Lehrstuhl für Informatik 19 (sebis) Software Engineering betrieblicher Informationssysteme

Anwendung der „Metro-Map“ Metapher auf die Gestaltung von Unternehmensportalen

Stephan Ziemer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: .....
Prüfer der Dissertation:
1. Professor Dr. Florian Matthes
2. Professor Dr. Helmut Krcmar

Die Dissertation wurde am .....bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am ..... angenommen.

Zusammenfassung

Unternehmensportale als nächste Stufe der betrieblichen Informationssysteme versprechen eine integrierte Sicht auf die Daten eines Unternehmens und einen zentralen Zugang zu Funktionen. Die Entwicklung eines Portals beginnt häufig mit einem Kommunikationsproblem, da die beteiligten Gruppen das zu erstellende Portal unter ihren jeweils eigenen Gesichtspunkten betrachten. Entsprechend betonen bisherige Modelle und Entwicklungsmethoden für Portale und Dienste den ein oder anderen Aspekt. Ein einheitliches Modell eines Portals mit den angebotenen benutzerorientierten Diensten und den notwendigen Anpassungen, das für alle, also auch für Entwickler, Designer, Entscheider und Endanwender, zugleich nützlich ist, kann nicht nur die gemeinsame Kommunikation und damit das gemeinsame Verständnis fördern, sondern bei entsprechender Mächtigkeit auch erheblichen Aufwand bei der Implementierung im Vergleich zu bisherigen Lösungen einsparen.

Das Modell der Enterprise Portal Maps verwendet die gebräuchliche grafische Notation der Netzpläne im öffentlichen Personennahverkehr, der Metro-Maps, und Begriffe wie „Station“, „Dienst“, „Bedingung“ und „Linie“, um ein Portal mit seinen benutzerorientierten Diensten zu beschreiben. Dabei trifft das Modell nur einige wenige Annahmen über das allgemeine Verhalten bzw. die Eigenschaften von Diensten und Bedingungen, ohne sich näher mit der Implementierung zu beschäftigen. Das Modell erlaubt die Visualisierung der Struktur eines Portals in Form eines grafischen Netzplan für Designer und Endanwender, in der Seiten als Stationen und Navigationspfade als Linien dargestellt werden.

Die Anwendbarkeit und der Nutzen des Modells wird anhand von zwei sehr unterschiedlichen Portalen und Anwendungsgebieten demonstriert. Der Designer eines Portals, also derjenige, der festlegt, wem was unter welchen Bedingungen im Portal zur Verfügung gestellt werden soll, definiert auf Basis vorhandener Dienste und Bedingungen eine Enterprise Portal Map mit Hilfe eines grafischen Editors. Die Enterprise Portal Map wird dann je nach Ziel-System von einer Laufzeitkomponente interpretiert, die Laufzeitkomponente erzeugt dynamisch entsprechende Portalseiten angepasst auf den konkreten Benutzer. Es wurden zwei Laufzeitkomponenten erstellt, eine für den *infoAsset Broker*, eine für den *SAP Web Application Server*. Als Anwendungsbeispiele wurden für die Wissensmanagement-Dienste des infoAsset Brokers und die Flugbuchung-Dienste des SAP Web Application Servers kleine Hüllen geschrieben, die die Annahmen des EPM-Modells erfüllen. Dadurch konnten zwei Enterprise Portal Maps für komplexe Portale definiert werden, eines für das Wissensmanagement in Unternehmen und eines für die Online-Buchung von Flügen. Den Endanwendern wird über ein animiertes Applet ihr derzeitiger „Aufenthaltsort“ und die weiteren Navigationsmöglichkeiten auf der Enterprise Portal Map angezeigt.

Abstract

Enterprise Portals promise an integrated view on functions and data in a company. In the process of portal design many different groups are involved, each taking their own point of view. There are two major problems to be solved in this context: Firstly there is a lack of a common model of the portal which servers all groups at the same time and which can be the foundation of a common language between the groups. Secondly the implementation of portals is a time- and cost-consuming matter for many steps must be performed manually.

The suggested model, called "Enterprise Portal Maps Model", uses the metaphor of the public transport to solve the two problems stated above. Portal concepts like *pages*, *functions* and *links* are expressed by means of *stations*, *services* and *lines*. This yields a view of a portal which is common to most people in the sense of public transport and which is powerful enough not only to integrate "legacy"-functionality but also to be implemented in a way that allows automatic generation of the modelled portal even for very different portal systems.

