen sich Konfigurationsänderungen effizient auf ganze Anwendungskontexte oder ausgewählte Teilgraphen anwenden.
(Wahr, Wahr)	**Undefinierter Bereich** Dieser Fall muss nicht berücksichtigt werden, da eine private Eigenschaft an sich nicht vererbt werden kann und ist folglich auch nicht überschreibbar.	**Prototyp-Instanz** Diese Kombination ermöglicht das prototypische Vererben von Eigenschaften, die von ihren Nachkommen verändert werden können. Hier müssen aufgrund der Grapheigenschaften eines Anwendungskontextes (und der damit möglichen Referenzierbarkeit von Domänen), zwei Spezialfälle berücksichtigt werden: Referenz- und Wertesemantik. Bei Wertesemantik muss eine Domäne zunächst eine eigene Eigenschaft durch Erstellen einer Kopie der vererbten definieren. Bei Referenzsemantik wird die geerbte Eigenschaft in der definierenden Domäne direkt geändert.

Tabelle 6-1: Übersicht der sinnvollen Zustände für Eigenschaften innerhalb einer Domänenhierarchie.

Wertebereich: Der Wertebereich bestimmt, welchen semantischen Wert eine Eigenschaft besitzen darf. Dabei können beliebige Typen zum Einsatz kommen. Die Liste dieser Typen ist prinzipiell nicht abgeschlossen und wird lediglich durch eine erweiterbare Grundmenge vorgegeben. Gerade hinsichtlich des Einsatzes von Domänen-spezfischen Sprachen, die dann je einen eigenen Typ definieren, ist das Vorhalten dedizierter Eigenschaftseditoren sinnvoll, da auf diese Weise effizient DIM-Editoren (vgl. Abschnitt 0) erstellt werden können.

In Abbildung 6-6 ist ein generischer Eigenschaftseditor dargestellt, der in der Lage ist, alle in der WSLS-Referenzarchitektur definierten Eigenschaften zu verwalten. Neben dem manuellen Suchen mit textuellen Filtern, werden auch die unterschiedlichen Konzepte der Eigenschaften einer Domäne unterstützt. Dazu gehören die in der Konfigurationsebene eingeführten Menge von Eigenschaften: Eigenschaftsfunktion E(d), Verwaisungsfunktion V(d), Hüllenfunktion H(d).

Daneben gestattet ein generischer Editor die Auswertung der unterschiedlich klassifizierten Eigenschaften nach Einflussgröße (category), nach Komplexität (complexity) und deren Charakteristik. Der wichtigste Bereich besteht in der Zuordnung dedizierter Behandlungsroutinen abhängig vom zu editierenden Typ eines Wertes. Dadurch können so genannte »property handler« sukzessive dem generischen Editor hinzugefügt werden, ohne dass er geändert werden muss.

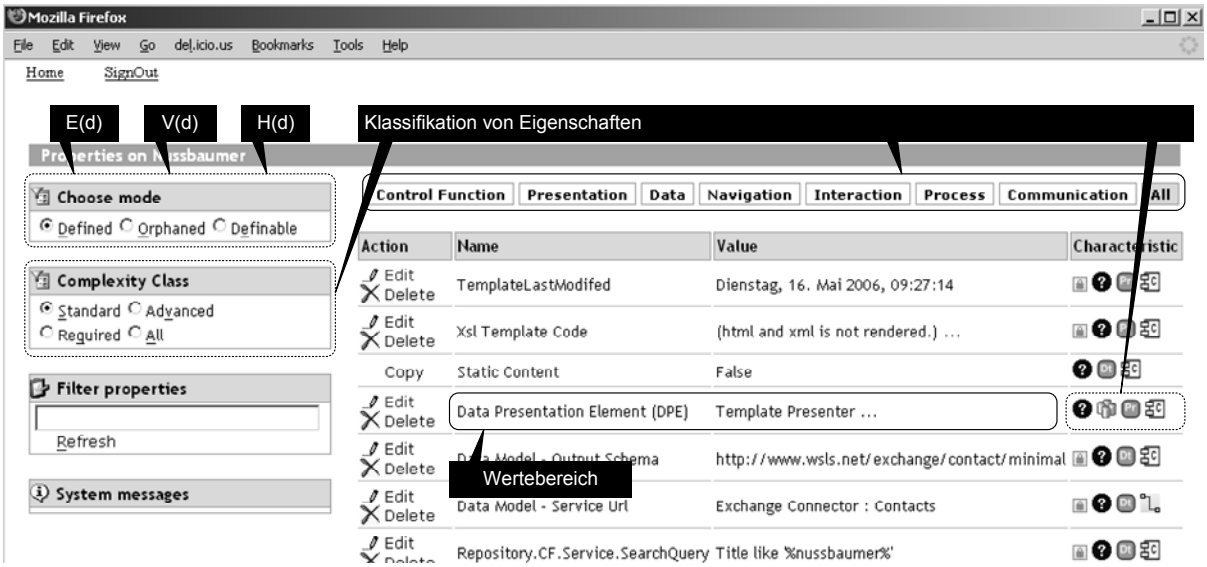


Abbildung 6-6: Ein generischer Eigenschaftseditor zur Unterstützung der Verwaltung von Eigenschaften wie sie in der WSLS-Referenzarchitektur definiert werden.

Der Editor selbst verfügt über eine Plug-in Architektur, die es ihm gestattet verschiedene property handler abhängig von einem bestimmten Typ zur Bearbeitung einer gewählten Eigenschaft aufzurufen. Jedem Typ kann dabei ein eigener Handler zugewiesen werden. Dadurch kann die Menge an neuen Typvereinbarungen stetig im Sinne der Evolution der Anwendung zunehmen und der Editor organisch wachsen.

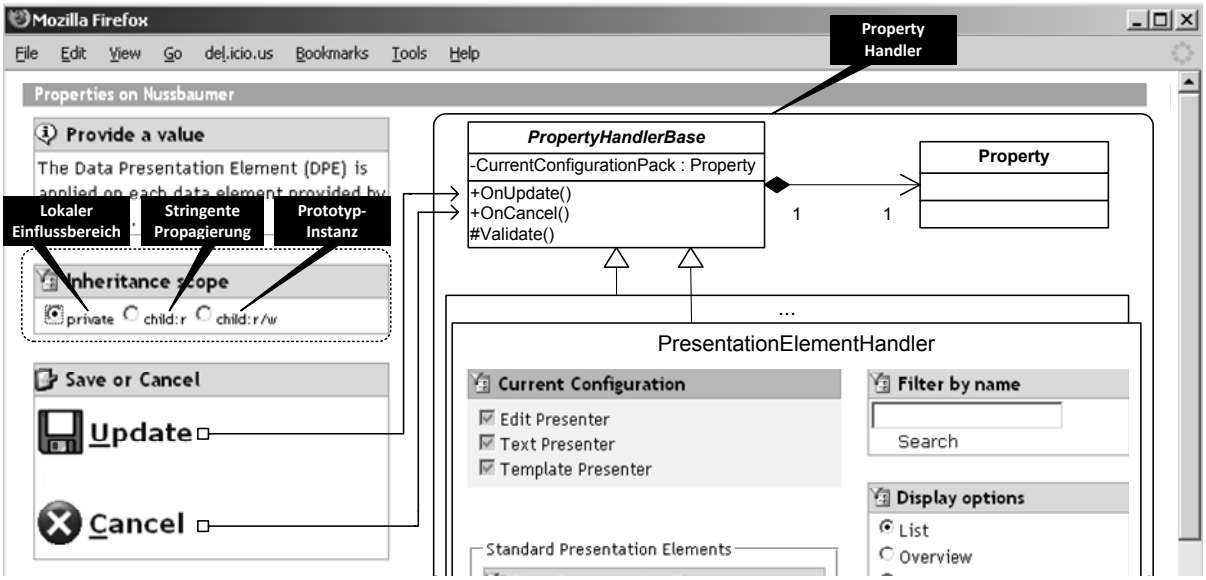


Abbildung 6-7: Property Handler für die Wahl eines Darstellungselements.

Der schematische Aufbau eines property handlers ist in Abbildung 6-7 dargestellt. Um ein Höchstmaß an Generik zu erzielen wird eine abstrakte Klasse PropertyHandlerBase definiert, die als Grundlage aller konkreter Handlerklassen dient. Die Bearbeitungsroutinen (Update und Cancel) sind dem Entwurfsmuster Schablone (Gamma, Helm, Johnson und Vlissides, 1995) nachempfunden. Dadurch lässt sich die Kommunikation mit einem dedizierten Datenraum für die Eigenschaften von ihrer inhaltlichen Bearbeitung trennen. Das gilt auch für die

drei unterstützten Vererbungsmechanismen »Lokaler Einflussbereich«, »Stringente Propagierung« und »Prototyp-Instanz«.

6.2.3 Dekoration als Realisierungsprinzip von Anwendungssichten

Um beliebige Aufgaben für die aus Domänen aufgebaute Web-Anwendung zu gestatten bedarf es einer Möglichkeit solche Aufgabenbündel zu definieren und entsprechend zu kapseln. Außerdem sollen sie erweiterbar sein, um für dedizierte Anwendungstypen auch entsprechend dedizierte Aufgabenbündel hinzufügen zu können. Dazu wird zunächst der Begriff der Anwendungssicht definiert.

Definition 6-1: Eine **Anwendungssicht** stellt ein Bündel von Funktionen unter einer gegebenen Intention zusammen. Diese Intention kann dabei inhaltlich sowohl die verschiedenen Entwurfsphasen einer Web-Anwendung beschreiben, als auch Qualitätskriterien umfassen. Anwendungssichten sind paarweise unabhängig voneinander.

Eine besondere Anwendungssicht stellt die initiale Anwendungssicht dar. Sie ist immer verfügbar und ihre Intention stellt die einer Domäne primär zugeordnete Konzeption dar, also ihren eigentlichen Anwendungszweck. Sie bedarf keiner zusätzlichen Aufgabenbündel, da ihre Intention von den aufgeprägten fachlichen Komponenten erbracht wird. Beispiele weiterer Anwendungssichten sind mannigfaltig und ihre Menge ist nicht abgeschlossen. So können beispielsweise qualitätssichernde Maßnahmen, zusätzliche Entwurfsphasen oder unterstützende Funktionen in Form eigenständiger Anwendungssichten definiert werden.

In WSLS werden Anwendungssichten mit Hilfe von speziellen Dienstelementen realisiert, den »Dekorateuren«. Diese werden abhängig von einer gewählten Anwendungssicht zu einer Domäne hinzukonfiguriert und stellen ihr dedizierte – ihrer Intention entsprechende – Aufgaben zur Verfügung. Dieses Dienstelement wird konform zum Software Entwurfsmuster *Decorator* (Gamma, Helm, Johnson und Vlissides, 1995) realisiert. Einen Überblick über die konzeptionelle Beschaffenheit eines Dekorateurs gibt Abbildung 6-8.

Die Bündelung von Aufgaben in einem Dienstelement, das auf eine Domäne dekoriert wird, bildet den Ausgangspunkt für die *dienstorientierte* Nutzung der Benutzungsschnittstelle.

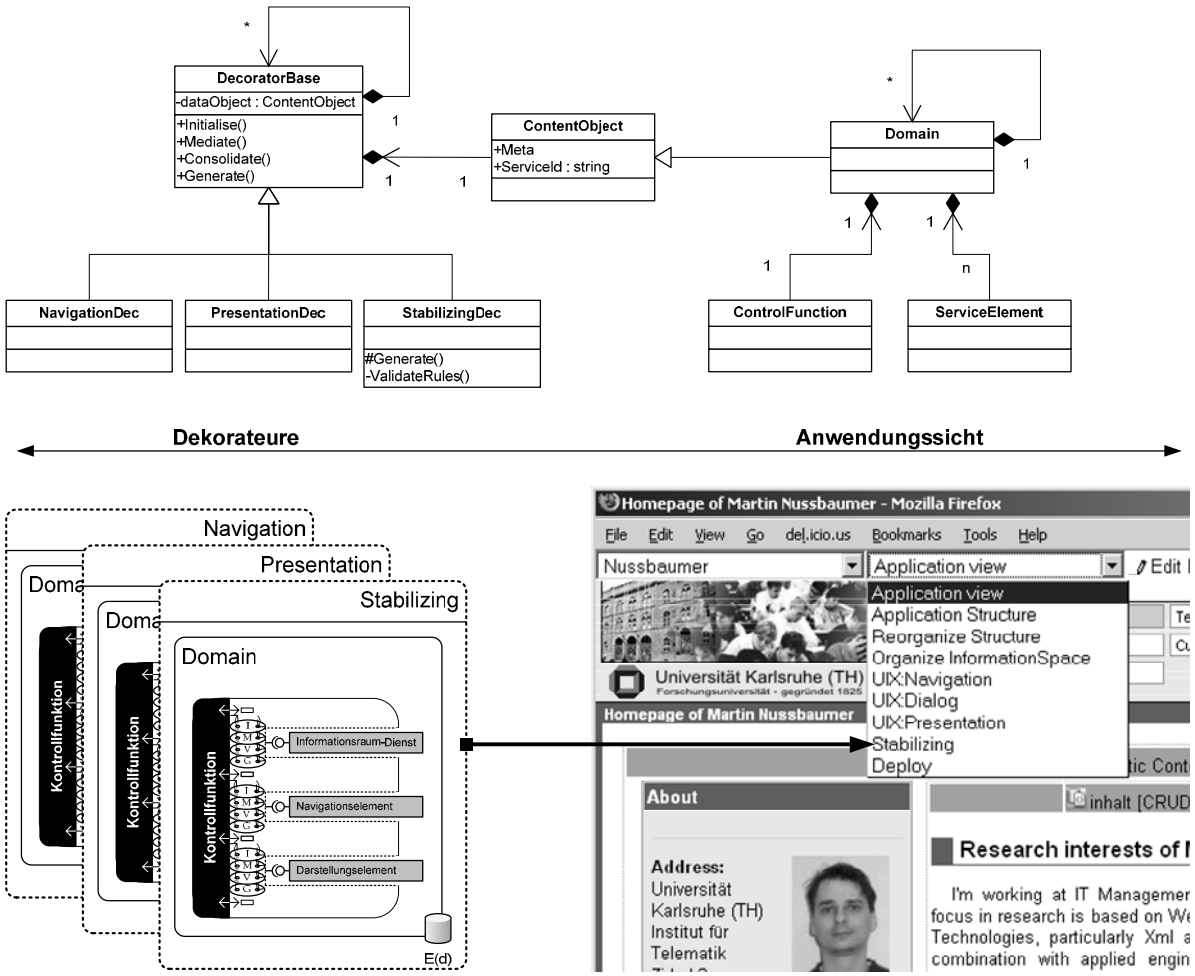


Abbildung 6-8: Zusammenspiel der Dekoration von Domänen zu dedizierten Anwendungssichten.

Dekorateure können den Lebenszyklus der WSLS-Elemente ideal nutzen, um für sie relevante Informationen und Eigenschaften aus ihrer Domäne und den aufgeprägten fachlichen Komponenten zur Erfüllung ihrer Aufgaben heranzuziehen. Hinsichtlich der »Vergesslichkeit« und der funktionalen Abkopplung bildet eine Anwendungssicht damit einen Aspekt im Sinne der Aspekt-orientierten Unterstützung der WSLS-Referenzarchitektur (vgl. Abschnitt 5.5.2). Die Quantifizierung zur Bestimmung der Dekorationsaspekte ist dynamisch ausgelegt, was bedeutet, dass sie abhängig von bestimmten Bedingungen hinzugewoben werden können. Im vorliegenden Fall heißt das, jede Anwendungssicht definiert einen Aspekt der mittels Auswahl durch einen Stakeholder hinzugewoben wird (siehe Abbildung 6-8 – Anwendungssicht).

Die Vorteile, die sich durch diese Herangehensweise ergeben, sind vielfältig. So lässt sich einerseits die Effizienz von administrativen Aufgaben verbessern, da eine Parallelisierung unterschiedlicher von einander unabhängiger Aufgaben ermöglicht wird. Dadurch lässt sich auch der Entwicklungsprozess flexibler und teilweise parallel gestalten. Außerdem können Experten besser auf die nur sie betreffenden Aufgaben verteilt werden.

Das Hinzufügen weiterer Anwendungssichten kann weitestgehend unabhängig von Anwendungskontexten, ihren Domänen und den aufgeprägten fachlichen Komponenten erfolgen. Somit herrscht zwischen den Dekorateuren und den Domänen der wünschenswerte Zustand der »Obliviousness« und eine Dekoration entspricht einem dynamisch quantifizierten As-

pekt. Einerseits können mit Hilfe von Dekorateuren Anwendungssichten unabhängig von der zu erbringenden Funktion einer Domäne entwickelt werden. Andererseits lassen sich Web-Anwendungen entwickeln ohne alle Eventualitäten unterschiedlicher administrativer Aufgaben im Vorfeld kennen zu müssen. Auf diese Weise können je nach (zukünftigen) Erfordernissen weitere notwendige Dekorationen erstellt werden.

6.2.3.1 Qualitätsgewährleistung durch Stabilisierung

Die Stabilisierung definiert eine eigene Anwendungssicht in der WSLS-Referenzarchitektur. Sie umfasst Funktionen, die die Qualität der Benutzungsschnittstelle beschreiben und messen. Dabei kann die durch die Domänen induzierte dienstorientierte Sichtweise der Benutzungsschnittstelle genutzt werden, um unterschiedliche Strategien für dedizierte Bereiche der Anwendung zu erbringen.

Barrierefreiheit als Qualitätsmerkmal

Eines der Hauptprobleme bei der Gewährleistung der Barrierefreiheit von Web-Anwendungen liegt darin, dass keine Ansätze zur Systematisierung ihrer Durchsetzung existieren (Rosson, Ballin, Rode und Toward, 2005). Vielmehr werden Werkzeuge wie Tawdis[61], Watchfire[62] oder HERA[63] verwendet, um abgeschlossene Web-Anwendungen in einer nachgelagerten Reparaturphase zu überprüfen. Diese Verfahren handeln kollektiv *reagierend,* d.h. sie sind nicht in der Lage vorhandenes Wissen über einen Anwendungsbereich in eine Überprüfung mit einzubeziehen. Entwurfsartefakte, die durch die Anwendung des *Separation-of-Concerns* Paradigmas wertvolle Hinweise auf nötige »Reparaturen« eines Anwendungsbereichs hinsichtlich der Barrierefreiheit geben könnten, sind dann in der Regel im grobgranularen Implementierungsmodell verschwunden.

Um diese Entwurfsartefakte in einen Reparaturprozess einzubinden bedarf es also einer vorgelagerten Unterstützung der Überprüfung von Web-Anwendungen in Form von *proaktiven* Verfahren.

Zur Lösung dieser Problematik wird im folgenden Abschnitt ein Formalismus zur Beschreibung automatisierbarer Richtlinien basierend auf standardisierten Web Technologien vorgestellt, der eine Anwendung innerhalb des zuvor eingeführten Rahmens einer Anwendungssicht ermöglicht.

6.2.3.2 Automatisierung der Barrierefreiheit der Benutzungsschnittstelle

Systematische Zugänglichkeit und Barrierefreiheit sollte Teil des Entwurfs sein und nicht, wie heutzutage üblich, ein Reparatur Prozess. Das bedeutet, dass Werkzeuge und Entwurfsmethodiken eine systematische Überprüfung für Web-Inhalte liefern sollten. Die unterschiedlichen staatlichen Regelwerke (ITAW, 1998; Bundesministerium der Justiz, 2002; Japanese Standards Association, 2004), die zur Überprüfung von Web-Inhalten herangezogen werden können, basieren in großen Teilen auf den Regeln des W3C.

---

[61] http://www.tawdis.net,

[62] http://webxact.watchfire.com/

[63] http://www.sidar.org/hera/

Das größte Problem mit den entsprechenden Regelwerken stellt das hohe Abtraktionsniveau dar, in dem sie verfasst wurden. So lassen sowohl der WCAG 1.0 Standard als auch der aktuelle WCAG 2.0 Draft eine Vielzahl subjektiver Interpretationsspielräume. Ein wichtiger erster Schritt stellt daher eine Kategorisierung hinsichtlich der Automatisierung dieser Regeln dar. In (Luque Centeno, Delegade Kloos, Gaedke und Nussbaumer, 2005a) wird eine Klassifikation einzelner Regeln auf Basis ihrer Implementierungskosten unter Verwendung existierender W3C Standards vorgenommen. Dabei werden sie zunächst auf das Maß ihrer Automatisierbarkeit eingestuft:

1. **Objektiv automatisierbar**: Hierunter fallen Regeln, die klar definiert sind und deren objektiver Charakter allgemein anerkannt ist. Typische Vertreter solcher Regeln sind obligatorische Elemente und Attribute einer Auszeichnungssprache, die durch eine DTD (Document Type Definition) oder ein XML Schemas ausgedrückt werden können.
2. **Subjektiv automatisierbar**: Hierunter fallen Regeln, die zwar automatisierbar sind, deren Erfüllungsgrad allerdings nur unscharf definiert werden kann. So kann beispielsweise überprüft werden, ob ein Bild eine zugehörige Bildunterschrift besitzt; die Aussagekraft einer solchen Bildunterschrift ist allerdings nur sehr schwer zu bewerten und unterliegt einer subjektiven Einschätzung.
3. **Manuell prüfbar**: Hierunter fallen all jene Regeln, die nicht verlässlich automatisiert evaluiert werden können. Ein Beispiel dafür ist die Anforderung, dass klare Navigationsmechanismen zur Verfügung gestellt werden sollen.

Während die Kategorie 1 und 2 sich gut formalisieren und damit automatisieren lassen und damit kosteneffektiv eingesetzt werden können, lassen sich Regeln aus Kategorie 3 nur manuell prüfen. Daher ist es wünschenswert, dass eine Vielzahl an Regeln durch die ersten beiden Kategorien abgedeckt wird. Regeln der Kategorie 3 wie beispielsweise das Vorhalten einer klaren und konsistenten Navigation (Richtlinie 13 der WCAG) bedürfen daher einer gesonderten Behandlung. Im Falle der Gestaltung von Dialogen oder dem Vorhalten einer konsistenten Navigation wird dies in WSLS deshalb über den gesonderten Entwurf forciert (siehe Abschnitt 6.4).

Integration des Formalismus in den Entwurfsprozess mittels Dekoration

In einem Kooperationsprojekt mit der Universität Carlos III in Madrid wurde im Rahmen dieser Arbeit ein formales Modell basierend auf XML-basierten Technologien entwickelt, das eine teilweise automatische Überprüfung von Web-Inhalten ermöglicht. Dabei wurde der Fokus gezielt auf den Kompositionscharakter einer Web-Anwendung gelegt (Luque Centeno, Delegade Kloos, Gaedke und Nussbaumer, 2005b). Ein Beispiel für solche Regeln basierend auf dem XPath Sprache (Berglund, Boag, Chamberlin, Fernández et al., 2007) sind in Listing 6-1 aufgestellt. In beiden Fällen wird ein notwendiges Attribut *alt*[64] für die definierenden Elemente *img* und *input* gefordert. Hierbei kann jedoch lediglich die Existenz des Attributs gefordert werden; über die Qualität des Inhalts lässt sich so keine Aussage treffen.

[64] Vom Englischen »alternate text«, das zusätzlich zur visuellen Aussage eine alternative Beschreibung für ein Bild bereitstellt.

```
01.  //input[@type="image"][not(@alt)]
02.  //img[not(@alt)]
```

Listing 6-1: Zwei XPath Ausdrücke zur Beschreibung notwendiger Attribute.

Um methodisch Hilfestellungen über möglicherweise relevante Inhalte von Attributbelegungen wie denen in Listing 6-1 vorgestellten zu geben, bedarf es des feingranularen Zugriffs auf die unterschiedliche Einflussgrößen, die auf diesen Inhalt wirken. Eine solch feingranulare Zugriffsstruktur wird in WSLS durch die Domänen erreicht, die den Zugriff auf die ihnen aufgeprägten fachlichen Komponenten ermöglichen.

Die technische Überprüfung und Durchsetzung wird in einem Dekorateur umgesetzt, der dazu die formalisierten Regeln auf Basis eines einzigen XSL(T) Programmes (Clark, 1999) zusammenfasst. Eine umfassende Beschreibung findet sich in (Centeno, Kloos, Gaedke und Nussbaumer, 2006). Der Dekorateur übernimmt dabei die entsprechende Parametrisierung des XSL(T) Programmes mit Hilfe der aufgeprägten fachlichen Komponenten. Dadurch lassen sich Domänen unter Zuhilfenahme ihrer Konfiguration dediziert anpassen und ermöglichen statt einer heute üblichen »reagierenden« Validitätsprüfung, zusätzlich eine »proaktive Zugänglichkeit«.

Abbildung 6-9 stellt die Validierungsunterstützung in WSLS mit Hilfe der Anwendungssicht »Stabilisierung« dar. Dabei wurde das Anwendungsbeispiel aus dem vorigen Abschnitt zu Grunde gelegt (vergleiche Abbildung 6-2). Die unterschiedlichen Domänen werden entsprechend der Stabilisierungsphase in einer eigenen Anwendungssicht dekoriert. Dadurch wird die Durchsetzung von Richtlinien zur Gewährleistung von Barrierefreiheit zu einer Managementaufgabe innerhalb dieser dedizierten Anwendungssicht. So lässt sich die Entwicklung von Anwendungskontexten aus Domänen und deren Überprüfung voneinander trennen und beispielsweise mit unterschiedlichen Rollen durchführen.

Wie in der Abbildung dargestellt lassen sich Überprüfungen dediziert auf Domänenbasis durchführen. Eine Überprüfung kann dabei mit zusätzlichen Parametern konfiguriert werden, etwa welchem Konformitätsgrad das Ergebnis entsprechen soll. Die in den WCAG Richtlinien definierten Konformitätsgrade umfassen dabei:

- A: alle Regeln der Priorität 1 erfüllt,
- AA: alle Regeln der Priorität 1+2 erfüllt und
- AAA: alle Regeln der Priorität 1-3 erfüllt.

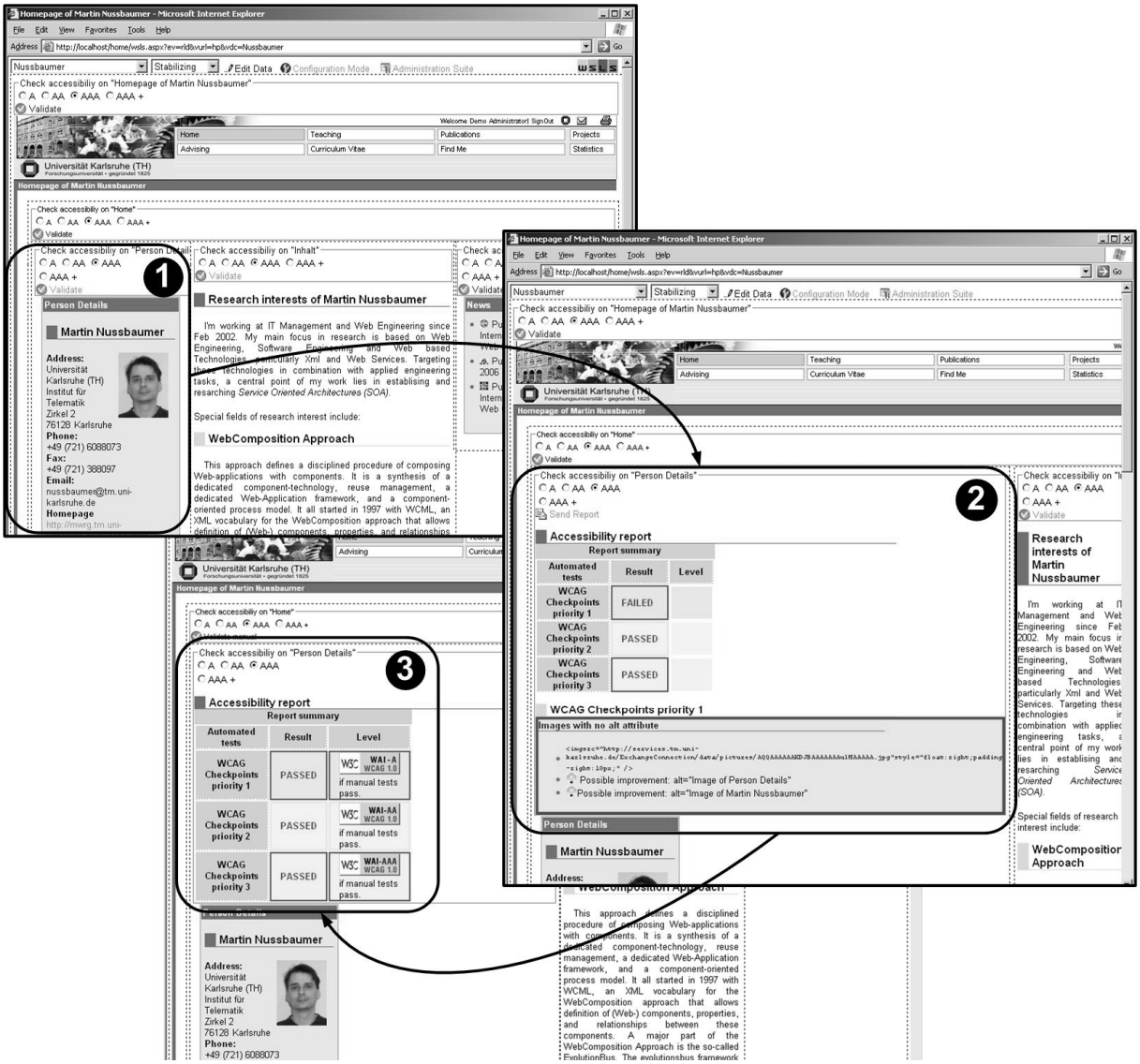


Abbildung 6-9: Die einzelnen Validierungsschritte in der Phase der Stabilisierung einer Web-Anwendung auf Basis der lose gekoppelten Domänendienste. Die Methodik wurde in der WSLS-Referenzarchitektur umgesetzt und demonstriert, wie ein Dienst mit proaktiver Unterstützung auf das Regelwerk für Barrierefreiheit des W3C (WCAG) geprüft werden kann.

Die in der Abbildung vorgestellten Schritte umfassen die Anwendung (1), Entdeckung (2) und Reparatur (3) einer Verletzung der WCAG-Regel *7.1 »Short text equivalents for images«*. In Schritt 1 wird die Überprüfung für die dargestellte Domäne initiiert. Zu diesem Zeitpunkt verstößt das dargestellte Bild gegen die Regel 7.1, es stellt also keine alternative Beschreibung zur Verfügung. In Schritt 2 wird das Ergebnis der Validitätsprüfung präsentiert, der Regelverstoß dokumentiert und zusätzlich, unter Zuhilfenahme des Domänenwissens, plausible Vorschläge für die Reparatur gemacht. Aufgrund einer offenen Schnittstelle zum XSL(T) Programm auf Basis von Parametern, lassen sich auf diese Weise sukzessive Erweiterungen hin zu einer Anbindung unterschiedlicher Heuristiken betreiben. Die Parametrisierung im vorliegenden Fall wird mit Hilfe des Zugriffs auf das entsprechende Informationsraum-Dienstelement erreicht, das Personen-basierte Datenobjekte abgeleitet vom ContentObject zur Verfügung stellt. Hier wird der Umstand ausgenutzt, dass jedes dieser Objekte über die DublinCore Metadaten verfügt (vergleiche Forderung 4-4 in Abschnitt 4.2.2 des GTS), die dann in einem entsprechenden XSL-Parameter an die Validierungsunterstützung weiterge-

geben werden können. Schritt 3 stellt anschließend den Zustand nach erfolgter Reparatur dar. Im vorliegenden Fall wurde das Darstellungsdienstelement entsprechend angepasst.

6.3 Anwendungssicht Präsentation

Die Forderung nach »Trennung von Inhalt und Design (Präsentation)« ist eines der meistzitierten Mantras im Zusammenhang mit dem Entwurf von Web-Anwendungen in Verbindung mit Cascading Stylesheets (Bos, Çelik, Hickson und Lie, 1998). Von dieser Trennung verspricht man sich gravierende Vorteile bei der Verarbeitung von elektronischen Dokumenten. Diese resultieren im Wesentlichen aus:

- **Isolation des Inhalts**: Ermöglicht eine effiziente Verwaltung des Inhalts in Content-Management-Systemen (CMS) oder aus generierenden Komponenten wie der WSLS-Referenzarchitektur.
- **Isolation der Präsentation**: Reduziert den Aufwand bei auftretenden Änderungen am Präsentations-Design, sowohl bei kleinen Änderungen als auch bei einer kompletten Neugestaltung.

Eine bloße Separierung in Auszeichnungssprache (engl. Markup Language)[65] und Formatierungssprache[66] stellt zwar eine *notwendige* aber nicht *hinreichende* Anforderung dar. Die Darstellung, insbesondere die Ästhetische, umfasst mehr Dimensionen als nur die Formatierung von Inhalten mit verschiedenen Schriften und Farben. So sollte ein Präsentationsdesign nicht nur in Bezug auf unterschiedliche Ausgabemedien so flexibel wie möglich sein, sondern auch in Bezug auf die Inhalte, die durch eine hohe Dynamik gekennzeichnet sind (Brewington und Cybenko, 2000). Ein intuitives Verständnis von »Inhalt« und »Präsentation« findet sich in der Begriffserklärung in den »Zugänglichkeitsrichtlinien für Web-Inhalte 1.0« (Chisholm, Vanderheiden und Jacobs, 1999) der W3C-Organisation.

Inhalt (W3C): »Als Inhalt eines Dokuments wird das bezeichnet, was das Dokument dem Benutzer durch natürliche Sprache, Bilder, Ton, Filme, Animationen usw. mitteilt.“

Struktur eines Dokuments (W3C): »Die Struktur ist sein logischer Aufbau (z. B. nach Kapiteln, mit einer Einführung und einem Inhaltsverzeichnis usw.). Ein Element (z. B. P, STRONG, BLOCKQUOTE in HTML), das Struktur spezifiziert, wird Struktur-Element genannt.«

Präsentation (W3C): »Die Präsentation eines Dokuments ist die Art seiner Darstellung (z.B. als Ausdruck, als zweidimensionale grafische Präsentation, als Text-Präsentation, als synthetisierte Sprache, in Blindenschrift usw.)«

---

[65] Eine **Auszeichnungssprache** ist eine Sprache, die es erlaubt, Daten mit maschinenlesbaren Markierungen zu versehen. Beispiele solcher Sprachen sind XML, SGML und XHTML.

[66] In einer **Formatierungssprache** können Regeln für die Darstellung von Elementen einer Auszeichnungssprache definiert werden.

Die vorgestellte Aufteilung erscheint auf den ersten Blick schlüssig. In der Praxis lässt sich allerdings ein Teil des Markup *keinem* der drei Bereiche zuordnen. Insbesondere Elemente zur Positionierung (*div* und *span* tags), Identifikatoren (*id* Attribute) sowie CSS-Klassenzuweisungen (über das class Attribut), die der Festlegung eines Layouts dienen, fallen weder unter »Präsentation« und »Inhalt», noch unter die „Struktur“ — bezeichnet letztere ja lediglich den *logischen* Aufbau des Dokumentes, also eine rein semantische Untergliederung. Gerade diese Elemente sind aber von zentraler Bedeutung für das Zusammenwirken von Inhalt und Präsentation hin zu einem ästhetischen Gesamtbild.

Abbildung 6-10 zeigt den Zusammenhang zwischen den möglichen Bestandteilen eines Dokuments, gruppiert in Inhalt, Struktur, maschinelle Verarbeitung, Positionierung, und grafischer Repräsentation. Zusätzlich wird die Ebene der Perzeption eingeführt, welche die Verarbeitung durch Mensch und Maschine beschreibt. Dem gegenüber wird die Zuordnung des W3C gestellt. Dabei fällt auf, dass weder der Bereich Informationen für maschinelle Verarbeitung noch die Inhalts- und Darstellungsverknüpfenden Elemente (Positionierung) ausreichend Berücksichtigung finden (vgl. Spalte 3 und 4 der Abbildung). Diese Schwäche lässt sich besonders deutlich daran erkennen, dass zur Position von Inhalten in Web-Anwendungen nach wie vor überwiegend Tabellen eingesetzt werden (siehe Abschnitt 6.1). Diese erzwingen bei der grafischen Repräsentation durch einen Browser eine positionierende Strukturierung der darzustellenden Inhalte. Die dabei entstehenden Probleme sind mannigfaltig. Ein wichtiges Argument gegen die Verwendung von Tabellen findet sich hinsichtlich der Barrierefreiheit. Die Reihenfolge der Seitenelemente im Quelltext hängt von der gewünschten Darstellung ab und entspricht nicht der logischen Lesereihenfolge[67].

[67] Aufgrund der Positionierung von Inhalten durch eine Tabelle wird die Priorisierung der Inhalte automatisch von links nach rechts gelegt. Das steht im Kontrast zu den zumeist in der Mitte angebrachten relevanten Informationen, während links oftmals Verweise angebracht sind. Ein ScreenReader, der die Inhalte für Blinde und Sehgeschädigte vorliest, wird dann konsequenterweise zunächst diese Verweise berücksichtigen und danach erst zu den relevanten Informationen kommen.

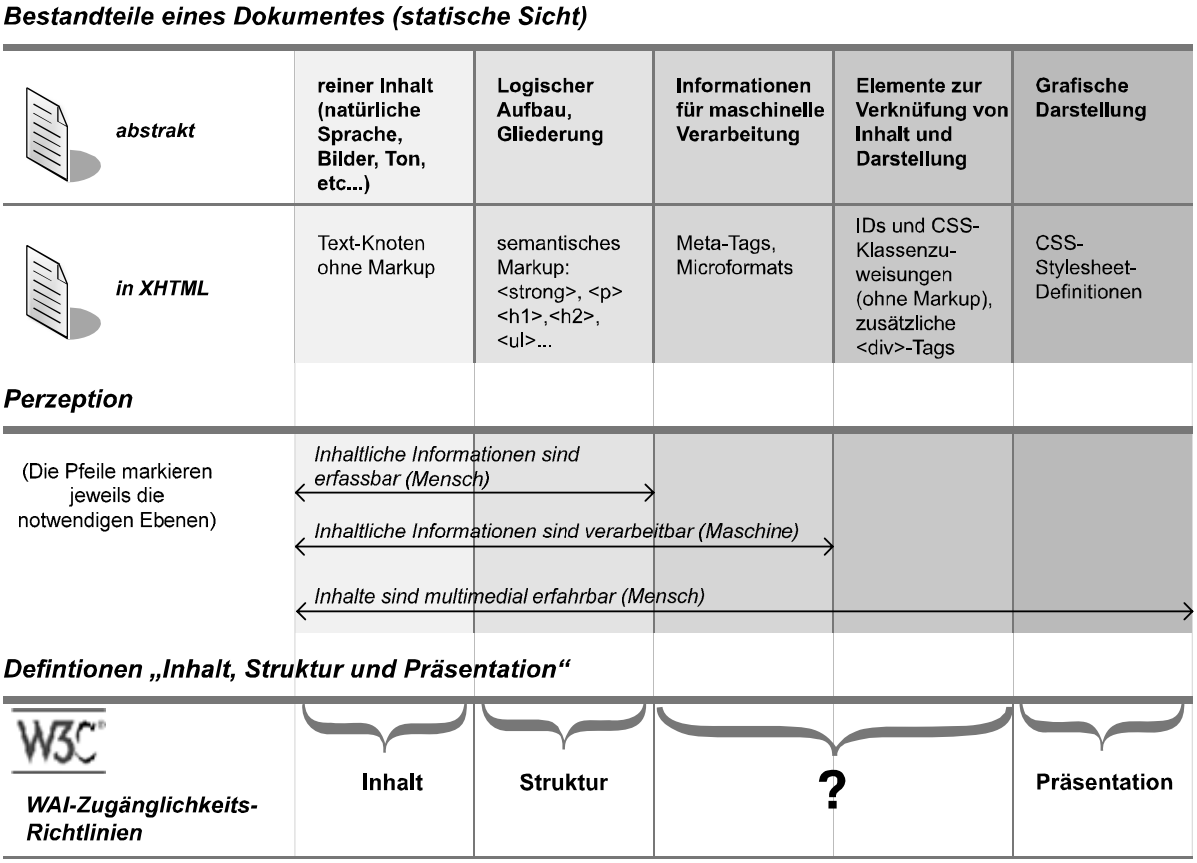


Abbildung 6-10: Bestandteile eines Dokuments nach der Definition des W3C.

Um die in der Abbildung dargestellte Lücke in der W3C-Definition zu schließen, wird im Folgenden eine Erweiterung der Struktur entwickelt. Dazu werden mit der s-Struktur und der p-Struktur zwei klar abgegrenzte Struktur-Typen definiert. Das Modell der s- und p-Struktur dient als Ausgangspunkt für die Lösung der Trennung zwischen Inhalt und Darstellung.

Im Folgenden wird, wenn nicht anders vermerkt die Auszeichnungssprache und deren Sprachelemente mit XHTML (Pemberton, Altheim, Austin, Boumphrey et al., 2000) identifiziert und als Formatierungssprache der CSS Standard verwendet.

Die **s-Struktur** umfasst die inhaltlich-logische Struktur eines Dokumentes. Semantiktragende Auszeichnungselemente, die inhaltliche Informationen einer elektronischen Auswertung zugänglich machen, etwa die Meta-Angaben im XHTML-Kopf oder Mikroformate (Suda, 2006), bilden die s-Struktur.

Die **p-Struktur** umfasst alle Elemente, die weder dem s-strukturierten Inhalt, noch der Präsentation zuzuordnen sind. Die p-Struktur hat die Aufgabe s-Struktur und Präsentation logisch miteinander zu verbinden. Hierzu zählen beispielsweise CSS-Klassenzuweisungen und Positionierungselemente (div und span tags).

Das folgende Beispiel soll den Unterschied zwischen der p- und s-Struktur verdeutlichen. Listing 6-2 stellt das Mikroformat *hCard* vor, das ein webbasiertes Analogon zum vCard Standard bereitstellt (Dawson und Howes, 1998). Hierbei werden CSS Klassenzuweisungen verwendet, um den Inhalt logisch zu strukturieren. Aus diesem Grund zählen so geartete Auszeichnungselemente auch zur s-Struktur und nicht zur p-Struktur.

```
01.  <div class="vcard">
02.      <div class="fn">Martin Nussbaumer</div>
03.      <div class="org">University of Karlsruhe (TH)</div>
04.      <div class="tel">+49 721 6808073</div>
05.   </div>
```

Listing 6-2: s-Strukturierte Personenbeschreibung am Beispiel des Mikroformats hCard.

Die p-Struktur ist notwendig, um eine komplexe, ästhetisch ansprechende Benutzungsschnittstelle umzusetzen, da s-strukturierter Inhalt in der Regel zu wenig Ankerpunkte für die Anwendung unterschiedlicher Präsentationsformate bietet. Inhalt und Präsentation lassen sich also nie vollkommen voneinander trennen. Der Inhalt muss in einer technischen Umsetzung mit Elementen der p-Struktur angereichert werden, die über die rein logische Strukturierung hinausgeht und die passenden Ansatzpunkte für eine dedizierte Design-Definition zur Verfügung stellt. Das Zusammenspiel der s- und p-Struktur und ihr Wirkungsgrad auf die eingangs vorgestellte Problematik der Perzeption ist in Abbildung 6-11 dargestellt.

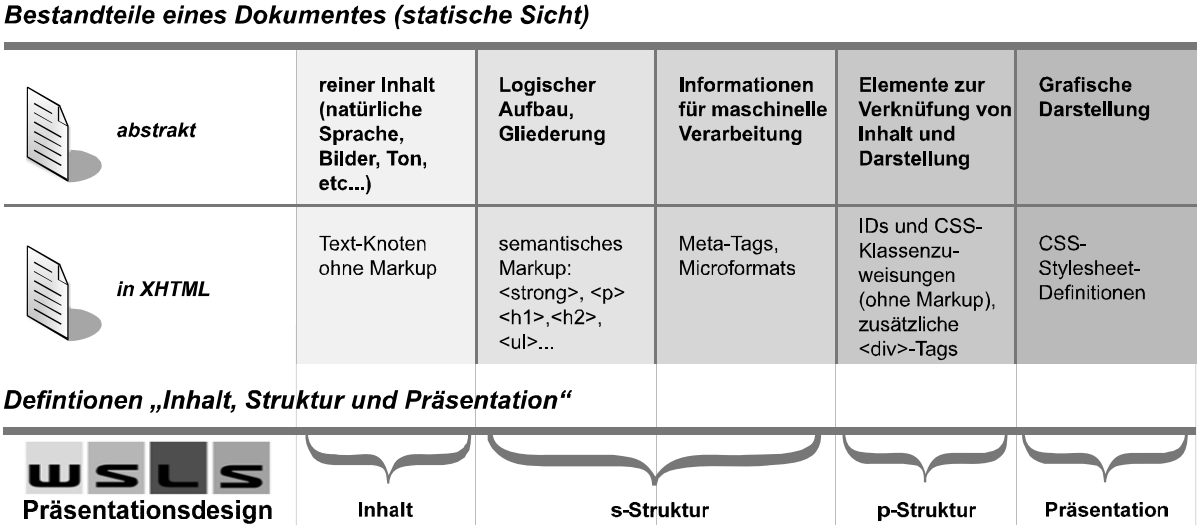


Abbildung 6-11: Bestandteile eines Dokumentes aus Sicht des WSLS Präsentationsdesigns mit der Erweiterung der s- und p-Struktur.

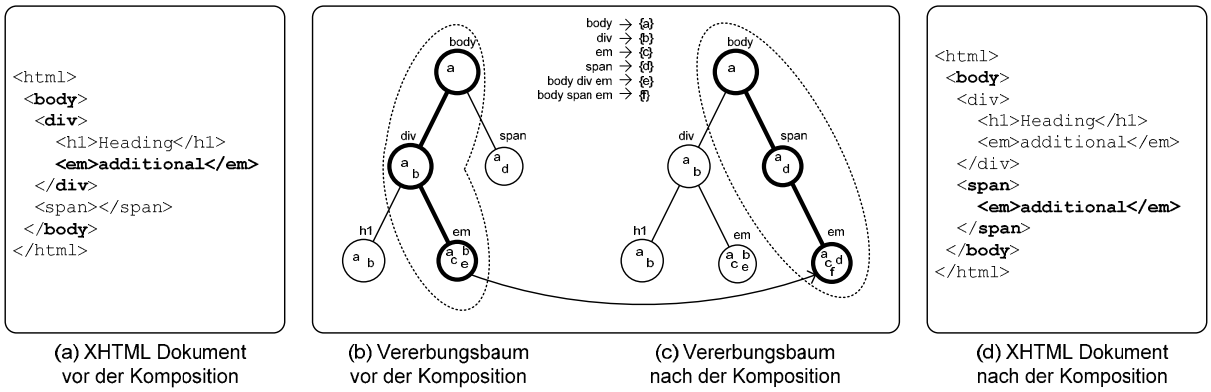
Der Nutzen von s- und p-Struktur für das komponentenbasierte Anwendungsmodell der WSLS-Referenzarchitektur wird in den folgenden Abschnitte ausführlich dargelegt.

6.3.1 Einbettung in das WSLS Komponentenmodell

Ein wesentliches Merkmal einer Domäne in ihrer Eigenschaft als konzeptuelle Komponente bildet die Abgeschlossenheit. Sie ist durch eine eindeutig definierte Schnittstelle als Black-Box wiederverwendbar. Dadurch wird eine effiziente Konstruktion von Web-Anwendungen nach dem Baukasten-Prinzip ermöglicht. Die Effizienz der Integration einer solchen Komponente in eine Web-Anwendung hängt dann neben dem Aufwand für ihre funktionale Integration entscheidend vom Aufwand für die grafische Einbettung in die Benutzungsschnittstel-

le der Web-Anwendung ab. Der von einer Komponente emittierte Client-Code (initiiert durch die Phase *Generieren* im Lebenszyklus) kann aus beliebig komplexen Fragmenten der Auszeichnungssprache bestehen. Die erzeugte Web-Anwendung stellt sich in ihrer Ganzheit in Form eines DOM-Baumes (Dokument-Objekt-Modell) dar; die einzelnen Fragmente bilden Knoten in diesem Baum.

Da die Formatsprache CSS eine Zuweisung von Formatregeln durch Mustersuche (*pattern matching*) mit Hilfe von Selektoren (Knoten oder Pfade im DOM-Baum) realisiert, ist die Herausnahme einer Komponente aus einem speziellen Darstellungskontext problematisch. Umsomehr, wenn durch die Hinzunahme neuer Komponenten eine gegenseitige Beeinflussung solcher Muster auftreten kann. Das bedeutet, dass durch Wiederverwendung und Komposition einer Komponente ein neuer, in der Regel durch die Komponente unplanbarer, grafischer Kontext entstehen kann. Abbildung 6-12 stellt eine solche Situation dar.



(a) XHTML Dokument vor der Komposition | (b) Vererbungsbaum vor der Komposition | (c) Vererbungsbaum nach der Komposition | (d) XHTML Dokument nach der Komposition

Abbildung 6-12: Anwendung der CSS Formate durch Pattern Matching auf dem DOM.

Das Beispiel zeigt das Ausgangsdokument (a), das in seiner Struktur als Baum in (b) abgebildet wird. Die einzelnen Knoten im Baum werden als Komponenten[68] aufgefasst. Der Knoten *em* sei durch eine Komponente K_{em} erzeugt worden und in einer Komponente K_{div} verwendet. Die Komponente K_{em} soll nun in Komponente K_{span} wiederverwendet werden. Das Ergebnis dieser Wiederverwendung wird in (c) als Baum dargestellt, das zugehörige Dokument in (d).

Die in den Knoten dargestellten Buchstabenkombinationen aus $\{a, b, c, d, e, f\}$ symbolisieren die zutreffenden Formatangaben, die auf diesen Knoten Anwendung finden. Deren Formatierungsregeln sind in CSS verfasst und finden sich in der Abbildung Mitte. Deutlich zu erkennen sind die unterschiedlichen grafischen Kontexte des Knotens *em* in den Pfaden *body→div→em* = $\{a, b, c, e\}$ und *body→span→em* = $\{a, b, c, d, e, f\}$.

Allgemeine Lösungsansätze wie Maximale Spezifizität (Lawver, 2006) oder die eindeutige Identifizierung greifen zu kurz, da sie lediglich für eine Darstellungsproblematik ausgelegt sind, die *nicht* auf einer Wiederverwendung von Darstellungskomponenten beruht. Eine eindeutige Identifikation verbietet sich sogar, da Identifikatoren innerhalb eines DOM-Baums,

[68] Im abgebildeten Beispiel ist eine Komponente durch einen einzelnen Knoten repräsentiert. Dieser Umstand dient der verbesserten Lesbarkeit. Prinzipiell können Komponenten beliebig komplex aufgebaut sein und dementsprechend mehrere Knoten umfassen.

das entspricht einer dargestellten XHTML-Seite, nur einmal vergeben werden dürfen und so eine Wiederverwendung innerhalb dieses DOM-Baumes kategorisch ausgeschlossen wird.

Eine Lösung muss also vielmehr in einer gesamtheitlichen Sicht auf eine Web-Anwendung und ihrer Komponenten – wie im Abschnitt 6.2 vorgestellt – erfolgen. Es muss einerseits die Struktur der durch die Komponenten induzierten Anwendung Berücksichtigung finden, als auch deren Abgeschlossenheit beachtet werden. Eine solche Lösung wird im nächsten Abschnitt mit Hilfe von dedizierten Namensräumen vorgestellt.

6.3.1.1 Entwicklung einer Struktur-Schnittstelle durch Namensräume

Aus logischer Sicht lassen sich Inhalt und Präsentation trennen, indem man die s-Struktur dem Inhalt und die p-Struktur der Präsentation zuordnet. Entscheidend bei der Verarbeitung und Pflege der beiden Bestandteile ist aber deren *physische* Trennung. Eine Web-Anwendung lässt sich zwar reduziert auf ihre Darstellung in ein Dokument der Auszeichnungssprache (etwa XHTML) und ein Dokument der Formatierungssprache (etwa CSS) physisch aufspalten. Diese Trennung überschneidet sich jedoch mit der logischen Separierung Inhalt/Präsentation, wenn das Dokument in der Auszeichnungssprache p-Strukturen enthält.

Eine solch feingranulare Trennung ist daher in einem grobgranularen und dokumentengetriebenen Präsentationsdesign, wie etwa dem von WebML (siehe Abschnitt 3.2.2.2), nicht möglich.

Die Domänen-Graph-Repräsentation der Anwendung hat hier den entscheidenden Vorteil, dass innerhalb der Anwendung Bereiche identifiziert werden können, die dann der eingangs eingeführten s- und p-Struktur zuordenbar sind. Domänen im Visualisierungsbaum können durch zusätzliche Elemente der p-Struktur angereichert werden, die dann Namensräume in Form logischer Pfade aufspannen und so das Zusammenwirken zwischen Auszeichnungs- und Formatierungssprache erlauben. Abbildung 6-13 verdeutlicht dieses Zusammenwirken über die s- und p-Struktur-Schnittstelle. Während die s-Struktur-Schnittstelle aufgrund ihres wohldefinierten Vokabulars (zum Beispiel entsprechende XHTML-Module) »schmal« gehalten werden kann, erstreckt sich die p-Struktur-Schnittstelle über die ganze Breite des Dokuments.

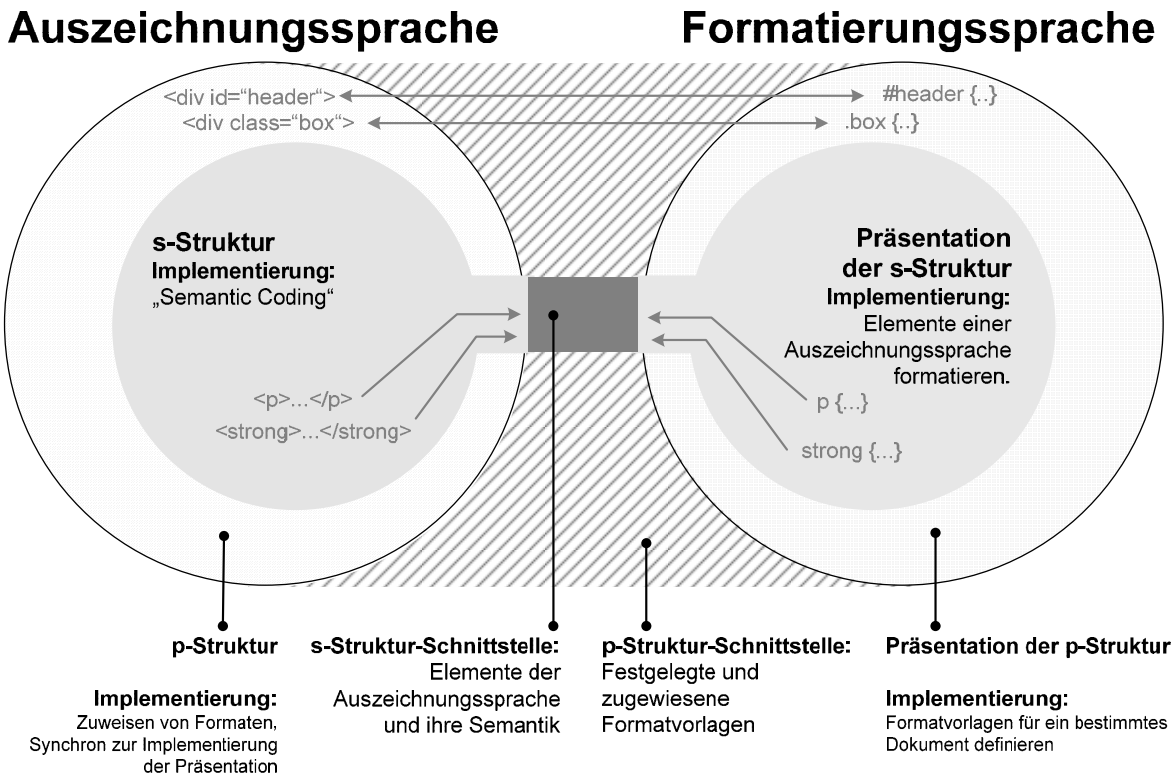


Abbildung 6-13: Zusammenwirken von s- und p-Struktur-Schnittstelle mit XHTML und CSS.