Inhaltsverzeichnis

1 Einleitung **9**
- 1.1 Motivation . . . 9
- 1.2 Ziele der Arbeit . . . 10
- 1.3 Vorgehen und Gliederung . . . 10

2 Portale und Portalsysteme **13**
- 2.1 Die Bedeutung Betrieblicher Informationssysteme . . . 13
- 2.2 Portale und Portalsysteme . . . 15
  - 2.2.1 Der Begriff des Portals . . . 15
  - 2.2.2 Anforderungen bei der Gestaltung von Portalen . . . 19
  - 2.2.3 Entwicklungsmethoden für Portale und Dienste . . . 21
  - 2.2.4 Die Modellierungssprache WebML . . . 23
  - 2.2.5 mySAP Portals und Web Application Server . . . 33
  - 2.2.6 infoAsset Broker . . . 45
- 2.3 Zusammenfassung . . . 55

3 Bausteine eines Portals **57**
- 3.1 Dienste in Unternehmensportalen . . . 57
- 3.2 Personalisierung in Unternehmensportalen . . . 58
  - 3.2.1 Ziele der Personalisierung . . . 58
  - 3.2.2 Der Prozess der Personalisierung . . . 59
  - 3.2.3 Profiling . . . 60
  - 3.2.4 Objekte und Methoden der Anpassung . . . 62
  - 3.2.5 Dienste für die Personalisierung . . . 63
- 3.3 Eine Referenzarchitektur für Portale und Dienste . . . 64
- 3.4 Zusammenfassung . . . 67

4 Das Modell der Enterprise Portal Maps **69**
- 4.1 Portale und Dienste als Streckennetze und Stationen . . . 69
  - 4.1.1 Karten als graphischer Kommunikationsprozess . . . 70
  - 4.1.2 Londons Netzplan: Ein historischer Abriss . . . 71
  - 4.1.3 Die Idee der Enterprise Portal Maps . . . 74
- 4.2 Kontexte . . . 75

4.3 Bedingungen . . . 75
4.4 Dienste . . . 76
4.4.1 Interaktionselemente . . . 77
4.4.2 Anforderung . . . 80
4.4.3 Ausführung und Ergebnis . . . 81
4.4.4 Beendigung . . . 81
4.4.5 Diensttypen . . . 82
4.4.6 Typisierung von Diensten . . . 82
4.5 Stationen . . . 83
4.5.1 Dienstplan . . . 83
4.5.2 Fahrplan . . . 84
4.5.3 Ableitung der Benutzersicht aus der Designersicht . . . 85
4.6 Linien . . . 85
4.7 Netzpläne . . . 86
4.8 EPM als Sprache . . . 89
4.9 Auswirkungen auf Komponenten der Referenzarchitektur . . . 90
4.10 Zusammenfassung . . . 90

5 Implementierungen des EPM-Modells 93
5.1 Ein Definitionswerkzeug für Enterprise Portal Maps . . . 93
5.1.1 Das EPM-Modell . . . 94
5.1.2 Der graphische Editor . . . 97
5.1.3 Die Schnittstelle zur EPM-Laufzeitumgebung . . . 99
5.2 EPM für den infoAsset Broker . . . 100
5.2.1 Eine EPM-Laufzeitkomponente für den infoAsset Broker . . . 101
5.2.2 Die Schnittstelle zum EPM-Editor . . . 104
5.2.3 Anpassung der Benutzungsoberfläche . . . 110
5.2.4 EPM für ein Wissensmanagementportal . . . 111
5.3 EPM für den SAP Web Application Server . . . 118
5.3.1 EPM-Laufzeitumgebung für den SAP Web AS . . . 118
5.3.2 Die Schnittstelle zum EPM-Editor . . . 125
5.3.3 Kapselung der SAP-Funktionen . . . 126
5.3.4 Die Benutzungsoberfläche . . . 129
5.3.5 EPM für Flugbuchungen . . . 130
5.4 Zusammenfassung . . . 134

6 Bewertung und Ausblick 137
6.1 Bewertung des Ansatzes . . . 137
6.1.1 Erfüllung der Anforderungen . . . 137
6.1.2 Nutzen und Grenzen der Metapher . . . 140
6.2 Erfahrungen mit der Notation der Netzpläne . . . 141
6.3 Ausblick . . . 142