Die **s-Struktur-Schnittstelle** folgt den Standardisierungsbemühungen des W3C und ist dementsprechend eindeutig durch eine Document-Type-Definition (DTD) definiert. Aus Sicht des Content bedeutet dies, dass die s-Struktur unmittelbar aus der Semantik des Inhalts hervorgeht. Es sind also auch keinerlei Entscheidungen darüber zu treffen, *wie* der Inhalt dargestellt wird, sondern ausschließlich, *welche* semantische Bedeutung den ausgezeichneten Daten zukommt (z. B. Überschrift oder Gruppierung).

Die **p-Struktur-Schnittstelle** ist – im Gegensatz zur s-Struktur-Schnittstelle – offen und frei gestaltbar und durch keine Definition festgelegt. Diese Offenheit bietet die nötige Flexibilität zur Definition von ästhetisch ansprechenden Entwürfen. Die Aufgabe der p-Struktur-Schnittstelle ist es nicht diese Ästhetik zu gewährleisten. Ihre Aufgabe besteht vielmehr in der Sicherstellung effektiver Mechanismen zu Ermöglichung eines ästhetischen Entwurfs einerseits, sowie effizienter Verfahren zu dessen Wiederverwendung andererseits.

Wann immer möglich, sollte die s-Struktur-Schnittstelle bevorzugt werden. Das bedeutet, dass für die Inhalte die Möglichkeiten des semantischen Markup voll ausgeschöpft werden können. Dieser Vorgang wird oft auch als »semantic coding« bezeichnet. Semantischer Code ist als Meta-Information über die Bedeutung und logische Struktur von Inhalten zu verstehen und daher zunächst nicht Präsentations-gebunden. Im Falle von XHTML können Auszeichnungselemente in der vom W3C vorgeschriebenen Art und Weise verwendet und eingesetzt, um ein Höchstmaß an semantischen Daten als Informationen zu kommunizieren. Neben den Vorteilen von semantischem Code bei der Darstellung mit CSS, sprechen eine Reihe von weiteren Argumenten für die bedeutungsvolle Auszeichnung. So lassen sich Arbeitsprozesse und Abstimmungsrunden in großen Projektteams beschleunigen, da lediglich semantisch annotierte Inhalte zur Verfügung gestellt werden. Der Markup ist stringent, leichter lesbar und fast selbsterklärend. Dadurch sind Änderungen schneller und leichter durchführbar, wodurch nicht zuletzt auch die Wiederverwendung erleichtert wird. Gerade hinsichtlich der Syndikati-

on von Inhalten durch Austauschformate wie RSS oder ATOM bietet eine klare Trennung Vorteile für eine effiziente Anwendung (Trefz, 2004). Außerdem lassen sich Anforderungen zur Barrierefreiheit leichter umsetzen, indem einerseits eine klare Trennung von semantisch annotierten Inhalten und deren Darstellung erfolgt. Die »obligatorische Textversion« vieler Web-Anwendungen kann dann durch ein gutes Präsentationsdesign ersetzt werden, was Ressourcen spart.

Ein weiterer Vorteil der Einführung einer p-Struktur-Schnittstelle stellt die Fähigkeit zur Verallgemeinerung eines Präsentationsdesigns dar. So können bei *ähnlich* strukturierten Web-Anwendungen Entwürfe in großen Teilen wiederverwendet werden oder sogar gezielt verschiedene Darstellungen vorgehalten werden (*skinning*). Gerade Änderungen im Corporate Design (CD) von Organisation lassen sich so effizient und kosteneffektiv durchführen. Aber auch in Hinblick auf Föderationen von Organisationen, die ihre (webbasierten) Dienstleistungen Dritten anbieten, ist es erforderlich, ein Darstellungsdesign effizient und effektiv an deren Designvorgaben anpassen zu können.

Unterstützung des Computer Aided Web Engineering (CAWE)

Das Modell der s- und p-Struktur-Schnittstelle fördert die Definition eines unterstützenden Editors und bildet so die Grundlage für ein Werkzeug im Sinne des Computer-Aided Web Engineering (CAWE). Dieses Werkzeug kann dann, basierend auf dem Visualisierungsbaum, dediziert Domänen über entsprechende Namensräume erreichen und verschiedene Präsentationen vorhalten. Domänendienste (Blätter im Visualisierungsbaum) stellen grundsätzlich s-Struktur-Inhalte zur Verfügung. Strukturierungsdomänen (Knoten im Visualisierungsbaum) definieren darüber hinaus zusätzliche Positionierungseigenschaften, die als Elemente der p-Struktur die Anordnung ihrer Kinder beschreiben können. Die Anordnung von Kindern einer Strukturierungsdomäne wird über spezielle Darstellungselemente erreicht, deren Aufgabe lediglich in deren Positionierung besteht.

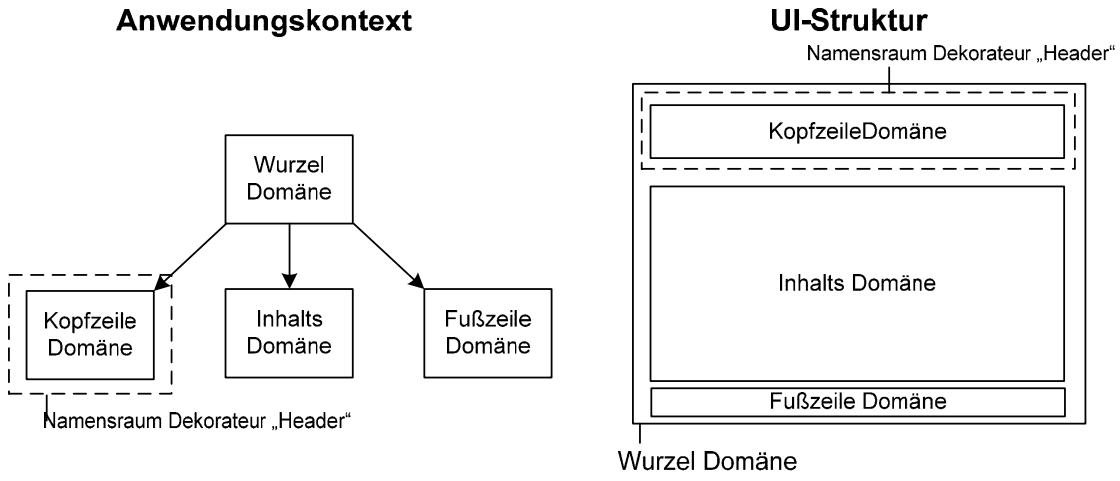


Abbildung 6-14: Umsetzung des Konzepts der Namensräume durch die p-Struktur-Schnittstelle.

Die selektive Zuweisung von CSS Formaten zu den s-Strukturen der Domänendienste wird anhand logischer Pfade im Visualisierungsbaum vorgenommen. Die dafür nötigen Selektoren werden durch Namensräume über diese logischen Pfade identifiziert. Jeder Namensraum

entspricht dabei einem Selektor. Das Vorhalten einzelner Namensräume wird durch Dekorateure der Anwendungssicht Darstellung erreicht. Die durch diese Dekorateure aufspannbaren Namensräume definieren in ihrer Gesamtheit die p-Struktur-Schnittstelle einer Web-Anwendung in WSLS. Abbildung 6-14 demonstriert die Definition von Namensraum-Dekorateuren auf Basis des Visualisierungsbaums.

WSLS Namensraum-Dekorateure

Die WSLS Suite stellt eine Basismenge vordefinierter Namensraum-Dekorateure in der p-Struktur-Schnittstelle zur Verfügung, um die Wiederverwendung unterschiedlicher Darstellungsentwürfe zu vereinfachen. Diese Grundmenge umfasst die üblicherweise in Web-Anwendungen vorgefundenen Strukturierungselemente wie:

- Header (Seitenkopf)
- Navigation (Navigation)
- Content (Inhalt)
- Additional Content (nebengeordnete Inhalte z.B. am rechten Rand)
- Footer (Seitenfuß)

Abbildung 6-15:Angepasstes Layout in der Anwendungssicht »Präsentation«. Jede Domäne des Visualisierungsbaumes kann mit Hilfe eines Namensraum-Dekorateurs angepasst werden.

Weiterentwicklungen dieser Grundmenge lassen sich abhängig von verschiedenen Faktoren leiten. So können einerseits Richtlinien eines Corporate Design (CD) einer Organisation herangezogen werden, um eine gezielte Unterstützung zu erreichen (vgl. Abbildung 6-15). So sind auch empirische Analysen einer Reihe von Web-Anwendungen möglich, um eine gezielte Erweiterung der p-Struktur in Richtung des Darstellungsentwurfes dieser Anwendungen zu forcieren. Auch das zum Zeitpunkt dieser Arbeit noch im Draft-Zustand befindliche XHTML *Role Access Module* (Birbeck, McCarron, Pemberton, Raman et al., 2006) bietet eine Unter-

teilung von Seiteninhalten an, die eine sinnvolle Ergänzung der p-Struktur-Schnittstelle darstellt.

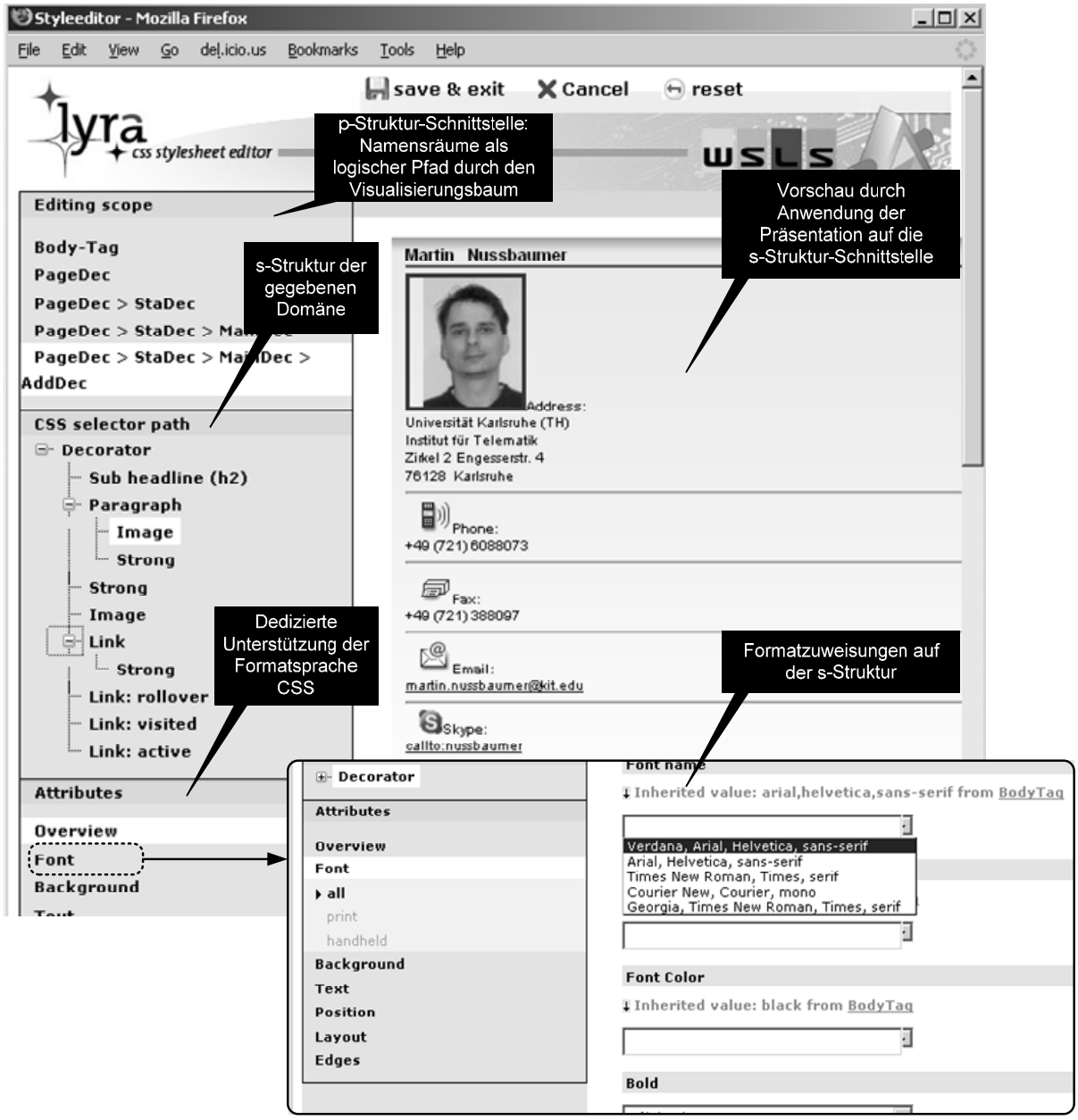


Abbildung 6-16: Ein Werkzeug zur Bearbeitung der Präsentation eines Knotens des Visualisierungsbaums. Hier wird ein über die p-Struktur-Schnittstelle identifizierter Domänendienst bearbeitet.

6.4 Anwendungssicht Navigation

Die Aufgabe, Informationen aus verschiedenen Informationsquellen Endnutzern passend zu den jeweiligen Anforderungen und Kontexten zur Verfügung zu stellen, stellt eine große Herausforderung dar. Bei der Einbindung solcher Informationsquellen finden in zunehmendem Maß dienstorientierte Architekturen, insbesondere Web Service Technologien, Einsatz. Mit ihrer Hilfe lassen sich Datenbanken und andere Softwaresysteme durch eine lose Kopplung

über Organisationsgrenzen hinweg plattformunabhängig auf eine föderative Art und Weise zusammenschließen. In diesem Zusammenhang ergibt sich für das Web Engineering die wichtige Aufgabe, geeignete Mittel zur Modellierung und Realisierung von *Navigationsmöglichkeiten* zu Verfügung zu stellen, mit deren Hilfe der Benutzer den zusammengeführten Informationsraum erschließen kann.

Es existiert zwar bereits eine Vielzahl von Ansätzen und Methoden, die sich gezielt mit der Navigation im Web auseinander setzen (vgl. hierzu Abschnitt 3.2) und zum großen Teil ihren Ursprung in der Erforschung von Hypertextsystemen haben. Bei ihrer Anwendung auf moderne Einsatzszenarien zeigen sie jedoch Defizite im Sinne einer mangelnden Berücksichtigung von autonomen Diensten als Informationsquellen auf. Problemstellungen ergeben sich hierbei insbesondere aus der Unkontrollierbarkeit fremder Dienste, deren Daten sich auf einer semantischen Ebene verknüpfen lassen, *ohne* dass Dienstanbieter dieses bei der Bereitstellung gezielt beachten müssen. Darüber hinaus erfordert die Evolution des Anwendungsumfelds geeignete Mittel zum Umgang mit der dynamischen Einbindung neuer Dienste, dem Entfernen von nicht mehr zu Verfügung stehenden Diensten bzw. dem Ersetzen durch andere äquivalente Dienste.

Um die geforderte Orthogonalität zwischen dem Informationsraum mit seinen autonomen Diensten und der zu unterstützenden Navigation auf Anwendungsebene zu ermöglichen, bedarf es einer von den erzeugenden Datenquellen unabhängigen Navigationsmodellierung. Für einen unterstützenden Navigationsentwurf ist also vorwiegend von Bedeutung, *welche* Entitäten navigiert werden sollen und *wie* diese Entitäten in Beziehung zueinander stehen, weniger wie diese konkret ausgeprägt sind. Konkrete Ausprägungen dieser Entitäten, beispielsweise in Form von Attributen, sind nur dann von Interesse, wenn sie für die Realisierung dedizierter Navigationsstrukturen benötigt werden.

6.4.1 Navigationsmodellierung

Um die Wiederverwendung von Navigationsstrukturen zu fördern und die Evolution besser zu unterstützen, wird auf das in Abschnitt 5.3.1.1 definierte Komponentenmodell zurückgegriffen. Die Kapselung der Navigationsstrukturen in konfigurierbare Komponenten hat den Vorteil, dass diese in unterschiedlichen Web-Anwendungen wieder verwendet werden können. Außerdem lassen sich so sukzessive neue Navigationsstrukturen in Form von fachlichen Komponenten in Form dedizierter Dienstelemente hinzufügen oder bestehende Komponenten weiterentwickeln (vergleiche Komponentenevolution).

Das Ziel der neuen Navigationsmodellierung ist es, die Ausschließlichkeit eines konzeptuellen Modells wie dies beispielsweise in den Objekt-orientierten Ansätzen (vgl. Abschnitt 3.2.3) üblich ist, zu Gunsten einer feineren Untergliederung zu erweitern. Um den Anforderungen an die Integration heterogener Informationsquellen, die Wiederverwendung von Navigationsstrukturen und der Berücksichtigung der Evolution des Informationsraums gerecht zu werden, bedarf es zusätzlicher Beschreibungen und Abstraktionen. Abbildung 6-17 stellt einen Überblick der dafür benötigten Phasen für eine unterstützende Methodik zur Navigationsmodellierung dar.

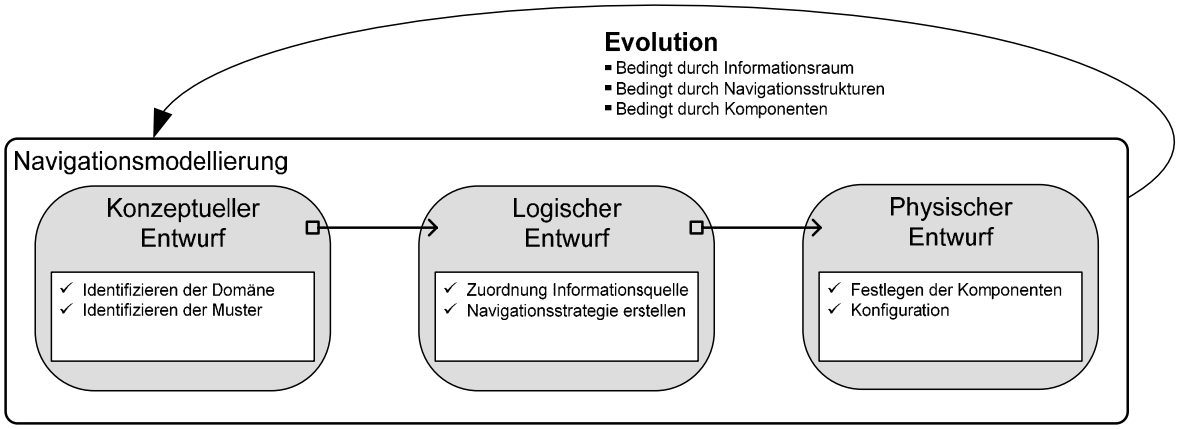


Abbildung 6-17: Die drei Phasen des Navigationsentwurfs.

Die dargestellte Methodik umfasst drei Schritte, auf die im Folgenden genauer eingegangen wird.

Konzeptueller Entwurf

Die Konzeptualisierungsphase legt zunächst die Anwendungsdomäne des Anwendungskontextes fest, die Gegenstand der Navigationsunterstützung werden soll. Dabei werden zwei Domänenformen unterschieden: Strukturierungsdomänen und Domänendienste. Zusätzlich werden zu unterstützende Navigationsstrategien festgelegt. Eine Navigationsstrategie ist eine Umsetzung eines allgemein beschriebenen Navigationsmusters. Beschreibungen für verschiedene Navigationsmuster wie »Index Navigation« oder »Hybrid Collection« finden sich im Hypermedia Design Pattern Katalog (Lowe, 1999).

Logischer Entwurf

Der logische Entwurf nimmt eine Zuordnung der Navigationsstrategie und einer möglichen Informationsquelle vor (*Bindung*). Wurde eine Strukturdomäne in der Konzeption identifiziert, so bilden beispielsweise die Kinder dieser Domäne eine Informationsquelle. Wurde ein Domänendienst identifiziert, kann eine Bindung beispielsweise durch Informationsraum-Dienstelemente erfolgen. Auf diese Weise lassen sich beliebige, im Informationsraum vorhandene Dienste kombinieren und deren Informationen den Benutzern passend zu ihren jeweiligen Anforderungen und Kontexten zur Verfügung stellen.

Physischer Entwurf

Der physische Entwurf umfasst die Zuordnung von Komponenten, die, basierend auf den Beziehungsgeflechten der Ontologie und der logischen Informationsraum-Bindung, Navigationsstrategien umsetzen und die darin enthaltenen Navigationsmuster realisieren. Bei Strukturierungsdomänen ist sowohl das Beziehungsgeflecht, als auch die Informationsquelle durch den Anwendungskontext implizit gegeben. Im Falle eines Domänendienstes, der prinzipiell beliebige Teile des Informationsraums (in Form von IR-Diensten) navigierbar machen kann, wird auf die semantischen Beschreibungen des Informationsraum-Entwurfs zurückgegriffen. Die physiche Umsetzung der Navigationsstrategien erfolgt mit Hilfe dedizierter fachlicher Komponenten des WSLS Komponentenmodells, den Navigations-Dienstelementen.

6.4.2 Navigationsunterstützung für Strukturierungsdomänen

Eine Strukturierungsdomäne übernimmt die logische Anordnung ihrer Kinder. Wie sie diese Aufgabe erfüllt wird durch entsprechende Dienstelemente festgelegt. So wurden im vorangegangenen Abschnitt Darstellungselemente beschrieben, die eine visuell geprägte Strukturierung gestatten. Im Folgenden wird aufgezeigt, wie Navigations-Dienstelemente zu einer Strukturierungsdomäne hinzugefügt werden, um eine strukturierte Navigation über deren Kinder zu ermöglichen.

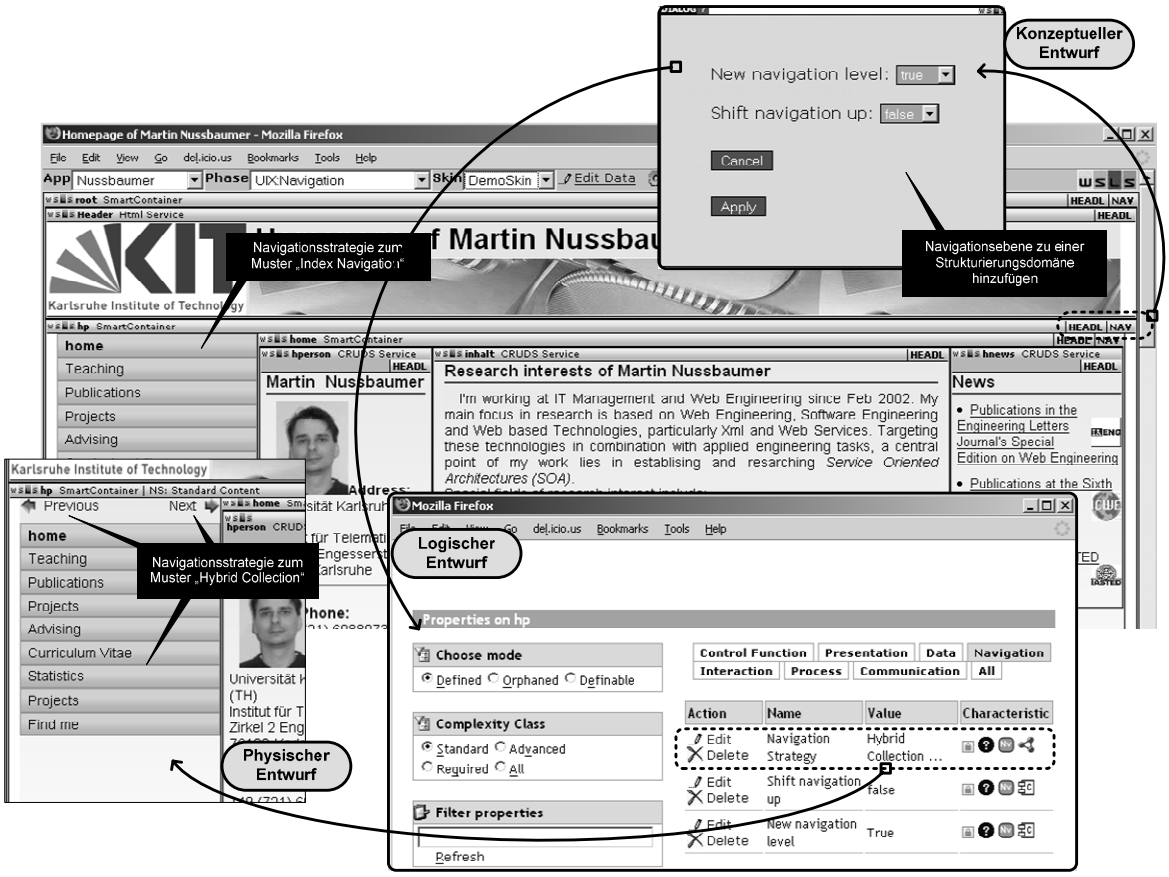


Abbildung 6-18: Zusammenspiel der unterschiedlichen Entwurfsphasen beim Navigationsentwurf auf einer Strukturierungsdomäne.

Abbildung 6-18 zeigt das Zusammenspiel der unterschiedlichen Entwurfsphasen auf, die zur Konfiguration von Navigationsstrategien einer Strukturierungsdomäne nötig sind. Die Anwendungssicht Navigation gestattet hierzu die nötige methodische Unterstützung, um die entsprechenden Phasen des Navigationsentwurfs durchzuführen. Im dargestellten Szenario wird zunächst eine Anwendungsdomäne ausgewählt und – basierend auf den Kindern dieser Domäne – eine Navigationsebene eingerichtet. Dies geschieht durch Hinzufügen eines Navigationsdienstelements, das über den generischen Andockpunkt des Anwendungsspezifischen Basisdienstes mit der Domäne verbunden wird (vgl. Abschnitt 5.4). Die Menge der im Kontext einer gewählten Strukturierungsdomäne anwendbaren Navigations-Dienstelemente wird – wie in Abschnitt 5.5.1 beschrieben – gebildet. Die logische Bindung einer solchen Strategie an die Domäne kann dann durch Auswahl aus der gebildeten Menge an passenden

Komponenten erfolgen. In der Abbildung ist die Anwendung der zwei Navigationsstrategien »Index Navigation[69]« und »Hybrid Collection[70]« dargestellt. Sie können über die Eigenschaften der Strukturierungsdomäne angepasst werden.

6.4.3 Navigationsunterstützung für Domänendiensten

Im Folgenden wird eine Navigationsunterstützung vorgestellt, die aufbauend auf den Phasen der vorgestellten Navigationsmodellierung beschreibt, wie Informationsraum-Dienste auf Basis ihrer semantischen Beschreibungen in der WSLS-Referenzarchitektur umgesetzt und zum Gegenstand einer Navigationsmodellierung gemacht werden können.

Dabei wird die semantische Konzeptualisierung der Informationsraum-Dienste zur Erstellung einer Ontologie wiederverwendet. Die in Abschnitt 4.3 eingeführten Szenarien »Erstellen neuer Informationsquellen«, »Integration inkompatibler Altsysteme durch IR-Dienste« und »Evolution existierender Informationsraum-Dienste« können aufgegriffen werden, um eine systematische Entwicklung und Evolution der Ontologie für den konzeptuellen Entwurf zu gewährleisten.

Abbildung 6-19 stellt den Zusammenhang zwischen den einzelnen Phasen der Navigationsmodellierung durchgeführt an einem Beispiel dar. Das fiktive Szenario beschreibt ein virtuelles Museum, das seinen Ausstellungskatalog, der Gemälde, Skulpturen usw. enthält, in Form einer Web-Anwendung für Besucher oder interessierte Personen anbieten möchte. Zusatzinformationen zu Künstlern oder Kunstgegenständen, die durch Informationsangebote Dritter bereitgestellt werden, sollen über IR-Dienste integrierbar sein mit dem Ziel einer einheitlichen Navigationssicht.

[69] Das Muster *Index Navigation* stellt einer Menge von Knoten einen Index von Verweisen zur Verfügung.

[70] Das Muster *Hybrid Collection* bildet ein Hybrid aus den zwei Mustern Index Navigation und Guided Tour. Letzteres stellt zusätzliche Intra-Navigationsunterstützungen mit vor- und rückwärts Verweisen zur Verfügung.

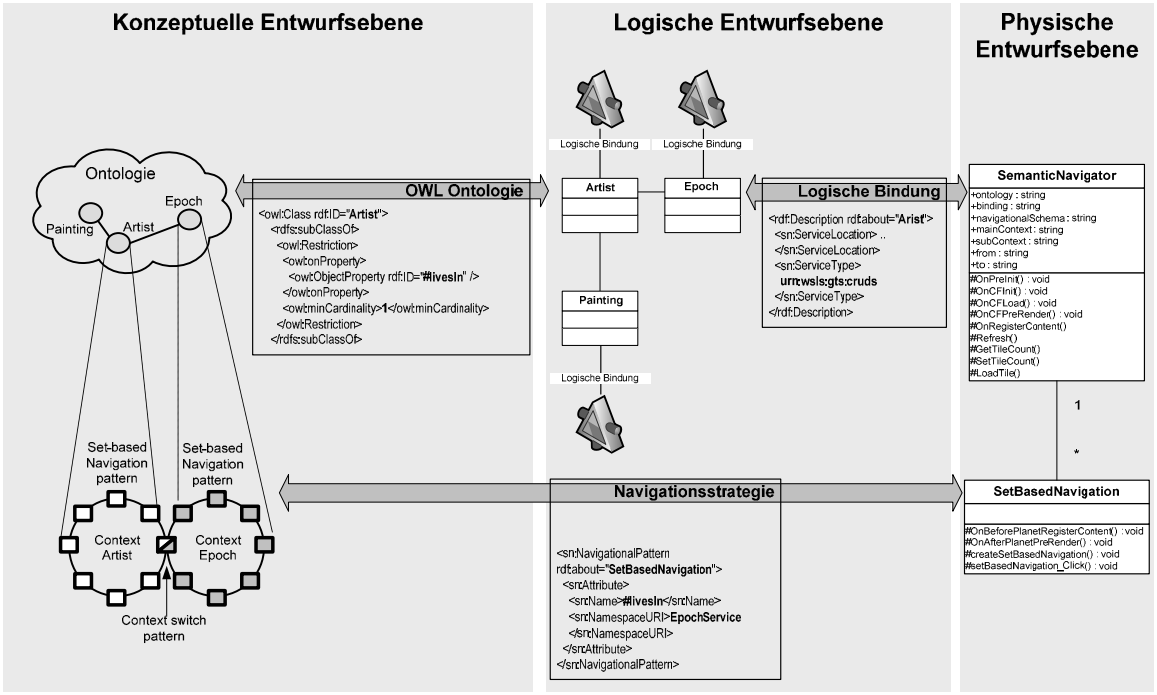


Abbildung 6-19: Navigationsentwurf für Domänendienste auf Basis von Ontologien.

Ontologien stellen in der konzeptuellen Entwurfsphase eine Beschreibung der abstrakten Navigation dar. Sie bieten also ideale Voraussetzungen, um die Evolution in Form von Erweiterungen des Navigationskonzeptes – beispielsweise durch Hinzunahme weiterer autonomer Dienste, die a priori nicht bekannt sind – durch Inferieren zusätzlichen Wissens zu ermöglichen (vgl. hierzu Abschnitt 3.1.5.1). Listing 6-3 zeigt einen Ausschnitt einer OWL Ontologie, die eine Klasse mit einer beschränkten Assoziation beschreibt.

```
01.  <owl:Class rdf:ID="Artist">
02.   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
03.   <rdfs:subClassOf>
04.    <owl:Restriction>
05.     <owl:maxCardinality df:datatype="http://www.w3.org/2001/XMLSchema#int">2
06.     </owl:maxCardinality>
07.     <owl:onProperty>
08.      <owl:ObjectProperty rdf:about="#livesIn" />
09.     </owl:onProperty>
10.    </owl:Restriction>
11.   </rdfs:subClassOf>
12.  </owl:Class>
13.  <owl:ObjectProperty rdf:about="#livesIn">
14.   <rdfs:domain rdf:resource="#Artist" />
15.   <rdfs:range rdf:resource="#Epoch" />
16.  </owl:ObjectProperty>
```

Listing 6-3: Ausschnitt einer Ontologie, die ein Konzept mit einer Restriktion definiert.

Die Klasse *Artist* (Zeile 1) besitzt eine Eigenschaft *livesIn* (Zeile 13), die ausdrückt, dass ein Artist in einer Epoche lebt. Durch eine zusätzliche semantische Konsistenzregel wird diese Eigenschaft mit einer maximalen Anzahl Epochen beschränkt (*maxCardinality*) (Zeile 5).

In der sich anschließenden logischen Entwurfsphase werden die Informationsraum-Dienste zu den in der Ontologie beschriebenen Klassen zugeordnet. Hierbei wird neben dem Endpunkt des Dienstes auch seine Kanonik spezifiziert. Die Modelltrennung ist besonders vorteilhaft, wenn IR-Dienste aus unterschiedlichen Informationsräumen eingebunden werden müssen. Diese *föderierten* Dienste verfügen über keinerlei gegenseitiges Wissen bezüglich des dem jeweils Anderen zugrunde gelegte Datenmodell. Um diese gewünschte lose Kopplung aufrecht zu erhalten ist auch eine nachträgliche Kopplung nicht wünschenswert. Eine semantische Annotation als eine quasi »lose Informationskopplung« bietet sich daher an, um diese Trennung logisch zu überbrücken.

Listing 6-4 stellt eine logische Bindung vor. Das Element *ServiceLocation* spezifiziert, unter welcher URL ein entsprechender Informationsraum-Dienst zu finden ist. Der *ServiceType* beschreibt die jeweilige Kanonik des IR-Dienstes. Die Kanonik begünstigt eine effiziente Integration des IR-Dienstes, indem bereits existierender Informationsraum-Dienst-Proxies wiederverwendet und entsprechend konfiguriert werden können.

```
01.  <rdf:Description rdf:about="Artist">
02.    <sn:ServiceLocation>
03.  http://services.tm.uni-karlsruhe.de/ArtistService/Service.asmx
04.    </sn:ServiceLocation>
05.    <sn:ServiceType>urn:wsls:gts:cruds</sn:ServiceType>
06.  </rdf:Description>
```

Listing 6-4: Logische Bindung von IR-Diensten mit unterschiedlichen Kanoniken.

Die zu entwickelnde Navigationsstrategie dient der Realisierung von Navigationsstrukturen basierend auf verschiedenen Navigationsmustern. In (Ofer und Kneisl, 2006) wird ein Musterkatalog mit Hilfe der eingeführten Navigationsmodellierung vorgestellt und aufgezeigt, wie diese Muster in die WSLS-Referenzarchitektur integriert werden. Dabei wurde die Navigationsmodellierung durch den Einsatz prominenter Vertreter des Hypermedia Design Pattern Katalogs anhand des Beispielszenario eines virtuellen Museums validiert.

Abbildung 6-20 zeigt die Anwendung der vorgestellten Navigationsmodellierung im Einsatz mit verschiedenen Navigationsmustern. Die dafür erzeugte Ontologie findet sich im Anhang. Die einmal entwickelten Navigations-Dienstelemente (schematische Darstellung in der Abbildung linker Hand) können in beliebigen anderen Szenarien wiederverwendet werden. Dazu müssen lediglich die entsprechenden Modelle und Dienstbindungen angepasst werden.

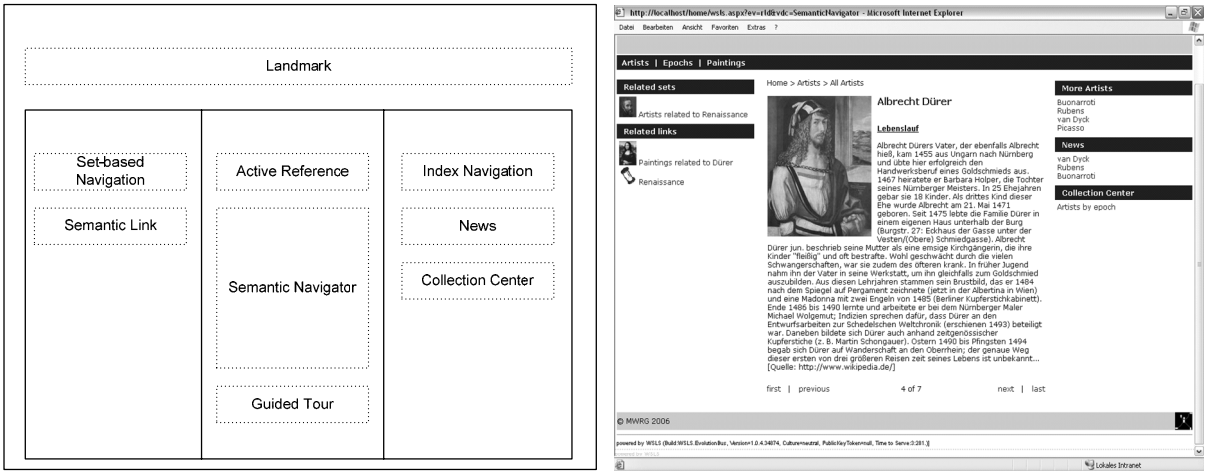


Abbildung 6-20: Beispiel der Navigationsunterstützung anhand eines virtuellen Museums basierend auf IR-Diensten.

6.5 Anwendungssicht Dialog

»Interaction is not animation.

It's not audio.

It's not video.

It's user control and dynamic experience.«

(Terry R. Schuster)

Die zunehmende Fokussierung auf den Anwendungscharakter kombiniert mit dem Wunsch nach *dynamischen* Benutzungsschnittstellen hat in jüngster Zeit zu einer neuen Generation von *dienstorientierten* Anwendungen geführt, die sich verstärkt aktueller Technologien wie Web Services bedienen. Im Umfeld dieser rein technologischen Neuerung entwickelt sich eine völlig neue Form von Web-Anwendungen, die stärker auf direkter Interaktion und Kommunikation zwischen den Anwendern beruht. Dabei kommt ihren Nutzern nicht mehr länger eine nur passive Betrachterrolle zu, vielmehr werden sie zu Akteuren, die als Lieferanten von Inhalten oder in der Bewertung von bestehenden Inhalten fungieren. Das führt zu Anforderungen an die Einflussgröße Dialog einer Benutzungsschnittstelle hinsichtlich einer besseren Bedienbarkeit durch weniger erfahrene Benutzer.

6.5.1 Die MVC²-Architektur

Eine dedizierte Verlagerung von Teilen der Geschäftslogik in den Browser verspricht aufgrund der durch den Anwendungskontext gegebenen losen Kopplung in der Benutzungsschnittstelle eine sinnvolle Erweiterung und Verteilung der Komplexität. Das Ausmaß einer solchen Verlagerung kann dabei, beginnend mit dem einfachen Überprüfen der Benutzereingaben auf ihre Validität in einem speziellen Kontext, bis hin zu einem breiten Spektrum vollständiger für sich abgeschlossener Anwendungsteile reichen. So können beispielsweise

Informationsraum-Dienste zur Berechnung oder Vervollständigung von Informationen aufgerufen oder fortschrittliche Hilfestellungen durch den Einsatz von Avataren[71] bereitgestellt werden. Um diese Ziele zu erreichen, muss die Client-Seite teilweise autonome Entscheidungen darüber treffen können, beispielsweise wann und ob IR-Dienste aufgerufen werden. Dazu wird ein neues Modell eingeführt, welches das bekannte MVC-Muster (Krasner und Pope, 1988) derart erweitert, dass einem Client eine solche Autonomie ermöglicht wird:

- Die Clientseite kann nach dem MVC-Modell selbstständig agieren, indem Daten entsprechend gewisser Vorgaben visualisiert *und* auf Benutzereingaben durch eine rückkoppelnde Kommunikation mit dem Informationsraum reagiert wird.
- Gleichzeitig findet auch eine Kommunikation mit der serverseitigen Anwendungslogik statt.

In Anlehnung an diese Doppelung des MVC-Modell durch Verlagerung auf die Clientseite wird dieses Architekturmuster als *(MVC)²-Modell*[72] bezeichnet. Abbildung 6-21 stellt den Wirkungsbereich auf die Benutzungsschnittstelle in einer (MVC)²-Web-Anwendung einer Web-Anwendung in heutzutage üblicherweise eingesetzten dreischichtigen Architektur (*engl*. 3-tier architecture) gegenüber.

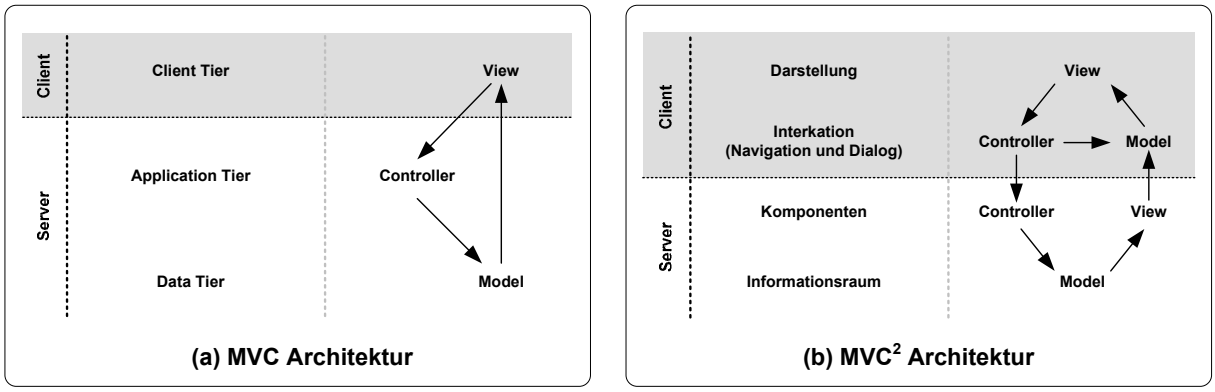


Abbildung 6-21: Anwendung des MVC Musters auf eine Web-Anwendung. Links dargestellt ist die klassische Variante (a), rechts die auf dem neuen (MVC)²-Modell (b).

Neu ist die Schichtung auf Clientseite, die neben der Präsentationsschicht auch die interaktiven Aspekte *Navigation* und *Dialog* berücksichtigt. Der Bereich Dialog umfasst dabei jede Interaktion eines Benutzers mit dem Modell, also den dargestellten Informationen an sich, sowie insbesondere die Möglichkeiten ihrer Manipulation. Für den Benutzer ist es dabei weitgehend transparent, ob er mit dem clientseitigen oder direkt mit dem serverseitigen Modell interagiert. Aus Benutzersicht stellt sich die Web-Anwendung als eine atomare Einheit dar, während Änderungen im Browser zu einem vorgegebenen, späteren Zeitpunkt an den Server übertragen werden können.

---

[71] Avatare bezeichnen laut Duden Indische Gottheiten in menschlicher Gestalt.

[72] Im Zusammenhang mit MVC wird gelegentlich auch vom MVC-2 Muster gesprochen, was im Gegensatz zum vorgestellten MVC² lediglich eine Trennung eines *Controls*, das HTTP-Requests verarbeitet, von einem View, das eine HTML-Response erzeugt.

Diese Entkopplung der Kommunikation findet also im Gegensatz zum heute üblichen klassischen Kommunikationsmuster der Client-/Server-Anwendungen *asynchron* statt. Die Asynchronität bezeichnet die Unabhängigkeit von Ereignissen, die direkt durch den Benutzer ausgelöst werden. Sie fördert darüber hinaus eine Verbesserung der Antwort- und Reaktionszeit einer Web-Anwendung, da eine nahtlose Interaktion mit dem bereits geladenen Teil der Web-Anwendung möglich ist. Hier wirkt sich die in Kapitel 4 eingeführte Dienstorientierung des Informationsraums besonders günstig aus, da einzelne IR-Dienste direkt für eine asynchrones Kommunikationsverhalten in die Anwendung einbezogen und wiederverwendet werden können (im Bild als Controller auf der Client Seite).

Effizienz durch dedizierte Informationsversorgung

Die effiziente Auslieferung von relevanten Informationen ohne ein erneutes Verschicken der kompletten Seiteninhalte über HTTP wird ermöglicht. Abbildung 6-22 stellt das Kommunikationsverhalten von Web-Anwendungen basierend auf einer MVC^2-Architektur denen basierend auf einer MVC-Architektur gegenüber.

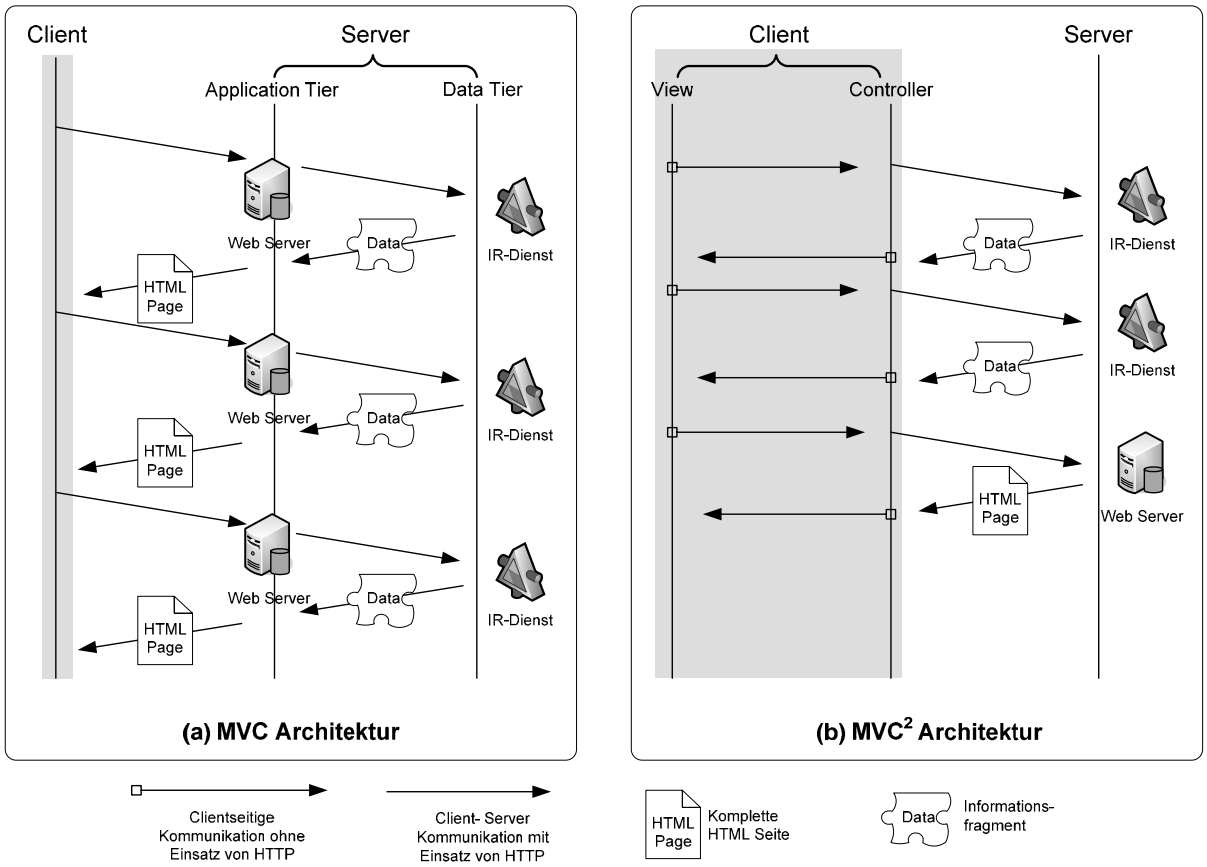


Abbildung 6-22: Vergleich der Informationsmenge und des damit verbundenen Kommunikationsaufwands einer Web-Anwendung nach MVC^2 (a) und einer herkömmlichen MVC Architektur (b).

Die Vorteile, die sich daraus ergeben sind zweifältig. Aus Sicht eines Betreibers wird die Serverlast minimiert, da nur noch Teile und nicht die ganze Web-Anwendung aktualisiert und ausgeliefert werden muss. Aus Sicht des Endnutzers werden Ladezeiten drastisch verkürzt,

wodurch das Arbeiten mit einer solchen Web-Anwendung zu einem positiven Erlebnis wird. Zudem begünstigt die MVC^2 Architektur die Skalierbarkeit[73] durch die Verteilbarkeit der Informationsraum-Dienste.

6.5.2 Die UIRML-DSL

Um Entwickler von Dialogen basierend auf der eingeführten MVC^2-Architektur zu unterstützen, wird ein methodisches Vorgehen entwickelt. Dazu wird zunächst eine Domänenspezifische Sprache UIRML (*User Interface Relationship Modeling Language)* vorgestellt, die auf dem in Abschnitt 0 vorgestellten DSL-Rahmenwerk basiert.

Die Sprachunterstützung basiert auf der Idee Abhängigkeiten zwischen Elementen innerhalb einer interaktiven Umgebung zu definieren. Ein weit verbreiteter Ansatz hierzu sind die so genannten ECA (**E**vent, **C**ondition, **A**ction) Regeln. Sie haben ihren Ursprung im Bereich der aktiven Datenbanken (Dayal, 1988) finden sich aber inzwischen in vielen unterschiedlichen Bereichen wieder (Bailey, Poulovassilis und Wood, 2002) (Bonifati, Ceri und Paraboschi, 2001). Eine ECA Regel besteht aus einem auslösendes Event (E), einer Bedingung (C), die erfüllt sein muss und einer durchzuführenden Aktion (A), falls die Bedingung erfüllt ist.

Um eine Sprachunterstützung auf Basis von ECA Regeln zu entwickeln, muss zunächst geklärt werden, welches die Trägerelemente für einen interaktiven Vorgang sind, die Gegenstand anwendbarer ECA Regeln werden sollen.

> *Definition 6.2* - **Interaktionsträger**: Als Interaktionsträger werden die Repräsentanten eines interaktiven Vorgangs bezeichnet. Sie sind beliebige, referenzierbare Objekte, die asynchron Ereignisse auslösen können und Methoden und Eigenschaften anbieten.

Die Definition umfasst sowohl visuelle wie nicht-visuelle Elemente, was konkret die Einbeziehung von Informationsraum-Diensten als Interaktionsträger ermöglicht. Ein Anwendungsfall, wie beispielsweise das dynamische Füllen einer Liste in einem Formular mittels eines Informationsraum-Diensts, wird ermöglicht, indem eine Abhängigkeit zwischen einem visuellen Listenelement und dem nicht-visuellen Informationsraum-Dienstelement hergestellt wird. Auf diese Weise kann beispielsweise ein Informationsraum-Dienst, der für die Bereitstellung von Bildern konzipiert wurde, eine Auswahlliste passend befüllen. Um allerdings ein Höchstmaß an Flexibilität bei interaktiven Vorgängen zu erhalten, bedarf es zusätzlich der Einführung eines Kontextes, in dem sich der korrespondierende Interaktionsträger befindet. Dadurch lässt sich im eben vorgestellten Beispiel eine bereits getätigte Eingabe als *Kontext* zur Durchführung einer Suche verwenden.

[73] Die Restriktionen mancher Browser nur Verbindungen zu URIs aus der gleichen Domäne zuzulassen können durch Proxy Mechanismen gelöst werden.

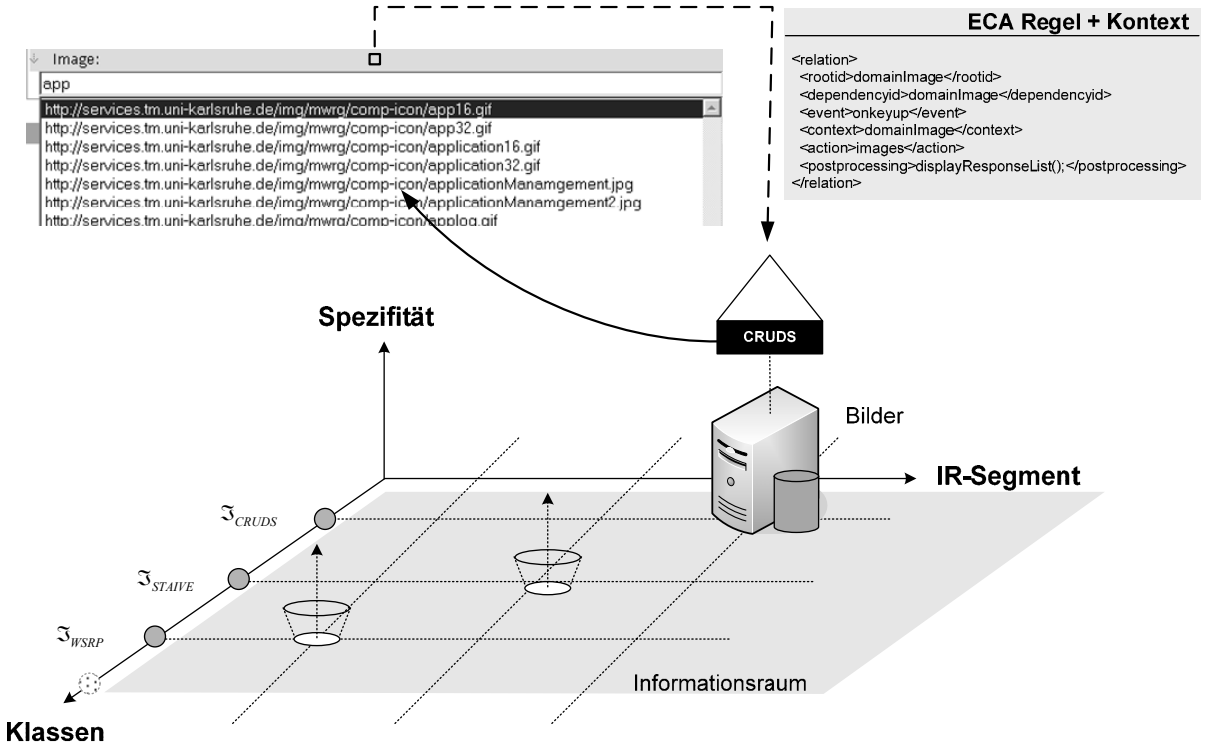


Abbildung 6-23: ECA Regel mit übergebenem Kontext zur Integration von Informationsraum-Diensten in den Dialogentwurf.

Abbildung 6-23 stellt das Zusammenwirken einer ECA Regel mit dem Kontext des Interaktionsträgers (Eingabefeld) dar. Durch ein Ereignis (*onkeyup*) auf dem Wurzelelement (*domainImage*) wird eine Aktion (*images*) durchgeführt. Der Identifikator »images« wird durch ein spezielles Diensteprofil (*service profile set*) aufgelöst. Die Notwendigkeit für ein Diensteprofil hat zwei Gründe. Einerseits können in einer MVC^2-Web-Anwendung nur Dienste aus der gleichen Netzwerk-Domäne aufgerufen werden. Um auch Informationsraum-Dienste von Drittanbietern verwenden zu können, bedarf es dann eines Proxy Mechanismus, der einen Tunnel über einen lokalen Informationsraum-Dienst zur Verfügung stellt. Über diesen Tunnel lassen sich dann beliebige IR-Dienste einbinden.

Der andere Grund liegt in einer einfacheren Handhabung durch ein unterstützendes Interaktionswerkzeug (DIM Editor). Anstatt mit einem URL (Uniform Resource Locator) oder Informationsraum-Identifikator (vgl. Abschnitt 4.2) umzugehen, wird auf einen durch Menschen besser erfassbaren Begriff zurückgegriffen.

Die formale Beschreibung des spezifischen Domänenmodells (DSM), welches die vorgestellten Relationskonzepte umsetzt findet sich im Anhang. Hierbei wird besonderer Wert auf eine querschnittliche Realisierung des Dialogaspekts gelegt. Das Interceptor-Entwurfsmuster (Schmidt, Stal und Rohnert, 2000) erlaubt eine entkoppelte Betrachtung zwischen Elementen und zusätzlich zu definierenden Aktionen. Dadurch können existierende Formulare mit Hilfe der UIRML erweitert werden ohne eine Änderung existierender Formulare vorzunehmen. In (Rudin, 2006) wird auf dem Prinzip von Beziehungen zwischen Elementen einer Benutzeroberfläche ein Ansatz vorgestellt, der auf der Grundlage dieser entkoppelnden Muster verschiedene Anwendungsszenarien mit der UIRML validiert.

Das Interaktionswerkzeug zur Bearbeitung eines Dialogs und Spezifizierung der DSL ist in Abbildung 6-24 dargestellt. Der Lösungsbaustein zur Ausführung der DSL wird durch ein Dia-

log-Dienstelement erbracht, das aus der Datenbeschreibung eines ContentObjects automatisch Formulare erzeugen kann (Allerding, 2007). Diese Formulare werden dann mit Hilfe des DIM Editors bearbeitet, um neue Relationen hinzuzufügen, bestehende zu ändern oder zu löschen.

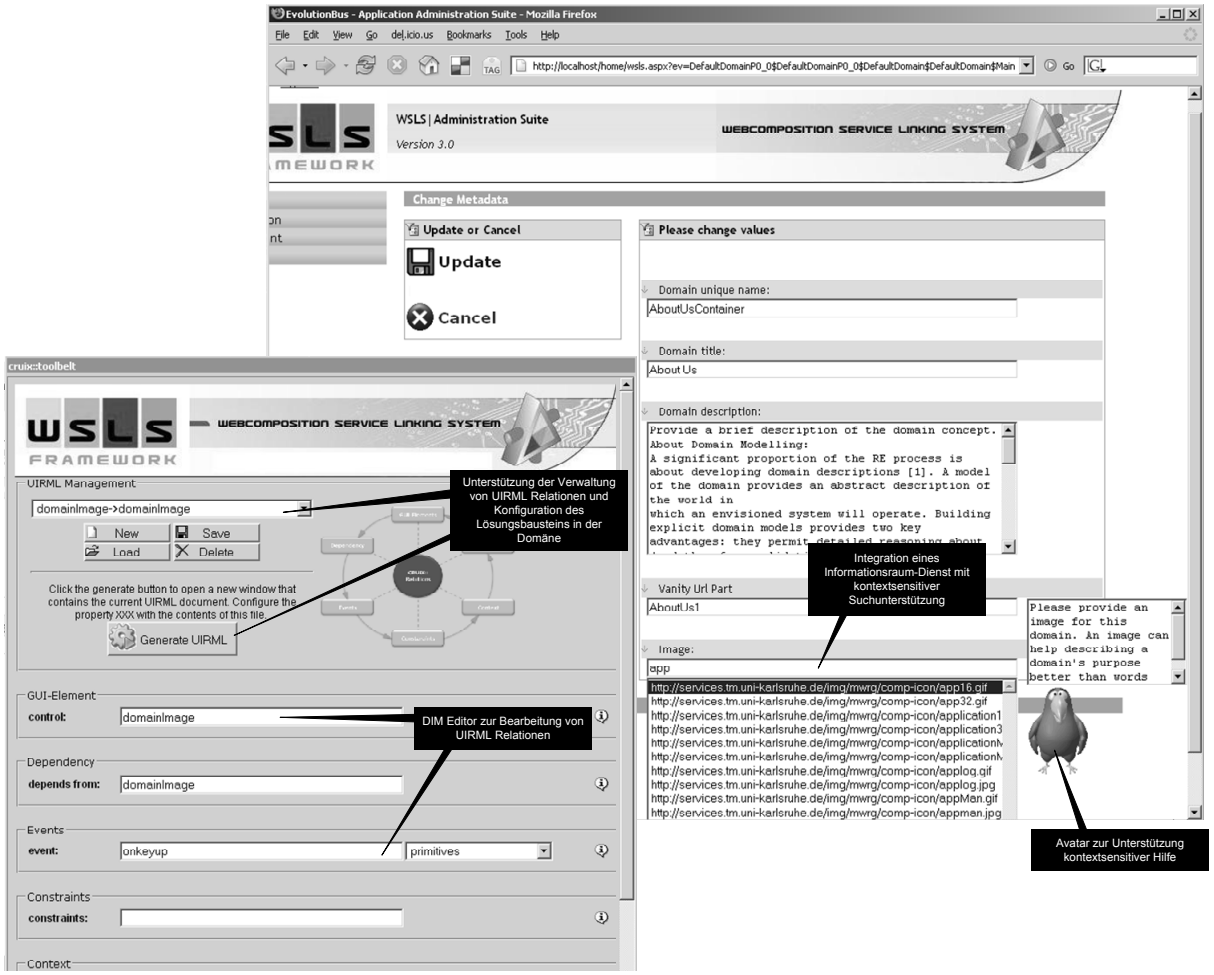


Abbildung 6-24: DIM Editor für die Bearbeitung der UIRML DSL in der Anwendungssicht Dialog.

6.6 Zusammenfassung

In diesem Kapitel wurde der Konstruktionsrahmen für die Modellierung von dienstorientierten Web-Anwendungen gelegt. Dies geschah zunächst durch die Formalisierung einer Anwendung in Form eines Anwendungskontextes, der einen Graph aus konzeptuellen Komponenten (Domänen) repräsentiert. Dadurch wird die Architektur in den Vordergrund gestellt, indem sie eine Anwendung als Instanz dieser Architektur betrachtet. Diese innovative Sichtweise offeriert dabei Vorteile bezüglich der Anwendbarkeit formaler Methoden. So konnte durch die Abstraktion einer Hyperanwendung die systematische Unterstützung zur Bestimmung maturierter Verbünde von Domänen erreicht werden. Der Reifegrad einer Teilmenge von Domänen stellt sich als eine mehrfach auftretende Wiederverwendung bestimmter, zusammenhängender Bereiche des Domänenraums dar. Diese Verbände indizieren damit

Kandidaten für einen eigenständig wiederverwendbaren Anwendungsbaustein zu sein und stellen somit einen Evolutionsschritt der beteiligten Domänen dar.

Die Übertragung der Dienstorientierung auf die Benutzungsschnittstelle wird durch das Konzept der Anwendungssicht erreicht. Mit Hilfe von »Dekorateuren« werden bestimmte – dieser Sicht zugeordnete – Funktionen unter einer bestimmten Intention zusammenfasst. Je nach Bedarf lassen sich dann zusätzliche Anwendungssichten definieren, die beispielsweise auf einen dediziert zu behandelnden Aspekt oder eine favorisierte Methodik zugeschnitten werden können. Das vorgestellte Anwendungsmodell lässt sich damit bezüglich der zu unterstützenden Phasen oder Aktivitäten prozessorientiert skalieren. Die vielseitige Einsetzbarkeit einer Anwendungssicht wurde an den Aspekten Präsentation, Navigation und Dialog aufgezeigt, die den orthogonalen Entwurf von Web-Anwendungen gezielt ermöglicht.

Die Darstellung von Informationen für den Benutzer wurde erstmalig aus zwei Blickwinkeln betrachtet. Zum einen als die logisch strukturierte Darstellung von Informationen mit Hilfe existierender Web-Standards (vgl. Abschnitt 6.3). Zum anderen wurde eine Methodik entwickelt, die eine automatisierte Überprüfung und Durchsetzung von Regeln zur Gewährleistung des barrierefreien Zugriffs ermöglicht und mittels standardisierter Web-Technologien umsetzt (vgl. Abschnitt 6.2.3.1). Das betrachtete Regelwerk ermöglicht die systematische Berücksichtigung der Web Content Accessibility Guidelines Empfehlung (WCAG) des World Wide Web Consortium (W3C), was die Grundlage bildet für die »Verordnung zur Schaffung barrierefreier Informationstechnik« (BITV) nach dem Behindertengleichstellungsgesetz, das Ende 2005 in Deutschland in Kraft trat. Die Integration dieser Methodik in den Entwurfsprozess konnte durch die eingangs definierte Anwendungssicht erfolgreich demonstriertert werden. Auf diese Weise können neben einer Qualitätssteigerung durch eine verbesserte Zugänglichkeit der Benutzungsschnittstelle erstmalig auch inhaltsbezogene Empfehlungen zum Abbau von Barrieren formuliert werden (proaktive Zugänglichkeit).

Durch die Einführung einer logischen Strukturschnittstelle konnte eine weitestgehende Trennung von Inhalt und Darstellung erreicht werden. Die formale Repräsentation des Anwendungsgraphen erlaubt darüber hinaus die gezielte Einbindung eines Unterstützungswerkzeuges für einzelne Knoten des Graphen. Dadurch wird die Wiederverwendung von bereits existierendenden ästhetischen Darstellungsentwürfen erheblich gesteigert. In Hinsicht auf föderative Nutzungsszenarien offeriert dieser Ansatz enormes Potenzial, da fremde Entwürfe äußerst effizient in ein vorherrschendes Corporate Design überführt werden können.

Das vorgestellte Navigationsmodellierungskonzept berücksichtigt als Einziges die Dienstorientierung der Benutzungsschnittstelle. Durch die Kapselung von Navigationsmustern in konfigurierbare Komponenten wurde deren Wiederverwendbarkeit in unterschiedlichen Web-Anwendungen erreicht. Dadurch wird außerdem die Unterstützung der Evolution gefördert, da sich so sukzessive neue Navigationsstrukturen als fachliche Komponenten in Form dedizierter Dienstelemente hinzufügen und entfernen (Selektion) lassen oder bestehende Navigationsstrukturen gezielt weiterentwickelt (Variation) werden können.

Auf Grundlage des DSL-Rahmenwerks wurde die UIRML (*UI Relationship Markup Language*)-DSL vorgestellt. Sie beruht auf dem Prinzip von Beziehungen zwischen Elementen der Benutzeroberfläche, ausgehend von verschiedenen Entwurfsmustern und Beobachtungen über implizit existierende Abhängigkeiten zwischen diesen Elementen. Sie dient der formalen Beschreibung von Abhängigkeiten zwischen *Interaktionsträgern* sowie deren asynchronen Verknüpfungen mit den Informationsraum-Diensten. Dadurch können Teile eines Formulars mit

den Inhalten dieser Dienste effizient verknüpft werden, wodurch die Basis für neuartige Dialogstrukturen geschaffen wurde.

7 Zusammenfassung und Ausblick

Die vorliegende Arbeit untersucht eine methodische und systematische Herangehensweise für die Entwicklung und Evolution von dienstorientierten Anwendungen im Web Engineering. Die eingangs vorgestellten Herausforderungen sind mit existierenden Ansätzen kaum zu überwinden. Die stark steigende Verbreitung webbasierter Anwendungsportale sowie die rasante Verbreitung dienstorientierter Konzepte stellen neuartige Anforderungen an die Entwicklung und insbesondere an die Evolution von dienstorientierten Web-Anwendungen. Der im Rahmen dieser Arbeit entwickelte WSLS Ansatz stellt eine Lösung bereit, welche die aufgezeigten Problemstellungen und Herausforderungen wirksam erfüllt.

7.1 Ergebnisse der Arbeit

In der vorliegenden Arbeit wurden mehrere Methoden des Web Engineering auf ihre Eignung für die Entwicklung und insbesondere Evolution von dienstorientierten Web-Anwendungen untersucht. Dabei wurde festgestellt, dass in den existierenden Ansätzen die Dienstorientierung bei der Entwicklung von Web-Anwendungen bisher kaum Berücksichtigung findet. Existierende Entwicklungsmethodiken beschäftigen sich insbesondere mit den Aspekten Daten und Hypertext. Die mangelnde Berücksichtigung der Dienstorientierung liegt im Aufbau und ideologischen Schwerpunkt dieser Methoden begründet, die deren interdisziplinäre Ausrichtung und Weiterentwicklung erschwert. Dem gegenüber stehen Prognosen, dass bis 2009 über 80% der neuentwickelten Anwendungen auf Diensten basieren werden (Hayward, 2005). Die diskutierten Ansätze des Web Engineering legen den Fokus vornehmlich auf die Neuentwicklung einer Web-Anwendung und vernachlässigen dabei Wartung, Weiterentwicklung und Anpassbarkeit. Die Anwendungslogik – im Sinne einer dienstorientierten Web-Anwendung also die Verknüpfung mit Diensten – muss dann im Nachhinein innerhalb eventuell automatisiert erzeugter Fragmente hinzuprogrammiert werden. Änderungen an den Modellen ziehen zwangsläufig Änderungen an diesen automatisiert erzeugten Teilen mit sich und erschweren ein nahtloses Zusammenfügen mit bereits modifizierten Teilen. Gerade hinsichtlich der Realisierung dienstorientierter Web-Anwendungen stellt der Wunsch nach lose gekoppelten, wieder verwendbaren Diensten erhebliche Anforderungen

an eben diese Phasen. So spielt gerade die *Evolution* von Diensten und Anwendungen eine wichtige Rolle, da in der Regel eine komplette Neuentwicklung nicht wünschenswert ist.

Besonders nachteilig wirkt sich die Vernachlässigung der Evolution auch bei Web-Anwendungs-relevanten Aspekten der Benutzungsschnittstelle wie der Darstellung aus. Herkömmliche Methoden verlagern präsentationsrelevante Aufgaben wie Ästhetik und Zugänglichkeit in die Implementierungsphase. Dadurch kommt es zu einer grobgranularen Abbildung der vormals oft sehr feingranular betrachteten Aspekte Content und Navigation. Die fehlende Reversibilität der vorgestellten Methoden verhindert eine durchgängig durchführbare Änderung an den Entwurfsartefakten. Dadurch kann die Darstellung nicht *dediziert angepasst*, *verändert* oder *stabilisiert* werden. Die durch diese Herangehensweise entstehende Problematik hinsichtlich Gewährleistung und Durchsetzung von Regeln zur Barrierefreiheit wird dadurch noch weiter verschärft. So fehlen konsequenterweise Ansätze und zumeist sogar Ansatzpunkte zu einer Unterstützung, obwohl Forderungen nach einer methodischen Unterstützung zur Qualitätssteigerung der Nutzerschnittstelle existieren (G. Costagliola et al., 2004).

Die schwergewichtigen Ansätze und Modelle des Web Engineering eignen sich nur bedingt für eine gemeinsame Anwendung mit verschiedenen Kunden, Endnutzern oder Projektbeteiligten. In den USA beispielsweise wird geschätzt, dass den rund 2,75 Millionen professionellen Entwicklern (*Software professionals*) ca. 53 Millionen Endnutzer-Progammierer (*Enduser Developers*) gegenüberstehen (W.Boehm, Horowitz, Madachy, Reifer et al., 2000). Nimmt man eine Gleichverteilung auf Projektteams an, was im Falle von Integrationsprojekten sinnvoll ist, heißt das im Umkehrschluss, dass lediglich 5% eines Teams eine ausreichend technische Ausbildung hat. Unter diesem Hintergrund kommt dem Aspekt der Kommunikation eine immense Bedeutung zu, da die Lösung technischer Probleme in erster Linie eine klar definierte, einheitliche Kommunikationsbasis, also eine gemeinsame Sprache, voraussetzt.

Den Stand der Technik zusammenfassend, wurde festgestellt, dass Entwicklung und Evolution von dienstorientierten Web-Anwendungen durch folgende Kernprobleme geprägt werden, mit deren Lösung sich die vorliegende Arbeit beschäftigte:

- Fehlende Berücksichtigung der Dienstorientierung
- Fehlende Berücksichtigung der Evolution durch Forward Engineering
- Fehlende Unterstützung für das Präsentationsdesign
- Fehlende Gewährleistung einer barrierefreien Benutzungsschnittstelle
- Fehlende Flexibilisierung schwergewichtiger Entwurfsmethoden

Ziel dieser Arbeit war es daher, eine methodische und systematische Herangehensweise an die Entwicklung von dienstorientierten Anwendungen im Web Engineering durch einen Architekturzentrierten Ansatz zu ermöglichen, der eine Web-Anwendung als Architekturinstanz in den Vordergrund rückt, deren Grundlage *autonome*, *wiederverwendbare* Komponenten in Form von Diensten darstellen. Dazu wurden Modelle, Methoden und Werkzeuge der auf Web-Technologien basierenden Referenzarchitektur entwickelt, die zwei Ziele stringent verfolgt:

- Förderung der Wiederverwendung von Komponenten und Diensten, sowie
- Unterstützung der Anwendungsevolution als der Evolution dieser Komponenten und Dienste.

Das Modell des Globalen Typ Systems (GTS) bildet den Kern der vorgestellten Informationsraum-Architektur, die auf der Axiomatik des Informationsraums des World Wide Web basiert. Um die Wiederverwendbarkeit von Dienstschnittstellen zu erhöhen, wird das Konzept der kanonischen Schnittstellen eingeführt, die den Zugriff auf den zugrunde gelegten heterogenen Informationsraum homogenisieren. Dabei werden speziell Dienstspezifikationen untersucht, die geeignet sind, eine dynamische Interaktion mit Web-Anwendungen effizient zu unterstützen. Neben der Lösung technischer Aspekte wie Sicherheit und Interoperabilität der umsetzenden Informationsraum-Dienste spielten hierbei auch nicht-funktionale Anforderungen wie beispielsweise die der Dienstzuschnitt eine wichtige Rolle. Eine grundlegende, in dieser Arbeit vorgestellte Dienstspezifikation orientiert sich an Dienstprimitiven zur Verwaltung und Manipulation des Informationsraums einer Web-Anwendung, die mit Hilfe von Web Services umgesetzt werden. Diese damit verbundene Schnittstellenkanonik wird exemplarisch am Beispiel der CRUDS-Schnittstelle aufgezeigt. Die systematische Unterstützung der Evolution wird durch ein Vorgehensmodell beschrieben, das unterschiedliche Ausprägungen der Evolution berücksichtigt. Diese vollzieht sich in zwei klar definierten Richtungen: horizontal und vertikal. Während die horizontale Evolution eine Erweiterung durch Hinzufügen weiterer Dienstspezifikationen darstellt, wird die vertikale Evolution durch eine fortwährende Weiterentwicklung und Anpassung einer Dienstspezifikation an ihre sich wandelnde Umgebung ermöglicht. Dies wird durch dedizierte Übergabekontexte (parametrisierte Polymorphie) erreicht, die entsprechend vorherrschender Anforderungen weiterentwickelt werden können.

Eine Besonderheit bildet das Konzept des Informationsraum-Identifikators, der eine Zuordnung zwischen einer Informationseinheit und einem Diensterbringer gestattet. Durch diese Entkopplung lassen sich auch fortgeschrittene Beschreibungsverfahren zur Dokumentation und dem Management ganzer Landschaften dienstorientierter Web-Anwendungen realisieren (Meinecke, Gaedke und Nussbaumer, 2005). So kann beispielsweise der dynamische Austausch von Diensten durch die zusätzliche Indirektion der Dienstidentifikation erleichtert werden, da nur die zentralen Teile einer Registrierungsdatenbank aktualisiert werden müssen. Fortgeschrittene Konzepte wie die *Selbstheilung von Diensten* können auf Basis einer geeigneten Zuordnung von Dienstidentifikatoren erreicht werden. Auch die systematische Zusammenführung verschiedener Informationsräume hin zur Ausbildung von *Informationsraum-Föderationen* lässt sich durch die Entkopplung einfacher arrangieren, da a priori von verteilten Adressräumen ausgegangen wird.

Das vorgestellte Rahmenwerk für Domänen-spezfische Sprachen (DSL) komplettiert die technisch-methodische Unterstützung der WSLS-Referenzarchitektur durch einen kommunikationsfördernden Aspekt. Unterstützungswerkzeuge lassen sich auf eine klar fokussierte Problemdomäne abbilden, die durch konzeptionelle Modelle (Domain Specific Model) und Notationen (Domain Interaction Model) beschrieben werden. In einer Umfrage bezogen auf die Web-Anwendungsentwicklung stellte sich heraus, dass nach wie vor überwiegend WYSIWYG (*what you see is what you get*)-Editoren eingesetzt werden, da sie einfach und intuitiv zu bedienen sind (Rosson, Ballin, Rode und Toward, 2005). Das DSL-Rahmenwerk öffnet die WSLS-Referenzarchitektur für die Anbindung unterschiedlicher, auf bestimmte Personengruppen zugeschnittener visueller Editoren, die auf dem selben Konzept basieren. WSLS dient hierbei als technische Integrationsplattform, um Anwendungsbausteine (Solution Building Blocks) zu Web-Anwendungen zusammenzufügen, die dann wiederum den Erfordernissen entsprechend durch DSL-Programme konfiguriert werden können. Dadurch lassen sich

diese Anwendungsbausteine beiliebig an Applikationsänderungen anpassen und erfüllen so die Anforderung der *konfigurierbaren Konvertibilität*.

Die Einführung fachlicher und konzeptueller Komponenten fördert die Wiederverwendung auf unterschiedlichen Entwurfsebenen und gestattet den Übergang des *kommunikationsorientierten* DSL-Vorgehens hin zu einer *funktionalen Prägung* der Problemdomänen durch Komponenten. Anwendungsspezifische Basisdienste unterstützen das Prinzip des »Separation-of-Concerns« durch die klare Auftrennung in unterschiedliche Aspekte einer Web-Anwendung durch fachliche Komponenten wie *Kontrollfunktionen* und dedizierte *Dienstelemente*, wodurch die Wiederverwendbarkeit der einzelnen Bestandteile erheblich gesteigert werden kann. Außerdem ermöglicht die Spezialisierung auf unterschiedliche Dienstelemente eine feingranulare Modellierung und Anwendung dedizierter, auf einen speziellen Aspekt fokussierter, Entwicklungsmodelle. Durch die Einführung eines Lebenszyklus werden Kontrollfunktionen befähigt, Dienstelemente miteinander zu verweben und fortgeschrittene Verwebestrategien von Aspekten zu unterstützen. Die funktionale Anpassung an ein sich änderndes Systemumfeld wird durch eine zwei-dimensionale Betrachtung der Komponentenevolution erreicht. Einerseits wird die etablierte Anwendung des Objekt-orientierten Paradigmas durch Generalisierung vorgenommen, um weitestgehend die Wiederverwendung existierenden Programmcodes und vorhandener Bibliotheken zu gewährleisten. Das dabei zwangsläufig entstehende Phänomen der dominanten Dekomposition wird durch die Anwendung Aspekt-orientierter Modelleigenschaften abgemildert. Die dafür entwickelten *wohldefinierten Verknüpfungspunkte* stellen den dafür notwendig Schnittpunkt zwischen dem Lebenszyklus des WSLS-Komponentenmodells einerseits und der Aspekt-orientierten Modellierung andererseits her. Dadurch lassen sich auch nicht-funktionale Eigenschaften, wie beispielsweise Sicherheit, domänenübergreifend modellieren (Meinecke, Nussbaumer und Gaedke, 2005).

Einen grundlegenden Beitrag stellt die Benutzungsschnittstelle als visuell geprägte Integrationsschicht zwischen den menschlichen Nutzern einer Web-Anwendung und den zugrunde liegenden Diensten der Referenzarchitektur dar. Eine Web-Anwendung gliedert Anwendungsdomänen mit Hilfe der eingeführten konzeptuellen Komponenten im formalen Modell des *Anwendungskontextes*. Die Übertragung der Dienstorientierung wird durch die Einführung von *Anwendungssichten* erreicht. Mit Hilfe von *Dekorateuren* werden bestimmte – dieser Sicht zugeordnete – fachliche Funktionen phasenspezifisch zusammengefasst und zu jedem Knoten des Anwendungskontexts hinzukonfiguriert. Zusätzliche Anwendungssichten können bedarfsgerecht hinzugefügt werden, um dezidiert Aspekte oder eine Phase einer Entwurfsmethodik zu unterstützen. Die dadurch erreichte Flexibilisierung des Anwendungsmodells ermöglicht eine prozessorientierte Evolution einer Web-Anwendung entlang dieser Anwendungssichten. Deren vielseitige Einsetzbarkeit wurde an den Aspekten Präsentation, Navigation und Dialog aufgezeigt, die dadurch einen orthogonalen Entwurf von Web-Anwendungen unter diesen Aspekten ermöglichen.

Die Präsentation von Informationen für den Benutzer wurde erstmalig aus zwei Blickwinkeln betrachtet. Zum einen in Form einer logisch strukturierten Präsentation mit Hilfe von Web-Standards auf der Grundlage von Domänen des Anwendungskontextes. Durch Einführung des *Namensraumkonzepts* konnte eine feingranulare Beschreibung der visuellen Ausprägung der Domänen erreicht werden, die in föderativen Nutzungsszenarien enormes Potenzial für die visuelle Überführung fremder Dienste in ein gemeinsames Corporate Design offeriert. Zum anderen wurde eine Methodik entwickelt, die eine automatisierte Überprüfung zur Gewährleistung des barrierefreien Zugriffs ermöglicht und mittels standardisierter Web-

Technologien umsetzt. Die Integration dieser Methodik in den Entwurfsprozess konnte durch eine Anwendungssicht erfolgreich realisiert werden. Dadurch können neben einer Qualitätssteigerung der Benutzungsschnittstelle durch eine verbesserte Zugänglichkeit erstmalig auch eine *proaktive Zugänglichkeit* in Form inhaltsbezogener Empfehlungen zum Abbau von Barrieren formuliert werden.

Das vorgestellte Navigationsmodellierungskonzept wurde explizit für die Dienstorientierung der Benutzungsschnittstelle konzipiert. Durch die Kapselung von Navigationsmustern in Form von konfigurierbaren Komponenten wurde erreicht, dass diese in unterschiedlichen Web-Anwendungen wieder verwendet werden können. Die Unterstützung ihrer Evolution wird gezielt gefördert, indem sich neue Navigationsstrukturen selektiv hinzufügen und entfernen lassen oder bestehende Varianten weiterentwickelt werden können. Auf Grundlage des DSL-Rahmenwerks wurde die UIRML (*UI Relationship Markup Language*)-DSL vorgestellt. Sie beruht auf dem Prinzip von Beziehungen zwischen Elementen der Benutzeroberfläche und dient einer formalen Beschreibung von Abhängigkeiten zwischen *Interaktionsträgern* sowie deren asynchronen Verknüpfungen mit Informationsraum-Diensten. Dadurch können Teile eines Formulars mit den Inhalten dieser Dienste effizient verknüpft werden, wodurch die Basis für innovative Dialogstrukturen gelegt werden konnte.

Die Evaluierung der beschriebenen Ansätze und Lösungsstrategien konnte in unterschiedlichen Projekten erfolgreich aufgezeigt werden. Dazu gehören neben international ausgerichteten Forschungskooperationen, wie mit der Universität Madrid oder dem Aufbau des Forschungs-Portals webengineering.org, auch zahlreiche Forschungs- und Entwicklungsprojekte, in denen der WSLS-Ansatz erfolgreich eingesetzt wurde.

Dazu gehörte das durch das BMBF geförderte Projekt *Notebook-Universität Karlsruhe (TH)*, in dem die Entwicklung und Evolution von anwendungsspezifischen Basisdiensten zur Unterstützung bei der Konstruktion ubiquitärer E-Learning Anwendungen verfolgt wurde. Im von Microsoft Research geförderten Nachfolge-Projekt *Living, Learning, and Teaching in Mobile Universities* (M-University) wurde die Entwicklung des erfolgreichen Konzepts der anwendungsspezifischen Basisdienste konsequent weitergeführt und deren Evolution in einer föderal-geprägten interorganisatorischen Wechselwirkung untersucht. Ein wichtiges Ziel stellt dabei die Fähigkeit einer M-University dar, diese Wechselwirkungen durch die Einführung eines Globalen Typ Systems mit kanonischen Schnittstellen ganzheitlich zu verbinden. Im Rahmen des durch das Land Baden-Württemberg geförderten Projekts *Karlsruher Integriertes InformationsManagement* wurde den Anforderungen an ein modernes geschäftsprozessorientiertes Informationsmanagement begegnet. Dabei stehen dienstorientierte Web-Anwendungen, die einen massiven Einsatz von Web Services basierend auf den entwickelten kanonischen Schnittstellen fokussieren, im Zentrum. Das durch Microsoft Research geförderte Projekt *Software Engineering for Information Appliances at Home* befasste sich mit den Anforderungen an die Benutzungsschnittstelle in Web-Anwendungen und dem Einsatz unterschiedlicher Ausgabemedien. Aufbauend auf dem WSLS-Ansatz entstanden eine Reihe dienstorientierter Web-Anwendungen mit dem Fokus der Visualisierung und Interaktion mit heterogenen Datenquellen im Web. Die dienstorientierte Modellierung der Benutzungsschnittstelle konnte mit Hilfe des WSLS-Ansatzes erfolgreich unter Beweis gestellt und am Beispiel eines Windows Media Centers demonstriert werden.

7.2 Weiterführende Arbeiten

Der entwickelte WSLS-Ansatz stellt eine effektive und effiziente Lösung zur Entwicklung und Evolution dienstorientierter Web-Anwendungen zur Verfügung. Durch seine prinzipielle Offenheit und Erweiterbarkeit bietet der neue Ansatz die Möglichkeit für weiterführende Arbeiten. Die Modellierung von Geschäftsprozessen in Form von Workflows stellt in Integrationsprojekten eine große Herausforderung dar. Während einfache Integrationsszenarien sich auf eine parametrisierte Kommunikation mit einzelnen Informationsraum-Diensten und der anschließenden Präsentation der zurück gelieferten Daten abbilden lassen, treten in der Praxis oft umfangreichere Abfolgen von Dialogen, Dienstkommunikation und Datenpräsentation auf. In Anbetracht dieser Komplexität sowie der Vielfalt von Diensten, die im Rahmen mittlerer und großer SOA-basierter Systeme bzw. Enterprise Application Integration (EAI) Projekte über dienstorientierte Web-Anwendungen den Nutzern zugänglich gemacht werden müssen, ist eine effiziente Methodik zur Entwicklung solcher Portalkomponenten notwendig. Erste vielversprechende Versuche konnten auf Basis des WSLS-Ansatzes bereits entwickelt und evaluiert werden (Freudenstein, Nussbaumer, Majer und Gaedke, 2007).

Weitere Arbeiten sind darüber hinaus nötig, um die mit dienstorienterten Web-Anwendungen verbundenen Anforderungen wie das effiziente Zusammenschalten dieser Komponenten, Föderation verteilter Workflows und die Betrachtung Benutzungsschnittstellen-relevanter Aspekte wie die Interaktion zu untersuchen. Aufgrund der starken Verbreitung unterschiedlich präferierter Modellierungsnotationen gepaart mit den unterschiedlich ausgeprägten Fertigkeiten in Projektteams bietet sich ein kommunikationsorientierter Ansatz auf Basis des DSL-Rahmenwerks an (Freudenstein, Buck, Nussbaumer und Gaedke, 2007).

Die in der Arbeit betrachtete Methodik zum Entwurf des Informationsraums stellt einen Ausgangspunkt zur Ableitung einer weitreichenden Unterstützung dar, die eine globale Vernetzung von Informationsraum-Diensten zum Ziel hat. Die dabei betrachtete organisationsübergreifende Nutzung von Diensten legt eine Zusammenführung mit Ansätzen des Grid Computing nahe. Während im Grid Computing ein Fokus auf Berechnungsstärke und massive Parallelisierbarkeit gelegt wird, eignen sich föderale Informationsraum-Dienste eines Globalen Typ Systems zur Realisierung und Gewährleistung der Kommunikation inner- und außerhalb des Grids. Grid-Ansätzen werden neben ihren klassischen Einsatzgebieten wie dem *High Performance Computing* auch zunehmend für Unternehmen attraktiver (Srinivasan und Treadwell, 2005). Dieses Aufkeimen weckt den Wunsch nach integrierten und vereinheitlichten Lösungen, die neben technischen auch organisatorische Problemstellungen wie Plattformunabhängigkeit, Verteilbarkeit, Lizenzierbarkeit, Sicherheit oder Föderierbarkeit in Betracht ziehen. Weiterführende Untersuchungen sind nötig, um eine gemeinsame, homogene Informationsbasis zu schaffen, die hilft, eine hocheffiziente, kosten-effektive und stark gekoppelte Grid-Lösung in eine wertschöpfende Prozesskette zu integrieren. Die in der vorliegenden Arbeit entwickelten Modelle, Methoden und Werkzeuge stellen dabei eine grundlegende Voraussetzung für diese Arbeiten dar.

Anhang

A Schema der Aspect Markup Language (AML)

```
01.  <?xml version="1.0" encoding="utf-8" ?>
02.  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
03.   <xsd:element name="Aspect">
04.    <xsd:complexType>
05.     <xsd:sequence>
06.      <xsd:element ref="Pointcut" minOccurs="1" maxOccurs="1" />
07.     </xsd:sequence>
08.     <xsd:attribute name="Name" type="xsd:string" use="required" />
09.     </xsd:complexType>
10.  </xsd:element>
11.  <xsd:element name="Pointcut">
12.    <xsd:complexType>
13.     <xsd:sequence>
14.      <xsd:element ref="Joinpoint" minOccurs="1" maxOccurs="1" />
15.      <xsd:element ref="Advice" minOccurs="1" maxOccurs="1" />
16.     </xsd:sequence>
17.    </xsd:complexType>
18.  </xsd:element>
19.  <xsd:element name="Joinpoint">
20.    <xsd:complexType>
21.     <xsd:simpleContent>
22.      <xsd:extension base="Joinpoints">
23.       <xsd:attribute name="Type" type="Joinpoint-Type" fixed="LifecycleMethod" />
24.       <xsd:attribute name="Scope" type="Scope-Type" fixed="Domain" />
25.      </xsd:extension>
26.     </xsd:simpleContent>
```

```
27.     </xsd:complexType>
28.   </xsd:element>
29.   <xsd:element name="Advice">
30.     <xsd:complexType>
31.       <xsd:simpleContent>
32.         <xsd:extension base="xsd:string">
33.           <xsd:attribute name="Type" type="Advice-Type" use="required" />
34.           <xsd:attribute name="Component" type="ComponentPlace-Type" fixed="Included" />
35.           <xsd:attribute name="ComponentType" type="ComponentType-Type" fixed="Source" />
36.         </xsd:extension>
37.       </xsd:simpleContent>
38.     </xsd:complexType>
39.   </xsd:element>
40.   <xsd:simpleType name="Joinpoint-Type">
41.     <xsd:restriction base="xsd:string">
42.       <xsd:enumeration value="LifecycleMethod" />
43.       <xsd:enumeration value="OtherMethod" />
44.       <xsd:enumeration value="Event" />
45.     </xsd:restriction>
46.   </xsd:simpleType>
47.   <xsd:simpleType name="Scope-Type">
48.     <xsd:restriction base="xsd:string">
49.       <xsd:enumeration value="Application" />
50.       <xsd:enumeration value="Domain" />
51.     </xsd:restriction>
52.   </xsd:simpleType>
53.   <xsd:simpleType name="Joinpoints">
54.     <xsd:restriction base="xsd:string">
55.     <xsd:enumeration value="Initialization" />
56.     <xsd:enumeration value="Mediation" />
57.     <xsd:enumeration value="Consolidation" />
58.     <xsd:enumeration value="Generation" />
59.     </xsd:restriction>
60.   </xsd:simpleType>
61.   <xsd:simpleType name="Advice-Type">
62.     <xsd:restriction base="xsd:string">
63.       <xsd:enumeration value="before" />
64.       <xsd:enumeration value="after" />
65.     </xsd:restriction>
```

```
66.    </xsd:simpleType>
67.    <xsd:simpleType name="ComponentPlace-Type">
68.      <xsd:restriction base="xsd:string">
69.        <xsd:enumeration value="Included" />
70.        <xsd:enumeration value="Referenced" />
71.      </xsd:restriction>
72.    </xsd:simpleType>
73.    <xsd:simpleType name="ComponentType-Type">
74.      <xsd:restriction base="xsd:string">
75.        <xsd:enumeration value="Binary" />
76.        <xsd:enumeration value="Source" />
77.      </xsd:restriction>
78.    </xsd:simpleType>
79.  </xsd:schema>
```

B OWL-Ontologie zum virtuellen Museum

```
01.  <rdf:RDFxmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
02.      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns="http://www.owl-ontologies.com/unnamed.owl#"
                xml:base="http://www.owl-ontologies.com/unnamed.owl">
03.    <owl:Ontology rdf:about="" />
04.    <owl:Class rdf:ID="Artist">
05.      <rdfs:subClassOf>
06.        <owl:Restriction>
07.          <owl:onProperty>
08.            <owl:ObjectProperty rdf:ID="livesIn" />
09.          </owl:onProperty>
10.          <owl:minCardinality
11.    rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
12.        </owl:Restriction>
13.      </rdfs:subClassOf>
14.      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
15.      <rdfs:subClassOf>
16.        <owl:Restriction>
17.          <owl:maxCardinality
       rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</owl:maxCardinality>
18.          <owl:onProperty>
```

```
19.            <owl:ObjectProperty rdf:about="#livesIn" />
20.          </owl:onProperty>
21.        </owl:Restriction>
22.      </rdfs:subClassOf>
23.      <rdfs:subClassOf>
24.        <owl:Restriction>
25.          <owl:minCardinality
     rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
26.          <owl:onProperty>
27.            <owl:ObjectProperty rdf:ID="creates" />
28.          </owl:onProperty>
29.        </owl:Restriction>
30.      </rdfs:subClassOf>
31.    </owl:Class>
32.    <owl:Class rdf:ID="Epoch">
33.      <rdfs:subClassOf>
34.        <owl:Restriction>
35.          <owl:minCardinality
     rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
36.          <owl:onProperty>
37.            <owl:ObjectProperty rdf:ID="hasPaintings" />
38.          </owl:onProperty>
39.        </owl:Restriction>
40.      </rdfs:subClassOf>
41.      <rdfs:subClassOf>
42.        <owl:Restriction>
43.          <owl:minCardinality
     rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
44.          <owl:onProperty>
45.            <owl:ObjectProperty rdf:ID="hasArtists" />
46.          </owl:onProperty>
47.        </owl:Restriction>
48.      </rdfs:subClassOf>
49.      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
50.    </owl:Class>
51.    <owl:Class rdf:ID="Painting">
52.      <rdfs:subClassOf id="test">
53.        <owl:Restriction>
54.          <owl:onProperty>
55.            <owl:ObjectProperty rdf:ID="createdBy" />
```

```
56.          </owl:onProperty>
57.          <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
58.         </owl:Restriction>
59.       </rdfs:subClassOf>
60.       <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
61.       <rdfs:subClassOf>
62.         <owl:Restriction>
63.           <owl:onProperty>
64.             <owl:ObjectProperty rdf:ID="createdIn" />
65.           </owl:onProperty>
66.           <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
67.         </owl:Restriction>
68.       </rdfs:subClassOf>
69.     </owl:Class>
70.     <owl:ObjectProperty rdf:about="#hasPaintings">
71.       <rdfs:domain rdf:resource="#Epoch" />
72.       <rdfs:range rdf:resource="#Painting" />
73.     </owl:ObjectProperty>
74.     <owl:ObjectProperty rdf:about="#createdBy">
75.       <rdfs:range rdf:resource="#Artist" />
76.       <rdfs:domain rdf:resource="#Painting" />
77.     </owl:ObjectProperty>
78.     <owl:ObjectProperty rdf:about="#creates">
79.       <rdfs:range rdf:resource="#Painting" />
80.       <rdfs:domain rdf:resource="#Artist" />
81.     </owl:ObjectProperty>
82.     <owl:ObjectProperty rdf:about="#hasArtists">
83.       <rdfs:range rdf:resource="#Artist" />
84.       <rdfs:domain rdf:resource="#Epoch" />
85.     </owl:ObjectProperty>
86.     <owl:ObjectProperty rdf:about="#createdIn">
87.       <rdfs:domain rdf:resource="#Painting" />
88.       <rdfs:range rdf:resource="#Epoch" />
89.     </owl:ObjectProperty>
90.     <owl:ObjectProperty rdf:about="#livesIn">
91.       <rdfs:domain rdf:resource="#Artist" />
92.       <rdfs:range rdf:resource="#Epoch" />
93.     </owl:ObjectProperty>
94.   </rdf:RDF>
```

C Schema der UIRML

Das folgende Schema beschreibt die User Interface Relationship Markup Language.

```
01.  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
02.  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
03.    <xs:element name="constraints" type="xs:string"/>
04.    <xs:element name="contexts" type="xs:string"/>
05.    <xs:element name="dependencyid" type="xs:string"/>
06.    <xs:element name="events" type="xs:string"/>
07.    <xs:element name="id" type="xs:string"/>
08.    <xs:element name="postprocessing" type="xs:string"/>
09.    <xs:element name="preprocessing" type="xs:string"/>
10.    <xs:element name="rootid" type="xs:string"/>
11.    <xs:element name="semantictype" type="xs:string"/>
12.    <xs:element name="servicemethod" type="xs:string"/>
13.    <xs:element name="element">
14.      <xs:complexType>
15.        <xs:sequence>
16.          <xs:element ref="id"/>
17.          <xs:element ref="semantictype"/>
18.        </xs:sequence>
19.      </xs:complexType>
20.    </xs:element>
21.    <xs:element name="template">
22.      <xs:complexType>
23.        <xs:sequence>
24.          <xs:element ref="id"/>
25.          <xs:element ref="rootid"/>
26.          <xs:element ref="dependencyid"/>
27.          <xs:element ref="events"/>
28.          <xs:element ref="constraints"/>
29.          <xs:element ref="contexts"/>
30.          <xs:element ref="preprocessing"/>
31.          <xs:element ref="servicemethod"/>
32.          <xs:element ref="postprocessing"/>
33.        </xs:sequence>
34.      </xs:complexType>
35.    </xs:element>
```

```
36.     <xs:element name="relation">
37.       <xs:complexType>
38.         <xs:sequence>
39.           <xs:element ref="id"/>
40.           <xs:element ref="rootid"/>
41.           <xs:element ref="dependencyid"/>
42.           <xs:element ref="events"/>
43.           <xs:element ref="constraints"/>
44.           <xs:element ref="contexts"/>
45.           <xs:element ref="preprocessing"/>
46.           <xs:element ref="servicemethod"/>
47.           <xs:element ref="postprocessing"/>
48.         </xs:sequence>
49.       </xs:complexType>
50.     </xs:element>
51.     <xs:element name="relationsroot">
52.       <xs:complexType>
53.         <xs:sequence>
54.           <xs:element name="elements">
55.             <xs:complexType>
56.               <xs:sequence>
57.                 <xs:element ref="element" maxOccurs="unbounded"/>
58.               </xs:sequence>
59.             </xs:complexType>
60.           </xs:element>
61.           <xs:element name="templates">
62.             <xs:complexType>
63.               <xs:sequence>
64.                 <xs:element ref="template" maxOccurs="unbounded"/>
65.               </xs:sequence>
66.             </xs:complexType>
67.           </xs:element>
68.           <xs:element name="relations">
69.             <xs:complexType>
70.               <xs:sequence>
71.                 <xs:element ref="relation" maxOccurs="unbounded"/>
72.               </xs:sequence>
73.             </xs:complexType>
74.           </xs:element>
```

```
75.           </xs:sequence>
76.         </xs:complexType>
77.       </xs:element>
78.     </xs:schema>
```

Index

A

abstrakter Endpunkt ... 31
Accessibility Testing ... 17
adäquate Darstellung ... 17, 129
adäquate Schnittstelle ... 6
ADV ... 49
Advice ... 120
agile Plattform ... 15
Anwendungskontext ... 10, 135, 176
Anwendungsraum ... 133
Anwendungssicht ... 142
Anwendungsspezifischer Basisdienst ... 106, 107, 108
Anwendungsstabilisierung ... 146
ASP.NET ... 111
Aspekt ... 110, 119
Aspektklassifikation ... 103
Aspekt-orientierte Anwendungsmodellierung ... 143
Aspekt-orientierte Komponentenevolution ... 119
Aspekt-orientierte Modellierung ... 107, 119
Aspekt-orientierte Programmierung ... 119
Ästhetik ... 10, 18, 151, 154
ATOM ... 155
Ausfaktorien von Übergabekontexten ... 75
Ausgabeschema ... 73
Auszeichnungssprache ... 148
authoring-in-the-large ... 44
authoring-in-the-small ... 44
Automatisierung der Barrierefreiheit ... 144
Autoweb ... 45

B

Barrierefreiheit ... 17, 130, 149
Barrierefreiheit als Qualitätsmerkmal ... 144
Baukastenprinzip ... 11, 97
Benutzerschema-Evolutionsmuster ... 52
Benutzungsschnittstelle ... 16, 17, 108, 129
  Domänen-Graph-Repräsentation ... 153
  Komponentenmodell ... 151
  p-Struktur-Schnittstelle ... 154
  s-Struktur-Schnittstelle ... 154
  Struktur-Schnittstelle ... 153
  Visualisierungsbaum ... 135
bijektive Informationsabbildung ... 64
BITV ... 130, 170
browsing semantic ... 45

C

CAWE ... 15, 96, 155
CBSE ... 105
CBWE ... 104, 110
Choreographie ... 77
compile-time weaving ... 121
composition framework ... 15
conditional guided tours ... 47
conditional indexed guided tours ... 47
conditional indices ... 47
ContentObject ... 66, 73
CRUDS-Schnittstelle ... 67, 69, 71, 74, 122, 175
CSS ... 149, 155

D

DAML ... 41
DAR ... 102
Darstellungsdesign ... 18
Darstellungsdynamik ... 17
datenorientierte Ansätze ... 43
Decorator Entwurfsmuster ... 142
Dekoration von Anwendungssichten ... 142
Dematerializer ... 49
dezentrales Informationsmedium ... 2
Dialog ... 11, 18, 93, 103, 129, 165
Dialog als Anwendungssicht ... 164
Dienstelement ... 97, 109
Dienstendpunkt ... 23
Dienstlandschaft ... 12
Dienstorientierte Anwendung ... 6
dienstorientierte Web-Anwendung ... 2, 7, 11
dienstorientierte Web-Anwendungen ... 21
Dienstorientierung ... 11, 12, 14
Dienstschnittstelle ... 23
Dienstzugangspunkt ... 16
DIM ... 101, 169

DOM ... 152
Domänenbaugruppe ... 107, 117, 136
Domänendienst ... 135
Domänenraum ... 133
domänenspezifische Maturität ... 137
domänen-spezifische Sprache (DSL) ... 101
Domänen-spezifische Sprache (DSL) ... 170
domänen-spezifisches Modell (DSM) ... 101
Domänen-spezifisches Modell (DSM) ... 168
DSL Analyse ... 100
DSL Entwicklung ... 100, 101
DSL Verwendung ... 100
DSL-Rahmenwerk ... 100
dynamische Anwendungen ... 7, 15

E

ebXML ... 33
Effektivität ... 1
Effizienz ... 1
Effizienz durch dedizierte Informationsversorgung ... 166
Enterprise Application Integration (EAI) ... 12, 79, 131
Enterprise Information Integration (EII) ... 12, 79
Entitätsklassen ... 44
Entity-Relationship Modell ... 47
Entwicklung für Wiederverwendung ... 107
Entwicklung mit Wiederverwendung ... 107
Entwicklungsprozess ... 8, 16, 55, 98, 99, 119
Event Condition Action (ECA) ... 167
Evolution . 1, 13, 14, 21, 22, 43, 45, 47, 50, 51, 54, 59, 61, 65, 77, 91, 102, 132, 174
Evolution existierender IR-Dienste ... 80
Evolutionsanalyse ... 9
Evolutionsdesign ... 9
Evolutionsdurchführung ... 9
experience economy ... 17

F

Fachkomponente ... 106, 114
fitness to purpose ... 108
Föderation ... 80
formale Zugänglichkeit durch Dekoration ... 145
Formatierungssprache ... 148
funktionale Prägung ... 106

G

generischer Andockpunkt ... 111
Globalen Typ System ... 61
Globales Typ System (GTS) ... 61, 73, 125, 175
Google Maps ... 18

H

HDM ... 44
HDM-Lite ... 45
Horizontale Evolution ... 67
Hyperanwendung ... 136, 137
Hypergraph ... 136
Hypermedia Design Pattern ... 10, 19
Hypertextmodell ... 48
hypertextorientierte Ansätze ... 43

I

Implementierungsmodell ... 46
Informationsföderation ... 13
Informationsmuster ... 52
Informationsraum ... 7, 12, 59, 63
Informationsraum Entwurf
  Anpassung und Test ... 88
  Datenmodellierung ... 85
  Generierung ... 88
  Ontologieerstellung ... 87
  Verteilung ... 89
  Wiederverwendung ... 86
Informationsraum-Dienst (IR-Dienst) ... 63, 78, 90
Informationsraum-Identifikator (ISID) ... 64, 69, 126
Inhalt ... 9, 93, 103, 125
Inhaltsisolierung ... 148
Interaktionsmuster ... 52
Interaktionsträger ... 167
Interaktive Web-Anwendung ... 11
Interoperabilität ... 31
Introspektion ... 115
IR Segment ... 77
ITX (Internet Transaction ID) ... 71

J

Joinpoint ... 120
Joint Application Development ... 15
JSR 168 ... 111

K

kanonische Dienstschnittstelle ... 63, 72, 84, 93, 113, 124, 163
KIM Projekt ... 94
kognitive Überlast ... 10, 18
Kommunikation ... 9, 93, 103, 126
Komplexitätsreduktion ... 10
Komponenten ... 44
Komponentendilemma ... 114
Komponentenevolution durch Generalisierung ... 115
Komponentenmaturität durch Aspekte ... 123
Komponentenmodell ... 105
Komponentenorientierung ... 14
Kompositionsmodell ... 48
Kompositionsstrukturdiagrammen ... 51
Konfiguration ... 112
Konfigurationsebene ... 133
  Editor ... 141
  Graph-Repräsentation ... 139
  Hüllenfunktion ... 134
  Klassifizierung ... 139
  Verwaisung ... 134
  Wertebereich ... 140
Konfigurierbare Konvertibilität ... 14
Kontextualität ... 10
Kontrollfunktion ... 109
Kontrollfunktion ... 97
konzeptuelle Entwurf ... 46

L

Learning Object Metadata (LOM) ... 126

Lebensyzklus
Finalisieren ... 113
Generieren ... 113
Initialisieren ... 112
Konsolodieren ... 112
Vermitteln ... 112
literale Dokumentenbindung ... 32
logischer Dokumentenaufbau ... 149
lose Kopplung ... 11, 13, 60

M

Makler ... 24
Materializer ... 49
Maturität ... 114, 138
Maximale Spezifizität ... 152
Mediation ... 107
Mediator Service ... 76
menschlicher Dienstnehmer ... 6
Microformat ... 150
Modellevolution ... 10
Multidisziplinarität ... 10
M-University Projekt ... 94
MVC^2 Architektur ... 164
MVC-Muster ... 165

N

Namensraum-Dekorateure ... 156
Namespace ... 40
Navigation ... 11, 18, 93, 103, 129, 165
Navigation als Anwendungssicht ... 157
navigation design ... 47
Navigation in Domänendiensten ... 161
Navigation in Strukturierungsdomänen ... 160
Navigationsdesign ... 18
Navigationsdiagramme (NAD) ... 52
Navigationsmodell ... 48
Nichtlinearität von Informationen ... 2
NUKATH Projekt ... 94
Nutzerinteraktion ... 16
Nutzerschnittstelle ... 174

O

objektiv automatisierbare Zugänglichkeit ... 145
objektorientierte Ansätze ... 43
OIL ... 41
Ontologie ... 38, 42, 87, 90, 162
OO-H ... 52
OOHDM ... 49
Orchestrierung ... 77

P

parametrische Polymorphie ... 74
parametrisierte Polymorphie ... 59
Periodizität ... 112
physische Ausprägung ... 65
Pointcut Designatoren ... 120
Präsentation ... 11, 93, 103, 129
Präsentation als Anwendungssicht ... 148
Präsentationsdiagramme (APD) ... 52
Präsentations-Evolution ... 17
Präsentationsisolierung ... 148
Präsentationsmodell ... 48, 51
Präsentationsschema ... 45
proaktive Zugänglichkeit ... 146, 177
Proxy ... 24
Prozess ... 9, 93, 103, 127
Prozessdelegation ... 127

R

RDF ... 40
RDFS ... 40, 41
Registrierung von IR-Diensten ... 82
Rich Internet Application (RIA) ... 17
RMM ... 47
RSS ... 155
runtime weaving ... 121

S

SearchContext ... 75
Semantic Web ... 38
semantische Konzeptualisierung ... 65
semi-transparente Verfestigung ... 124
Separation of Concerns ... 8, 10, 14, 114
ServiceCard ... 66
slice design ... 47
SOA ... 5, 12, 21, 22, 33
Diensterbringer ... 23
Dienstnehmer ... 24
service broker ... 24
SOAP ... 25, 60
body ... 28
Envelope ... 26
fault ... 28
Header ... 27
intermediary ... 27
Protokoll ... 28
Software Engineering ... 7
Solution Building Block (SBB) ... 100, 102, 106
STAIVE ... 67
structural links ... 47
Strukturierungsdomäne ... 135
Strukturmodell ... 48
Strukturraum ... 133
subjektiv automatisierbare Zugänglichkeit ... 145
syntaktische Konzeptualisierung ... 65
systematische Zugänglichkeit ... 144

T

Tagging-Architektur ... 117
technischer Dienstnehmer ... 6
transiente Maturierung ... 123
transparente Verfestigung ... 124
Trends ... 17
Trust ... 42
Trust-Layer
Trust ... 42
Tyrannei der Dominanten Dekomposition ... 114

U

ubiquitärer Zugriff ... 2

UDDI 33, 68, 70
bindingTemplate 34
categoryBag 35
Föderationsunterstützung 37
tModels 35
Übergabekontext 74
UIRML 167, 177
Unicode 40
URI 40
User Modeling 45
UUID 115
UWE 50

V

Verarbeitungsprozess 7, 127
verteilte Anwendung 2
Verteilung 112
Verteilung von IR-Diensten 82
vertikale Evolution 66, 67, 76
Verwebungs-Unterstützung 121
Visualisierungsbaum 135

W

W3C 148, 170
Wasserfallmodell 16
WCAG 145, 170
Web Engineering 7, 21
Web Krise 15
Web Ontology Language (OWL) 41
Web Service 22, 49, 60, 65, 68, 72, 75, 78, 83, 90, 94, 109, 126, 157, 175
Web-Anwendung 107, 131
WebComposition Prozess Modell 9, 95
Webepunkte 110
WebML 48, 153
Wiederverwendung .. 13, 61, 65, 77, 82, 87, 95, 100, 102, 114, 125, 129, 135, 152, 156
Wiederverwendungs-Bibliothekar 104
wiederverwendungsorientierte Entwicklungsmodelle . 13
Wiederverwendungs-Repositorium 103
Windows Media Center Projekt 95
Wohldefinierte Verknüpfungspunkte 120
Workflow-basierte Anwendungen 48
WSDL 29, 72
address 29
binding 29
contract 29
Contract-first 29
definitions 30
message parts 30
messages 30
operation 31
RPC-Bindung 31
types 30
WSDL-first 29
WSDL-S 37
WSDM 45
WS-I 31, 109
WSLS 93, 175
WSLS Domäne 106
WSLS Komponentenevolution 114
WSLS Lebenszyklus 110
WSLS Referenzarchitektur 104
WSLS-Suite 94
WSRP 67, 111

X

XML 40
XML Schema 40
XPath 145
XSL(T) 146

Literaturverzeichnis

(Akanda und German, 2005) M. A. K. Akanda und D. M. German, *A System of Patterns for Web Navigation*, In Proceedings of 5th International Conference of Web Engineering (ICWE 2005), Sydney, Australia, Springer, 2005.

(Akhilesh und Ramayya, 2002) B. Akhilesh und K. Ramayya, *CMU-WEB: a conceptual model with metrics for testing and designing usability in Web applications***,** Advanced topics in database research vol. 1 %@ 1-930708-41-6. Idea Group Publishing, Seiten: 230-249, International Standard Book 2002.

(Allerding, 2007) F. Allerding, *Modellgetriebene Entwicklung dynamischer Benutzerschnittstellen im Web Engineering***,** Karlsruhe,Universität Karlsruhe, 2007.

(Alonso, Casati, Kuno und Machiraju, 2003) G. Alonso, F. Casati, H. Kuno und V. Machiraju, *Web Services - Concepts, Architectures and Applications*, Springer, International Standard Book Number: 3540440089, 2003.

(Alt, Heutschi und Österle, 2003) R. Alt, R. Heutschi und H. Österle, *WebServices - Hype oder Lösung?***,** io new management,71 (02/03): 63-70, International Standard Serial Number R. Alt, R. Heutschi und H. Österle, 2003.

(Andre, Barthelmeß, Brauch und Deussen, 2004) O. Andre, H. Barthelmeß, A. Brauch und P. Deussen, *Mobile Lehr- und Lernszenarien***,** Nukath - Die Notebook-Universität Karlsruhe (TH). P. Deussen, W. Juling and B. Thum, Karlsruhe University Press, Seiten: 14-25, International Standard Book Number: 3-937300-01-5, 2004.

(Andresen, 2003) L. Andresen, *Dublin Core Metadata Element Set, Version 1.1: Reference Description*, http://dublincore.org/documents/dces/, 2003.

(Antoniou und Harmelen, 2004) G. Antoniou und F. v. Harmelen, *A Semantic Web Primer*, MIT Press, International Standard Book Number: 0262012103, 2004.

(Aoyama, 2005) M. Aoyama, *Persona-and-scenario based requirements engineering for software embedded in digital consumer products*, In Proceedings of 13th IEEE International Conference on Requirements Engineering, Paris, France, IEEE, 2005.

(Baader et al., 2003) F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi und P. F. Patel-Schneider, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi und P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Imple-*

mentation, and Applications, Description Logic Handbook, Cambridge University Press, International Standard Book Number: 0-521-78176-0, 2003.

(Bailey, Poulovassilis und Wood, 2002) J. Bailey, A. Poulovassilis und P. T. Wood, *An event-condition-action language for XML***,** Honolulu, Hawaii, USA,ACM Press, 2002.

(Bajaj et al., 2003) S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, H. Lockhart, H. Maruyama, N. Nagaratnam, A. Nash, H. Prafullchandra und J. Shewchuk, *Web Services Federation Language (WS-Federation)*, http://www-106.ibm.com/developerworks/webservices/library/ws-fed/, 2003.

(Ballinger et al., 2006) K. Ballinger, D. Ehnebuske, C. Ferris, M. Gudgin, C. K. Liu, M. Nottingham und P. Yendluri, *Basic Profile Version 1.1*, http://www.ws-i.org/Profiles/BasicProfile-1.1.htm, 2006.

(Balzert, 1996) H. Balzert, *Lehrbuch der Software-Technik: Software-Entwicklung*, Heidelberg,Spektrum Akademischer Verlag, International Standard Book Number: 3-8274-0042-2, 1996.

(Barkmeyer et al., 2003) E. J. Barkmeyer, A. Barnard Feeney, P. Denno, D. W. Flater, M. Potts Steves und E. K. Wallace, *Concepts for Automating Systems Integration*, National Institute of Standards and Technology, Gaithersburg, 2003.

(Bechhofer et al., 2004) S. Bechhofer, F. v. Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider und L. A. Stein, *OWL Web Ontology Language Reference*, http://www.w3.org/TR/2004/REC-owl-ref-20040210/, 2004.

(Becker und Mottay, 2001) S. A. Becker und F. E. Mottay, *A Global Perspective on Web Site Usability***,** IEEE Software,18 (1): 54-61, International Standard Serial Number S. A. Becker und F. E. Mottay, 2001.

(Behrendt, 2003) W. Behrendt, *Semantisches Web - Das Netz der Bedeutungen im Netz der Dokumente***,** Web Engineering - Web Engineering - Systematische Entwicklung von Web Anwendungen. Gerti Kappel , Birgit Pröll, Siegried Reich and W. Retschitzegger, Heidelberg, dPunkt Verlag, Seiten: 345-367, International Standard Book Number: 3-89864-234-8, 2003.

(Bellwood et al., 2002) T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter und C. von Riegen, *Universal Description Discovery and Integration (UDDI) Version 3.0*, UDDI.org, uddi-v3.00-published-20020719, 2002.

(Bentley, 1986) J. L. Bentley, *Programming pearls: Little languages***,** Communications of the ACM,29 (8): 711-721, International Standard Serial Number J. L. Bentley, 1986.

(Berglund et al., 2007) A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie und J. Siméon, *XML Path Language (XPath) Version 2.0*, http://www.w3.org/TR/xpath20/, 2007.

(Berners-Lee, 1990) T. Berners-Lee, *Information Management: A Proposal*, http://www.w3.org/Proposal.html, 1990.

(Berners-Lee, 1996) T. Berners-Lee, *Universal Resource Identifiers -- Axioms of Web Architecture*, http://www.w3.org/DesignIssues/Axioms.html, 1996.

(Berners-Lee et al., 1994) T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen und A. Secret, *The World-Wide Web***,** Communications of the ACM,37 (8): 76-82, International Standard Serial Number Number: 0001-0782, T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen und A. Secret, 1994.

(Berners-Lee, Fielding, Irvine und Masinter, 1998) T. Berners-Lee, R. Fielding, U. C. Irvine und L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*, IETF, Network Working Group, 2396, 1998.

(Berners-Lee, Hendler und Lassila, 2001a) T. Berners-Lee, J. Hendler und O. Lassila, *The Semantic Web***,** Scientific American,(May 2001): International Standard Serial Number Number: 0036-8733, T. Berners-Lee, J. Hendler und O. Lassila, 2001a.

(Berners-Lee, Hendler und Lassila, 2001b) T. Berners-Lee, J. Hendler und O. Lassila, *Semantic Web*, http://www.w3.org/2001/sw/, 2001b.

(Birbeck et al., 2006) M. Birbeck, S. McCarron, S. Pemberton, T. V. Raman und R. Schwerdtfeger, *XHTML Role Attribute Module*, http://www.w3.org/TR/2006/WD-xhtml-role-20061113, 2006.

(Biron und Ashok, 2004) P. V. Biron und M. Ashok, *XML Schema Part 2: Datatypes Second Edition*, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html, 2004.

(Biscotti und Correia, 2006) F. Biscotti und J. M. Correia, *Forecast: AIM and Portal Software, Worldwide, 2005-2010, Update (Executive Summary)*, Gartner Research, Stamford, CT, G00141919, 2006.

(Bonifati, Ceri und Paraboschi, 2001) A. Bonifati, S. Ceri und S. Paraboschi, *Pushing reactive services to XML repositories using active rules***,** Hong Kong, Hong Kong,ACM Press, 2001.

(Bonn, Dieter und Schmeck, 2004) M. Bonn, S. Dieter und H. Schmeck, *Anwendungen mobiler Systeme***,** Nukath - Die Notebook-Universität Karlsruhe (TH). P. Deussen, W. Juling and B. Thum, Karlsruhe University Press, Seiten: 37-51, International Standard Book Number: 3-937300-01-5, 2004.

(Booth et al., 2003) D. Booth, M. Champion, C. Ferris, F. McCabe, E. Newcomer und D. Orchard, *Web Services Architecture*, http://www.w3.org/TR/ws-arch/, 2003.

(Borzo, 2005) J. Borzo, *Business 2010: Embracing the challenge of the change, Report from the Economist Intelligence Unit*, SAP, London, 2005.

(Bos, Çelik, Hickson und Lie, 1998) B. Bos, T. Çelik, I. Hickson und H. W. Lie, *CSS2 Specification*, http://www.w3.org/TR/CSS21/, 1998.

(Böttger, 2006) M. Böttger, *Workflow-Integration mit Web Services***,** Karlsruhe,Universität Karlsruhe, 2006.

(Box et al., 2000) D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte und D. Winer, *Simple Object Access Protocol (SOAP) 1.1*, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2000.

(Brambilla et al., 2003) M. Brambilla, S. Ceri, S. Comai, P. Fraternali und I. Manolescu, *Model-driven Development of Web Services and Hypertext Applications*, In Proceedings of SCI2003, Orlando, Florida, 2003.

(Braun, 1994) C. L. Braun, *NATO standard for the development of reusable software components*, Vol. 3, http://wuarchive.wustl.edu/languages/ada/docs/nato_ru/, 1994.

(Bray, Hollander und Layman, 1999) T. Bray, D. Hollander und A. Layman, *Namespaces in XML*, World Wide Web Consortium, 1999.

(Bray et al., 2006) T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler und F. Yergeau, *Extensible Markup Language (XML) 1.0*, http://www.w3.org/TR/xml, 2006.

(Brewington und Cybenko, 2000) B. E. Brewington und G. Cybenko, *How dynamic is the Web?*, In Proceedings of 9th international World Wide Web conference (WWW9), Amsterdam, The Netherlands, North-Holland Publishing Co., 2000.

(Brickley und Guha, 2004) D. Brickley und R. V. Guha, *RDF Vocabulary Description Language 1.0: RDF Schema*, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2004.

(Brockmans, Volz, Eberhart und Löffler, 2004) S. Brockmans, R. Volz, A. Eberhart und P. Löffler, *Visual Modeling of OWL DL Ontologies Using UML***,** Hiroshima ,Japan,Springer, 198-213, 2004.

(Brown, 1996) A. W. Brown, A. W. Brown, *Component-Based Software Engineering: Selected Papers from the Software Engineering Institute*, Wiley-IEEE Computer Society Press, International Standard Book Number: 978-0-8186-7718-2, 1996.

(Brown und Ellis, 2004) K. Brown und M. Ellis, *Best practices for Web services versioning*, http://www-128.ibm.com/developerworks/webservices/library/ws-version/, 2004.

(Budgen, 2003) D. Budgen, *David Budgen* Addison Wesley, International Standard Book Number: 978-0201722192, 2003.

(Bundesministerium der Justiz, 2002) Bundesministerium der Justiz, *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz*, http://bundesrecht.juris.de/bundesrecht/bitv/gesamt.pdf, 2002.

(Butek, 2005) R. Butek, *Which style of WSDL should I use?*, http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl/, 2005.

(Carlson, 2001) D. Carlson, *Modeling XML Applications with UML*, Addison-Wesley, International Standard Book Number: 0-201-70915-5, 2001.

(Casteleyn, Troyer und Brockmans, 2003) S. Casteleyn, O. d. Troyer und S. Brockmans, *Design time support for adaptive behavior in Web sites*, In Proceedings of 2003 ACM Symposium on Applied Computing, Melbourne, Florida, ACM Press, 2003.

(Centeno, Kloos, Gaedke und Nussbaumer, 2006) V. L. Centeno, C. D. Kloos, M. Gaedke und M. Nussbaumer, *WAEX: Web Accessibility Evaluator in a Single XSLT File*, In Proceedings of 2nd Int'l. Workshop on Automated Specification and Verification of Web Systems (WWV'06), Cyprus, IEEE, 2006.

(Ceri, Fraternali und Bongio, 2000) S. Ceri, P. Fraternali und A. Bongio, *Web Modeling Language (WebML): A Modeling Language for Designing Web Sites*, In Pro-

ceedings of 9th International World Wide Web Conference (WWW), Amsterdam, Nethderlands, 2000.

(Chen, 1976) P. P.-S. Chen, *The Entity-Relationship Model - Toward a Unified View of Data***,** ACM Transactions on Database-Systems,1 (1): 9-36, International Standard Serial Number P. P.-S. Chen, 1976.

(Chinnici, Gudgin, Moreau und Weerawarana, 2003) R. Chinnici, M. Gudgin, J.-J. Moreau und S. Weerawarana, *Web Services Description Language (WSDL) Version 1.2*, http://www.w3.org/TR/2003/WD-wsdl12-20030303/, 2003.

(Chisholm, Vanderheiden und Jacobs, 1999) W. Chisholm, G. Vanderheiden und I. Jacobs, *Web Content Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-WEBCONTENT/, 1999.

(Chiusano, Schmitt und Crawford, 2001) J. Chiusano, T. Schmitt und M. Crawford, *Requirements for an XML Registry*, http://xml.coverpages.org/LMI-registryreport.pdf, 2001.

(Clark, 1999) J. Clark, *XSL Transformations (XSLT), Version 1.0, W3C Recommendation*, http://www.w3.org/TR/xslt, 1999.

(Clarke, 2002) S. Clarke, *Extending standard UML with model composition semantics***,** Sci. Comput. Program.,44 (1): 71-100, International Standard Serial Number Number: 0167-6423, S. Clarke, 2002.

(Coallier, 1999) F. Coallier, *Interface Definition Language*, ISO, ISO/IEC 14750:1999, 1999.

(Constantine und Lockwood, 2001) L. L. Constantine und L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley, International Standard Book Number: 0201924781, 2001.

(Cover, 2000) R. Cover, *SGML: General Introductions and Overviews*, http://www.oasis-open.org/cover/general.html, 2000.

(Cowan und Lucena, 1995) D. D. Cowan und C. J. P. Lucena, *Abstract data views: an interface specification concept to enhance design for reuse***,** Software Engineering, IEEE Transactions on,21 (3): 229-243, International Standard Serial Number D. D. Cowan und C. J. P. Lucena, 1995.

(Cristina Cachero und Koch, 2002) Cristina Cachero und N. Koch, *Navigation Analysis and Navigation Design in OO-H and UWE*, Institute of Computer Science, Ludwig-Maximilians University of Munich, Technical Report 0205, Munich, Technical Report 0205, 2002.

(Cutter Consortium, 2000) Cutter Consortium, *Poor Project Management Number-One Problem of Outsourced E-Projects*, 2000.

(Daconta, Smith und Obrst, 2003) M. C. Daconta, K. T. Smith und L. J. Obrst, *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, John Wiley & Sons, Inc., International Standard Book Number: 0471432571, 2003.

(Dart, 2001) S. Dart, *Configuration Management: The Missing Link in Web Engineering*, Artech House Publishers, International Standard Book Number: 1580530982, 2001.

(Dawson und Howes, 1998) F. Dawson und T. Howes, *vCard MIME Directory Profile*, RFC 2426, 2426, 1998.

(Dayal, 1988) U. Dayal, *Active Database Management Systems***,** Conference on Data and Knowledge Bases. Morgan Kaufmann Publishers, Seiten: 150-169, International Standard Book Number: 0934613958, 1988.

(DCMI, 2006) DCMI, *DCMI Type Vocabulary*, http://dublincore.org/documents/2006/08/28/dcmi-type-vocabulary/, 2006.

(De Troyer und Leune, 1998) O. M. F. De Troyer und C. J. Leune, *WSDM: a user centered design method for Web sites***,** Computer Networks and ISDN Systems,30(1998) (Special Issue on the 7th Intl. World-Wide Web Conference, Brisbane, Australia): International Standard Serial Number O. M. F. De Troyer und C. J. Leune, 1998.

(Deshpande et al., 2002) Y. Deshpande, S. Murugesan, A. Ginige, S. Hansen, D. Schwabe, M. Gaedke und B. White, *Web Engineering***,** Journal of Web Engineering,1 (1): 3-17, International Standard Serial Number Y. Deshpande, S. Murugesan, A. Ginige, S. Hansen, D. Schwabe, M. Gaedke und B. White, 2002.

(Deursen, Klint und Visser, 2000) A. v. Deursen, P. Klint und J. Visser, *Domain-Specific Languages: An Annotated Bibliography***,** ACM SIGPLAN Notices,35 (6): 26-36, International Standard Serial Number A. v. Deursen, P. Klint und J. Visser, 2000.

(Deussen, Juling und Thum, 2004) P. Deussen, W. Juling und B. Thum, P. Deussen, W. Juling und B. Thum, *Nukath - Die Notebook-Universität Karlsruhe (TH)*, Nukath - Die Notebook-Universität Karlsruhe (TH), Karlsruhe, Karlsruhe University Press, International Standard Book Number: 3-937300-01-5, 2004.

(Dijkstra, 1982a) E. W. Dijkstra, *How do we tell truths that might hurt?***,** SIGPLAN Notices,17 (5): 13-15, International Standard Serial Number E. W. Dijkstra, 1982a.

(Dijkstra, 1982b) E. W. Dijkstra, *On the role of scientific thought***,** Selected Writings on Computing: A Personal Perspective. New York, Srpinger-Verlag, Seiten: 60-66, International Standard Book Number: 0-387-90625-5, 1982b.

(Dostal, M., Melzer und Zengler, 2005) W. Dostal, J. M., I. Melzer und B. Zengler, *Service-orientierte Architekturen mit Web Services. Konzepte - Standards - Praxis* Spektrum Akademischer Verlag, International Standard Book Number: 3-8274-1457-1, 2005.

(Elrad, Aldawud und Bader, 2002) T. Elrad, O. Aldawud und A. Bader, *Aspect-Oriented Modeling: Bridging the Gap between Implementation and Design*, In Proceedings of 1st ACM SIGPLAN/SIGSOFT conference on Generative Programming and Component Engineering, Pittsburgh, PA, Springer-Verlag, 2002.

(Engelen, Zhang und Govindaraju, 2006) R. V. Engelen, G. Zhang und M. Govindaraju, *Toward remote object coherence with compiled object serialization for distributed computing with xml web services*, In Proceedings of Compilers for Parallel Computing (CPC), Coruna, Spain, 2006.

(Estublier, 2000) J. Estublier, *Software configuration management: a roadmap.*, In Proceedings of ICSE - Future of SE Track, 2000.

(Fettke, Loos und Tann, 2001) P. Fettke, P. Loos und M. v. d. Tann, *Eine Fallstudie zur Spezifikation von Fachkomponenten eines Informationssystems für Virtuelle Finanzdienstleister – Beschreibung und Schlussfolgerungen*, In Proceedings of 2. Workshop Modellierung und Spezifikation von Fachkomponenten, Bamberg, 2001.

(Filman, Elrad, Clarke und Aksit, 2004) R. E. Filman, T. Elrad, S. Clarke und M. Aksit, *Aspect-Oriented Software Development*, Addison-Wesley Professional, International Standard Book Number: 978-0321219763, 2004.

(Filman und Friedman, 2005) R. E. Filman und D. P. Friedman, *Aspect-Oriented Programming Is Quantification and Obliviousness***,** R. E. Filman, T. Elrad, S. Clarke and M. Aksit, Boston, Addison-Wesley, Seiten: 21-35, International Standard Book Number: 0-321-21976-7, 2005.

(Fraternali und Paolini, 1998) P. Fraternali und P. Paolini, *A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications*, In Proceedings of 6th International Conference on Extending Database Technology: Advances in Database Technology, Springer-Verlag, 1998.

(Fraternali und Paolini, 2000) P. Fraternali und P. Paolini, *Model-driven Development of Web Applications: The AutoWeb System***,** ACM Trans. Inf. Syst.,18 (4): 323-382, International Standard Serial Number Number: 1046-8188, P. Fraternali und P. Paolini, 2000.

(Freudenstein, Buck, Nussbaumer und Gaedke, 2007) P. Freudenstein, J. Buck, M. Nussbaumer und M. Gaedke, *Model-driven Construction of Workflow-based Web Applications with Domain-specific Languages*, In Proceedings of 3rd Workshop on Model-driven Web Engineering (MDWE'07), Como, Italy, 2007.

(Freudenstein, Nussbaumer, Majer und Gaedke, 2007) P. Freudenstein, M. Nussbaumer, F. Majer und M. Gaedke, *A Workflow-Driven Approach for the Efficient Integration of Web Services in Portals*, In Proceedings of IEEE International Conference on Services Computing (SCC 2007), Salt Lake City, USA, 2007.

(Fritz, 2004) F.-J. Fritz, *An Introduction to the Principles of Enterprise Services Architecture (ESA)*, http://www.sapinsideronline.com/spijsp/article.jsp?article_id=37906&volume_id=51, 2004.

(G. Costagliola et al., 2004) G. Costagliola, F. Ferrucci, C. Gravino, G. Tortora und G. Vitiello, *The Impact of Accessibility and Usability on the Development of Web Applications*, In Proceedings of First International Workshop on Web Quality (WQ 2004), Munich, Germany, Rinton Press, 2004.

(G. Kappel, B. Pröll, S. Reich und Retschitzegger, 2003) G. Kappel, B. Pröll, S. Reich und W. Retschitzegger, *Web Engineering – Die Disziplin zur systematischen Entwicklung von Web-Anwendungen***,** Web Engineering - Web Engineering - Systematische Entwicklung von Web Anwendungen. Gerti Kappel , Birgit Pröll, Siegried Reich and W. Retschitzegger, Heidelberg, dPunkt Verlag, Seiten: 1-28, International Standard Book Number: 3-89864-234-8, 2003.

(Gaedke, 1998) M. Gaedke, *WebComposition: Ein Unterstützungssystem für das Web Engineering***,** GI Softwaretechnik-Trends,18 (3): 20-25, International Standard Serial Number M. Gaedke, 1998.

(Gaedke, 2000) M. Gaedke, *Komponententechnik für Entwicklung und Evolution von Anwendungen im World Wide Web*, Aachen,Shaker Verlag, International Standard Book Number: 3-8265-8059-1, 2000.

(Gaedke und Graef, 2000) M. Gaedke und G. Graef, *WebComposition Process Model: Ein Vorgehensmodell zur Entwicklung und Evolution von Web-Anwendungen*, In Proceedings of 2. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 2), Wien, Austria, Wirtschaftsuniversität Wien, 2000.

(Gaedke, Juling und Nussbaumer, 2004) M. Gaedke, W. Juling und M. Nussbaumer, *Anwendungsspezifische Basisdienste***,** Nukath - Die Notebook-Universität Karlsruhe (TH). P. Deussen, W. Juling and B. Thum, Karlsruhe University Press, Seiten: 79-91, International Standard Book Number: 3-937300-01-5, 2004.

(Gaedke, Meinecke und Heil, 2006) M. Gaedke, J. Meinecke und A. Heil, *FDX - Federating Devices and Web Applications*, In Proceedings of Sixth International Conference for Web Engineering (ICWE2006), Palo Alto, USA, ACM Press, 2006.

(Gaedke, Meinecke und Nussbaumer, 2004a) M. Gaedke, J. Meinecke und M. Nussbaumer, *Security and Licensing for Components of Web-based Information Systems*, In Proceedings of 6th International Conference on Information Integration and Web Based Applications & Services (iiWAS2004), Jakarta, Indonesia, Austrian Computer Society, 2004a.

(Gaedke, Meinecke und Nussbaumer, 2004b) M. Gaedke, J. Meinecke und M. Nussbaumer, *Supporting Secure Deployment of Portal Components*, In Proceedings of 4th Int. Conference on Web Engineering (ICWE 2004), Munich, Germany, Springer, 2004b.

(Gaedke, Meinecke und Nussbaumer, 2005) M. Gaedke, J. Meinecke und M. Nussbaumer, *i2Map: An Approach To Model The Landscape Of Federated Systems*, In Proceedings of IEEE International Conference on Web Services (ICWS 2005), Orlando, USA, IEEE Computer Society, 2005.

(Gaedke und Nussbaumer, 2002) M. Gaedke und M. Nussbaumer, *Formularbasierte Benutzerinteraktion mit Fachkomponenten*, In Proceedings of 4. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 4), Augsburg, 2002.

(Gaedke, Nussbaumer und Meinecke, 2004) M. Gaedke, M. Nussbaumer und J. Meinecke, *WSLS: A Service-Based System for Reuse-Oriented Web Engineering*, In Proceedings of Fourth Int. Workshop on Web-oriented Software Technology (IWWOST 2004), Munich, Germany, Rinton Press, 2004.

(Gaedke, Nussbaumer und Meinecke, 2005) M. Gaedke, M. Nussbaumer und J. Meinecke, *WSLS: An Agile System Facilitating the Production of Service-Oriented Web Applications***,** Engineering Advanced Web Applications. S. C. M. Matera, Rinton Press, Seiten: 26-37, International Standard Book Number: 1-58949-046-0, 2005.

(Gaedke und Rehse, 2000) M. Gaedke und J. Rehse, *Supporting Compositional Reuse in Component-Based Web Engineering*, In Proceedings of 2000 ACM Sympo-

sium on Applied Computing (SAC 2000), Villa Olmo, Como, Italy, ACM, 2000.

(Gaedke, Rehse und Graef, 1999) M. Gaedke, J. Rehse und G. Graef, *A Repository to facilitate Reuse in Component-Based Web Engineering*, In Proceedings of International Workshop on Web Engineering at the 8th International World-Wide Web Conference (WWW8), Toronto, Ontario, Canada, 1999.

(Gaedke, Turowski und Rehse, 1999) M. Gaedke, K. Turowski und J. Rehse, *Föderierung betrieblicher Anwendungssysteme auf der Grundlage Web-basierter Dienste*, In Proceedings of 4. Workshop Föderierte Datenbanken, Berlin, Technische Universität Berlin, 1999.

(Gamma, Helm, Johnson und Vlissides, 1995) E. Gamma, R. Helm, R. Johnson und J. Vlissides, *Design patterns: elements of reusable object-oriented software*, Reading, Mass.,Addison-Wesley, International Standard Book Number: 0201633612, 1995.

(Garrett, 2003) J. J. Garrett, *The Elements of User Experience - User-Centered Design for the Web*, New Riders Press, International Standard Book Number: 0735712026, 2003.

(Gasevic, Djuric, Devedzic und Damjanovi, 2004) D. Gasevic, D. Djuric, V. Devedzic und V. Damjanovi, *Converting UML to OWL ontologies***,** New York, NY, USA,ACM Press, 2004.

(Gellersen und Gaedke, 1998) H.-W. Gellersen und M. Gaedke, *An Object-Oriented Model (not only) for Hypertext in the Web*, In Proceedings of HTF5: The Fifth International Workshop on Engineering Hypertext Functionality into Future Information Systems, The 20th International Conference on Software Engineering (ICSE), Kyoto International Conference Hall, Kyoto, Japan, 1998.

(Ginige und Murugesan, 2001) A. Ginige und S. Murugesan, *Guest Editors' Introduction: Web Engineering: An Introduction***,** IEEE MultiMedia,8 (1): 14-18, International Standard Serial Number Number: 1070-986X, A. Ginige und S. Murugesan, 2001.

(Goland et al., 1999) Y. Goland, E. Whitehead, A. Faizi, S. Carter und D. Jensen, *RFC 2518: HTTP Extensions for Distributed Authoring -- WEBDAV*, IETF, Network Working Group, 2518, 1999.

(Gomez, Cachero und Pastor, 2001) j. Gomez, C. Cachero und O. Pastor, *Conceptual Modeling of Device-Independent Web Applications***,** IEEE MultiMedia,8 (2): 26-39, International Standard Serial Number Number: 1070-986X, j. Gomez, C. Cachero und O. Pastor, 2001.

(Goos und Zimmermann, 2005) G. Goos und W. Zimmermann, *Vorlesungen über Informatik 1. Grundlagen und funktionales Programmieren: Band 1: Grundlagen Und Funktionales Programmieren*, Berlin,Springer, Berlin, International Standard Book Number: 3540244050, 2005.

(Gootzit und Phifer, 2003) D. Gootzit und G. Phifer, *Gen-4 Portal Functionality: From Unification to Federation*, Stamford, CT, 2003.

(Gruber, 1993) T. R. Gruber, *A translation approach to portable ontology specifications***,** Knowledge Acquisition 5 (2): 199-220, International Standard Serial Number Number: 1042-8143, T. R. Gruber, 1993.

(Halin, 1989) R. Halin, *Graphentheorie* Wissenschaftliche Buchgesellschaft, International Standard Book Number: 3534101405, 1989.

(Hayward, 2005) S. Hayward, *Positions 2005: Service-Oriented Architecture Adds Flexibility to Business Processes*, Gartner Research, Stamford, CT, G00126409, 2005.

(Henkel und J, 2005) M. Henkel und Z. J, *Approaches to Service Interface Design.*, In Proceedings of Workshop "Web services and Interoperability", Valencia, Spain, Hermes Science Publishing, 2005.

(Henninger, 1994) S. Henninger, *Supporting the Construction and Evolution of Component Repositories*, In Proceedings of 18th International Conference on Software Engineering (ICSE), Berlin, Germany, ACM Press, 1994.

(Hodgins und Duval, 2005) W. Hodgins und E. Duval, *Draft Standard for Learning Object Metadata*, http://ltsc.ieee.org/wg12/, 2005.

(Howard, 2004) R. Howard, *Provider Model Design Pattern and Specification, Part 1*, http://msdn2.microsoft.com/en-us/library/ms972319.aspx 2004.

(IBM, 2006) IBM, *Information service patterns, Part 1: Data federation pattern*, http://www-128.ibm.com/developerworks/webservices/library/ws-soa-infoserv1/, 2006.

(IBM und Sandpiper Software Inc., 2006) IBM und Sandpiper Software Inc., *Ontology Definition Metamodel*, http://www.omg.org/docs/ad/06-05-01.pdf, 2006.

(Imbusch, Langhammer und Walter, 2005) O. Imbusch, F. Langhammer und G. v. Walter, *Ercatons and organic programming: say good-bye to planned economy***,** San Diego, CA, USA,ACM Press, 2005.

(Isakowitz, Kamis und Koufaris, 1998) T. Isakowitz, A. Kamis und M. Koufaris, *Reconciling top-down and bottom-up design approaches in RMM***,** SIGMIS Database,29 (4): 58-67, International Standard Serial Number Number: 0095-0033, T. Isakowitz, A. Kamis und M. Koufaris, 1998.

(Isakowitz, Stohr und Balasubramaninan, 1995) T. Isakowitz, E. A. Stohr und P. Balasubramaninan, *RMM: A Methodology for Structured Hypermedia Design***,** Communications of the ACM,38, No. 8 (Aug. 1995): 34-44, International Standard Serial Number T. Isakowitz, E. A. Stohr und P. Balasubramaninan, 1995.

(ITAW, 1998) ITAW, *U.S. Section 508 Guidelines*, Office of Governmentwide Policy - U.S. General Services Administration, http://www.section508.gov/, 1998.

(Janus, 2006) A. Janus, *Konzepte der Aspekt-orientierten Programmierung im Komponenten-basierten Web Engineering***,** Karlsruhe,Universität Karlsruhe, 2006.

(Japanese Standards Association, 2004) Japanese Standards Association, *JIS X8341-3:2004 Guidelines***,** Japan Industrial Standards Committee, 2004.

(Jarzabek und Pettersson, 2006) S. Jarzabek und U. Pettersson, *Tutorial "Cost-effective engineering of web applications pragmatic reuse: building web application product lines"***,** Shanghai, China,ACM Press, 2006.

(Jeff, 1987) C. Jeff, *Hypertext: an introduction and survey***,** Computer %@ 0018-9162,20 (9): 17-41, International Standard Serial Number C. Jeff, 1987.

(Juling, 2005) W. Juling, *KIM Project Homepage*, http://www.kim.uni-karlsruhe.de/, 2005.

(Kagermann und Österle, 2006) H. Kagermann und H. Österle, *Geschäftsmodelle 2010. Wie CEOs Unternehmen transformieren*, Frankfurt,Frankfurter Allgemeine Buch, International Standard Book Number: 3899811143, 2006.

(Kaib, 2002) M. Kaib, *Enterprise Application Integration. Grundlagen, Integrationsprodukte, Anwendungsbeispiele*, Deutscher Universitäts-Verlag, International Standard Book Number: 3-8244-2163-1, 2002.

(Kerer, 2003) C. Kerer, *XGuide - Concurrent Web Development with Contracts***,** Wien, Osterreich,Technische Universität Wien, 1-229, 2003.

(Kerer und Kirda, 2004) C. Kerer und E. Kirda, *XGuide – Concurrent Web Engineering with Contracts*, In Proceedings of 4th Int. Conference on Web Engineering (ICWE 2004), Munich, Germany, Springer, 2004.

(Kiczales et al., 1997) C. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier und J. Irwin, *Aspect-Oriented Programming*, In Proceedings of ECCOP'97 - Object-Oriented Programming, 11th European Conference, Jyväskylä, Springer, 1997.

(Kinikin, Bartels und Harrington, 2004) E. Kinikin, A. Bartels und J. Harrington, *Packaged Apps Lag Business Requirements*, Forrester Research, Inc., n.a., 2004.

(Klaus und Scherwitz-Gallegos, 2004) J. Klaus und A. Scherwitz-Gallegos, *E-Learning - Chancen und Barrieren für Sehgeschädigte***,** Nukath - Die Notebook-Universität Karlsruhe (TH). P. Deussen, W. Juling and B. Thum, Karlsruhe University Press, Seiten: 64-75, International Standard Book Number: 3-937300-01-5, 2004.

(Klyne und Carroll, 2004) G. Klyne und J. J. Carroll, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2004.

(Knapp, Koch, Zhang und Hassler, 2004) A. Knapp, N. Koch, G. Zhang und H.-M. Hassler, *Modeling Business Processes in Web Applications with ArgoUWE***,** 7th International Conference on the Unified Modeling Language (UML2004). Heidelberg, Springer, LNCS 3273, International Standard Book Number: 978-3-540-23307-7, 2004.

(Koch, Kraus, Cachero und Melià, 2003) N. Koch, A. Kraus, C. Cachero und S. Melià, *Modeling Web Business Processes with OO-H and UWE*, In Proceedings of Third Int. Workshop on Web-oriented Software Technology (IWWOST 2003), Oviedo, Asurias, 2003.

(Koch, Kraus und Hennicker, 2001) N. Koch, A. Kraus und R. Hennicker, *The Authoring Process of the UML-based Web Engineering Approach*, In Proceedings of First International Workshop on Web-Oriented Software Technology IWWOST'2001, Valencia, Spain, 2001.

(Kossmann, Leymann und Taubner, 2004) D. Kossmann, F. Leymann und D. Taubner, *Web Services***,** Informatik Spektrum,27 (2): 117-128, International Standard Serial Number Number: 0170-6012, D. Kossmann, F. Leymann und D. Taubner, 2004.

(Krasner und Pope, 1988) G. E. Krasner und S. T. Pope, *A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80***,** Journal of Object-

Oriented Programming,1 (3): 26-49, International Standard Serial Number Number: 0896-8438, G. E. Krasner und S. T. Pope, 1988.

(Kreger, 2003) H. Kreger, *Fulfilling the Web services promise***,** Communications of the ACM,46 (6): 29-34, International Standard Serial Number Number: 0001-0782, H. Kreger, 2003.

(Kropp, Leue und Thompson, 2004) A. Kropp, C. Leue und R. Thompson, *Web Services for Remote Portlets Specification*, http://www.oasis-open.org/committees/wsrp/, 2004.

(Kunz, 2004) A. Kunz, *Semantic-Web-gestützte Realisierung verteilter Informationsräume in Service-orientierten Architekturen***,** Karlsruhe,Universität Karlsruhe, 2004.

(Lawver, 2006) K. Lawver, *Architecting CSS for Syndicated Content*, W3C Track at WWW2006, http://presentations.lawver.net/standards/architecting_css_for_syndicate, 2006.

(Leach, Mealling und Salz, 2005) P. Leach, M. Mealling und R. Salz, *RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace*, http://www.ietf.org/rfc/rfc4122.txt, 2005.

(Leymann und Roller, 2002) F. Leymann und D. Roller, *Using flows in information integration***,** IBM Systems Journal,41 (4): 732-742, International Standard Serial Number F. Leymann und D. Roller, 2002.

(Lima und Schwabe, 2003) F. Lima und D. Schwabe, *Application modeling for the semantic Web*, In Proceedings of First Latin American Web Congress, Sanitago, Chile, 2003.

(Livesey und Guinane, 1996) D. Livesey und T. Guinane, *Developing Object-Oriented Software: An Experience Based Approach*, Prentice Hall (PTR), International Standard Book Number: 0137372485, 1996.

(Lobin, 2003) H. Lobin, *Komplexität und Einfachheit in der Evolution von Dokumentgrammatiken***,** Zeitschrift für Literaturwissenschaft und Linguistik,(131): 106-122, International Standard Serial Number Number: 0049-8653, H. Lobin, 2003.

(Lopes, 1997) C. I. V. Lopes, *D: A LANGUAGE FRAMEWORK FOR DISTRIBUTED PROGRAMMING***,** Northeastern University, 274, 1997.

(Lowe, 1999) D. Lowe, *Hypermedia Patterns Repository (HPR)*, Version 1.0, http://www.designpattern.lu.unisi.ch, 1999.

(Luque Centeno, Delegade Kloos, Gaedke und Nussbaumer, 2005a) V. Luque Centeno, C. Delegade Kloos, M. Gaedke und M. Nussbaumer, *WCAG Formalization with W3C Standards*, In Proceedings of Fourteenth International World Wide Web Conference (WWW), Chiba, Japan, ACM, 2005a.

(Luque Centeno, Delegade Kloos, Gaedke und Nussbaumer, 2005b) V. Luque Centeno, C. Delegade Kloos, M. Gaedke und M. Nussbaumer, *Web Composition with WCAG in Mind*, In Proceedings of Fourteenth International World Wide Web Conference (WWW), International Cross-Disciplinary Workshop on Web Accessibility (W4A), Chiba, Japan, ACM, 2005b.

(Majer, Meinecke und Freudenstein, 2007) F. Majer, J. Meinecke und P. Freudenstein, *Die Landkarte – Rahmenwerk zur Unterstützung von Evolution und Betrieb service-*

orientierter Architekturen, In Proceedings of Workshop Integriertes Informationsmanagement an Hochschulen, Karlsruhe, Germany, Universitätsverlag Karlsruhe, 2007.

(Mayerl, Tröscher und Abeck, 2006) C. Mayerl, F. Tröscher und S. Abeck, *Process-Oriented Integration of Applications for a Service-Oriented IT Management*, In Proceedings of 1st IEEE / IFIP International Workshop on Business-Driven IT Management (BDIM 2006), Vancouver, Canada, IEEE Computer Society, 2006.

(Mayerl, Vogel und Abeck, 2005) C. Mayerl, T. Vogel und S. Abeck, *SOA-based Integration of Service Management Applications*, In Proceedings of IEEE International Conference on Web Services (ICWS 2005), Orlando, USA, IEEE Computer Society, 2005.

(McDonald und Welland, 2004) A. McDonald und R. Welland, *Evaluation of Commercial Web Engineering Processes*, In Proceedings of 4th Int. Conference on Web Engineering (ICWE 2004), Munich, Germany, Springer, 2004.

(McGuinness und Harmelen, 2004) D. L. McGuinness und F. v. Harmelen, *OWL Web Ontology Language Overview*, http://www.w3.org/TR/owl-features/, 2004.

(McIlroy, 1968) M. D. McIlroy, *Mass Produced Software Components*, In Proceedings of Software Engineering. Concepts and Techniques, Brüssel, Belgien, 1968.

(Meinecke, Gaedke und Nussbaumer, 2005) J. Meinecke, M. Gaedke und M. Nussbaumer, *A Web Engineering Approach to Model the Architecture of Inter-Organizational Applications*, In Proceedings of Conference on Component-Oriented Enterprise Applications (COEA 2005), Erfurt, Germany, Gesellschaft für Informatik, 2005.

(Meinecke, Nussbaumer und Gaedke, 2005) J. Meinecke, M. Nussbaumer und M. Gaedke, *Building Blocks for Identity Federations*, In Proceedings of Fifth International Conference for Web Engineering (ICWE2005), Sydney, Australia, Springer, 2005.

(Milner, 1978) R. Milner, *A Theory of Type Polymorphism in Programming.***,** Journal on Computer and System Sciences,17 (3): 348-375, International Standard Serial Number Number: 1064-2307, R. Milner, 1978.

(Mitro, 2002) N. Mitro, *SOAP Version 1.2 Part 0: Primer - W3C Proposed Recommendation 07 May 2003*, http://www.w3.org/TR/soap12-part0/, 2002.

(Mougayar, 2005) W. Mougayar, *The SOA in IT Benchmark Report - What CIOs Should Know about How SOA Is Changing IT*, Aberdeen Group, Boston, 2005.

(Nadalin, Kaler, Hallam-Baker und Monzillo, 2003) A. Nadalin, C. Kaler, P. Hallam-Baker und R. Monzillo, *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)*, http://www.oasis-open.org/committees/documents.php, 2003.

(Nanard und Nanard, 1995) J. Nanard und M. Nanard, *Hypertext design environments and the hypertext design process***,** Communications of the ACM,38 (8): 49-56, International Standard Serial Number Number: 0001-0782, J. Nanard und M. Nanard, 1995.

(Nelson, 1965) T. H. Nelson, *Complex information processing: a file structure for the complex, the changing and the indeterminate*, In Proceedings of 20th

national conference, Cleveland, Ohio, United States, ACM Press, 1965.

(Newcomer und Lomow, 2004) E. Newcomer und G. Lomow, *Understanding SOA with Web Services (Independent Technology Guides)*, Maryland,Addison-Wesley Professional, International Standard Book Number: 0321180860, 2004.

(Nielsen, 1993) J. Nielsen, *Usability Engineering*, San Francisco,Morgan Kaufmann, International Standard Book 1993.

(Norton, 2006) D. Norton, *View DSLs and UML as 'Fraternal Twins,' Not Competitors*, Stamford, CT, 2006.

(Nottingham und Sayre, 2005) M. Nottingham und R. Sayre, *RFC 4287: The Atom Syndication Format*, IETF, Network Working Group, 4287, 2005.

(Nussbaumer, Freudenstein und Gaedke, 2006a) M. Nussbaumer, P. Freudenstein und M. Gaedke, *The Impact of DSLs for Assembling Web Applications***,** Engineering Letters,13 (2006): 387-396, International Standard Serial Number Number: 1816-093X, M. Nussbaumer, P. Freudenstein und M. Gaedke, 2006a.

(Nussbaumer, Freudenstein und Gaedke, 2006b) M. Nussbaumer, P. Freudenstein und M. Gaedke, *Stakeholder Collaboration - From Conversation To Contribution*, In Proceedings of 6. International Conference on Web Engineering (ICWE), SLAC, Menlo Park, California, ACM, 2006b.

(Nussbaumer, Freudenstein und Gaedke, 2006c) M. Nussbaumer, P. Freudenstein und M. Gaedke, *Towards DSL-based Web Engineering*, In Proceedings of 15. International World Wide Web Conference (WWW), Edinburgh, UK, ACM, 2006c.

(Nussbaumer und Gaedke, 2006) M. Nussbaumer und M. Gaedke, *Web Engineering - Technologies for Web Applications***,** Web Engineering: The Discipline of Systematic Development. Gerti Kappel , Birgit Pröll, Siegried Reich and W. Retschitzegger, Wiley, Seiten: 111-132, International Standard Book Number: 0-470-01554-3, 2006.

(O'Reilly, 2005) T. O'Reilly, *What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software*, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2005.

(OASIS, 2004) OASIS, *UDDI 3.0.2 Spec Technical Committee Draft*, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, 2004.

(Ofer und Kneisl, 2006) S. Ofer und S. Kneisl, *Semantische Navigation in verteilten Informationsräumen***,** Karlsruhe,Universität Karlsruhe, 2006.

(Orchard, 2003) D. Orchard, *The four Major Constraints to Loosely Coupled Web Services*, http:///www.webservices.org/index.php/article/articleprint/1246/-1/24/, 2003.

(Ostertag, Hendler, Prieto-Díaz und Braun, 1992) E. Ostertag, J. Hendler, R. Prieto-Díaz und C. Braun, *Computing Similarity in a Reuse Library System: An AI-Based Approach***,** ACM Transactions on Software Engineering and Methodology,1 (3): 205, International Standard Serial Number E. Ostertag, J. Hendler, R. Prieto-Díaz und C. Braun, 1992.

(Pemberton et al., 2000) S. Pemberton, M. Altheim, D. Austin, F. Boumphrey, J. Burger, A. W. Donoho, S. Dooley, K. Hofrichter, P. Hoschka, M. Ishikawa, W. ten Kate, P. King, P. Klante, S. i. Matsui, S. McCarron, A. Navarro, Z. Nies, D. Raggett, P. Schmitz, S. Schnitzenbaumer, P. Stark, C. Wilson, T. Wugofski und D. Zigmond, *XHTML 1.0: The Extensible HyperText Markup Language - A Reformulation of HTML 4 in XML 1.0*, http://www.w3.org/TR/2000/REC-xhtml1-20000126, 2000.

(Peterson, Biron, Malhotra und Sperberg-McQueen, 2004) D. Peterson, P. V. Biron, A. Malhotra und C. M. Sperberg-McQueen, *XML Schema 1.1 Part 2: Datatypes*, W3C Working Draft 17 February 2006, http://www.w3.org/TR/xmlschema11-2/, 2004.

(Phifer, 2006a) G. Phifer, *The Fifth Generation of Portals Supports SOA and Process Integration*, Stamford, CT, 2006a.

(Phifer, 2006b) G. Phifer, *A Portal May Be Your First Step to Leverage SOA*, Stamford, CT, 2006b.

(Pine II und Gilmore, 1999) J. B. Pine II und J. H. Gilmore, *The Experience Economy. Work Is Theater an Every Business a Stage*, Boston,Harvard Business School Press, International Standard Book Number: 978-0875848198, 1999.

(Powell, Jones und Cutts, 1998) T. A. Powell, D. L. Jones und D. C. Cutts, *Web site engineering: beyond Web page design*, Upper Saddle River, NJ,Prentice Hall PTR, International Standard Book Number: 0136509207, 1998.

(Pressman, 2000) R. S. Pressman, *Applying Web Engineering. In: Software Engineering: A Practioner's Approach.*, McGraw-Hill, International Standard Book Number: 0071238409, 2000.

(Prieto-Díaz, 1991) R. Prieto-Díaz, *Implementing Faceted Classification for Software Reuse***,** Communications of the ACM,34 (5): 88, International Standard Serial Number R. Prieto-Díaz, 1991.

(Prommer, 2006) T. Prommer, *Development of an interoperable, reusable and agile web service with Apache Axis***,** Karlsruhe,Universität Karlsruhe, 2006.

(Quantz und Wichmann, 2003) J. Quantz und T. Wichmann, *Integration mit Web Services -- Konzept, Fallstudien und Bewertung*, http://www.berlecon.de/webservices, 2003.

(Rajapakse und Jarzabek, 2005) D. C. Rajapakse und S. Jarzabek, *An Investigation of Cloning in Web Applications*, In Proceedings of 5th International Conference of Web Engineering (ICWE 2005), Sydney, Australia, Springer, 2005.

(Rasmussen, 2005) L. Rasmussen, *Keynote: Google Maps and Browser Support for Rich Web Applications.*, In Proceedings of 5th International Conference of Web Engineering (ICWE 2005), Sydney, Australia, Springer, 2005.

(Razumna, 2003) Y. Razumna, *Einsatz LOM-basierter Metadaten in eLearning Plattformen***,** Karlsruhe,Universität Karlsruhe, 2003.

(Remus, 2006) U. Remus, *Critical Success Factors of Implementing Enterprise Portals*, In Proceedings of 39th Annual Hawaii International Conference on System Sciences (HICSS' 39), Island of Maui, USA, IEEE, 2006.

(Richter, Haller und Schrey, 2005) J.-P. Richter, H. Haller und P. Schrey, *Serviceorientierte Architekturen***,** 413-416, 2005.

(Riehm und Vogler, 1996) R. Riehm und P. Vogler, *Middleware - Begriff und Einordnung***,** Middleware - Grundlagen, Produkte und Anwendungsbeispiele für die Integration heterogener Welten. H. Österle, R. Riehm and P. Vogler, Wiesbaden, Vieweg, International Standard Book 1996.

(Rojas et al., 2002) I. Rojas, L. Bernardi, E. Ratsch, R. Kania, U. Wittig und J. Saric, *A database system for the analysis of biochemical pathways.***,** In Silico Biology,2 7, International Standard Serial Number I. Rojas, L. Bernardi, E. Ratsch, R. Kania, U. Wittig und J. Saric, 2002.

(Rosson, Ballin, Rode und Toward, 2005) M. B. Rosson, J. F. Ballin, J. Rode und B. Toward, *"Designing for the Web" Revisited: A Survey of Informal and Experienced Web Developers*, In Proceedings of 5th International Conference of Web Engineering (ICWE 2005), Sydney, Australia, Springer, 2005.

(Rudin, 2006) T. Rudin, *Modellierung dynamischer Benutzerschnittstellen im Web Engineering***,** Karlsruhe,Universität Karlsruhe, 2006.

(Sametinger, 1997) J. Sametinger, *Software Engineering with Reusable Components*, Berlin,Springer, International Standard Book Number: 3-540-62695-6, 1997.

(Schmidt, Stal und Rohnert, 2000) D. C. Schmidt, M. Stal und H. Rohnert, *Pattern-Oriented Software Architecture*, Wiley & Sons, International Standard Book Number: 0471606952, 2000.

(Schwabe, Pontes und Moura, 1999) D. Schwabe, R. d. A. Pontes und I. Moura, *OOHDM-Web: an environment for implementation of hypermedia applications in the WWW***,** ACM SIGWEB Newsletter,8 (2): 18-34, International Standard Serial Number Number: 1931-1745, D. Schwabe, R. d. A. Pontes und I. Moura, 1999.

(Schwabe und Rossi, 1998) D. Schwabe und G. Rossi, *An Object Oriented Approach to Web-Based Applications Design***,** TAPOS - Theory and Practice of Object Systems,4 (4): 207-225, International Standard Serial Number D. Schwabe und G. Rossi, 1998.

(Schwabe, Rossi und Barbosa, 1996) D. Schwabe, G. Rossi und S. Barbosa, *Systematic Hypermedia Design with OOHDM*, In Proceedings of ACM International Conference on Hypertext' 96, Washington, USA, 1996.

(Schwabe, Szundy, Moura und Lima, 2004) D. Schwabe, G. Szundy, S. S. d. Moura und F. Lima, *Design and Implementation of Semantic Web Applications*, In Proceedings of WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, 2004.

(Schwall, 2005) C. Schwall, *Rechtemanagement in Verteilten Systemen mit Web-Services***,** Karlsruhe,Universität Karlsruhe, 2005.

(Schwinger und Koch, 2003) W. Schwinger und N. Koch, *Modellierung von Web Anwendungen***,** Web Engineering - Web Engineering - Systematische Entwicklung von Web Anwendungen. Gerti Kappel , Birgit Pröll, Siegried Reich and W. Retschitzegger, Heidelberg, dPunkt Verlag, Seiten: 49-75, International Standard Book Number: 3-89864-234-8, 2003.

(Selmi, Kraiem und Ghézala, 2006) S. S. Selmi, N. Kraiem und H. H. B. Ghézala, *A Strategic-Oriented Method for Web Applications Design: A Case Study.*, In Proceedings of AICT/ICIW, 2006.

(Semia Sonia Selmi, Naoufel Kraiem und Ghezala, 2005) Semia Sonia Selmi, Naoufel Kraiem und H. B. Ghezala, *Toward a Comprehension View of Web Engineering*, In Proceedings of 5th International Conference of Web Engineering (ICWE 2005), Sydney, Australia, Springer, 2005.

(Sivashanmugam , Verma, Sheth und Miller, 2003) K. Sivashanmugam , K. Verma, A. P. Sheth und J. Miller, *Adding Semantics to Web Services Standards.*, In Proceedings of International Conference on Web Services (ICWS), CSREA Press, 2003.

(Srinivasan und Treadwell, 2005) L. Srinivasan und J. Treadwell, *An Overview of Service-oriented Architecture, Web Services and Grid Computing*, HP Software Global Business Unit, 2005.

(Suda, 2006) B. Suda, *Using Microformats* O'Reilly, International Standard Book Number: 0-596-52821-3, 2006.

(Sure et al., 2002) Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer und D. Wenke, *OntoEdit: Collaborative Ontology Development for the Semantic Web***,** Sardinia, Italy,Springer-Verlag, 2002.

(Szyperski, 1997) C. Szyperski, *Component software: beyond object-oriented programming*, Reading, Mass.,ACM Press; Addison-Wesley, International Standard Book Number: 0201178885, 1997.

(Szyperski, Gruntz und Murer, 2002) C. Szyperski, D. Gruntz und S. Murer, *Component software: beyond object-oriented programming*, Addison Wesley; 2nd Ed edition, International Standard Book Number: 978-0201745726, 2002.

(Tanenbaum, 2002) A. S. Tanenbaum, *Computer networks*, Upper Saddle River, N.J.,Prentice Hall PTR, International Standard Book Number: 0130661023, 2002.

(Tarr, Ossher, Harrison und Sutton, 1999) P. Tarr, H. Ossher, W. Harrison und S. M. Sutton, *N degrees of separation: multi-dimensional separation of concerns*, In Proceedings of 21st international conference on Software engineering, Los Angeles, California, United States, IEEE Computer Society Press, 1999.

(The Standish Group International, 1994-2005) The Standish Group International, *CHAOS Research*, http://www.standishgroup.com, 1994-2005.

(Thompson, Beech, Maloney und Mendelsohn, 2001) H. S. Thompson, D. Beech, M. Maloney und N. Mendelsohn, *XML Schema Part 1: Structures*, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/, 2001.

(Tolvanen, 2006) J.-P. Tolvanen, *Domänenspezifische Modellierung für vollständige Code-Generierung***,** JavaSpektrum,(1/2006): 9-12, International Standard Serial Number Number: 1431-4436, J.-P. Tolvanen, 2006.

(Trefz, 2004) C. Trefz, *Blogging unter dem Aspekt der Integration***,** Karlsruhe,Universität Karlsruhe, 2004.

(Turowski, 2002) K. Turowski, *Vereinheitlichte Spezifikation von Fachkomponenten*, Gesellschaft für Informatik (GI), Arbeitskreis 5.10.3, 2002.

(Veryard, 2004) R. Veryard, *Business Adaptability and Adaptation in SOA***,** CBDi Journal,2004 (Februar): 15-23, International Standard Serial Number Number: 1745-1884, R. Veryard, 2004.

(W3C-SOA, 2004) W3C-SOA, *Web Services Architecture*, http://www.w3.org/TR/ws-arch/wsa.pdf, 2004.

(W.Boehm et al., 2000) B. W.Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani und C. Abts, *Software Cost Estimation with Cocomo II*, Englewood Cliffs,Prentice Hall, International Standard Book Number: 0130266922, 2000.

(Wada und Suzuki, 2005) H. Wada und J. Suzuki, *Modeling Turnpike Frontend System: A Model-Driven Development Framework Leveraging UML Metamodeling and Attribute-Oriented Programming*, In Proceedings of 8th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica, 2005.

(Wales, 2004-2006) J. Wales, *Wikipedia*, http://www.wikipedia.org/, 2004-2006.

(Wales, 1999) M. G. Wales, *WIDL: interface definition for the Web***,** IEEE Internet Computing,3 (1): 55-59, International Standard Serial Number Number: 1089-7801, M. G. Wales, 1999.

(Warnke, 2006) K. Warnke, *BIK - Barrierefrei Informieren und Kommunizieren*, http://www.bik-online.info/test/bitv/bilanz_2.php, 2006.

(Warren, 2004) S. Warren, *Ein mundgerechtes Stück ASP.NET ViewState*, http://www.microsoft.com/germany/msdn/library/net/aspnet/EinMundgerechtesStueckASPNETViewState.mspx, 2004.

(Winer, 1999) D. Winer, *XML-RPC Specification*, http://www.xmlrpc.com/spec, 1999.

(Zarnekow, Brenner und Pilgram, 2005) R. Zarnekow, W. Brenner und U. Pilgram, *Integriertes Informationsmanagement*, Heidelberg,Springer Verlag, International Standard Book Number: 3-540-23303-2, 2005.

(Zimmermann, Tomlinson und Peuser, 2005) O. Zimmermann, M. Tomlinson und S. Peuser, *Perspectives on Web services*, Berlin,Springer, International Standard Book Number: 3-540-00914-0, 2005.