Literaturverzeichnis **145**

Abbildungsverzeichnis

2.1 Grafische Notation für Dateneinheiten und eine mögliche Umsetzung in HTML. 26
2.2 Grafische Notation von Indexeinheiten und Umsetzung in HTML. 26
2.3 Grafische Notation von Scroll-Einheiten und Umsetzung in HTML. 27
2.4 Grafische Notation von Filtereinheiten und Umsetzung in HTML. 28
2.5 Grafische Darstellung einer direkten Einheit. 28
2.6 Anlage eines neuen Musikers in WebML. 29
2.7 Grafische Darstellung von Seiten in WebML und eine Umsetzung in HTML. 30
2.8 Navigation mit Hilfe kontextabhängiger Verweise in WebML und HTML. . 31
2.9 Darstellung des Beispiel-Portals in WebML. 32
2.10 Überblick über die Architektur der mySAP Technologie. 35
2.11 Die SAP-Portals-Architektur ([SAPP02, Seite 8]) 36
2.12 Grundsätzlicher Aufbau einer SAP-Portals-Seite ([Zah02, Seite 19]). 37
2.13 Architektur des Web Application Servers ([SAPW02, Seite 18]). 39
2.14 Bestandteile einer BSP-Applikation ([SAPH02]). 41
2.15 Definition eines Seitenattributs. 42
2.16 Navigationsstruktur einer BSP-Applikation. 44
2.17 Architektur des infoAsset Brokers ([Weg02, Seite 142]). 47

3.1 Klassifizierung der Profiling-Daten.(Abbildung aus [Jac02, Seite 11]). . . . 61
3.2 Datengewinnungsmethoden der Personalisierung (Abbildung aus [Jac02, Seite 11]). 63
3.3 Eine Referenzarchitektur für Portale und Dienste. 66
3.4 Konkrete Portale als Zusammensetzung von Bausteinen mit dem Controller als verbindendes Element. 68

4.1 Frühe Darstellung der elektrischen Bahnen in London. 72
4.2 Das erste Diagramm von H. C. Beck. 74
4.3 Konzeptuelles Klassendiagramm der Interaktionselemente. 78
4.4 Interaktionselement eines Verzeichnisdienstes. 80
4.5 Der Netzplan als zentrale Klasse des EPM-Modells. 87
4.6 Netzplan mit 42 Stationen und 107 Einträgen in den Fahrplänen. 88
4.7 Die Enterprise Portal Map als zentrale Stelle für die Definition eines Portals. 91

5.1 Bildschirmausschnitt mit Menüleiste, Symbolleiste und Kartenfläche. . . . 97
5.2 Kontextmenü eines Streckenabschnitts. 98
5.3 Dialoge zur Änderung der Stationseigenschaften und eines Interaktionselementes. 99
5.4 Die Komponenten der EPM-Laufzeitumgebung für den infoAsset Broker. . 102
5.5 Ablauf einer Anfragebearbeitung im infoAsset Broker 103
5.6 Die Benutzungsoberfläche mit EPM-Applet. 110
5.7 Auszug aus einer Site Map. 112
5.8 Designersicht einer Enterprise Portal Map für ein Wissensmanagementportal. 116
5.9 Startseite des Portals nach der Anmeldung als Administrator. 117
5.10 Die Klassen der EPM-Laufzeitumgebung für den SAP Web Application Server. 120
5.11 Abfolge der Ereignisse während der Abarbeitung einer BSP-Seite. 121
5.12 Die SAP-EPM-Benutzungsoberfläche mit Applet. 130
5.13 Flugbuchungen: Geschäftsobjekte, ihre Schlüsselfelder und Methoden. . . . 132
5.14 Designersicht auf eine Enterprise Portal Map für Flugbuchungen im Internet. 134

Kapitel 1

Einleitung

Unternehmensportale bieten eine integrierte Sicht auf Daten und Funktionen betrieblicher Informationssysteme in Verbindung mit größerer Flexibilität, als diese bei traditionellen betrieblichen Informationssystemen der Fall ist. Bei der Gestaltung von Portalen sind verschiedene Gruppen eingebunden, die ihre jeweils eigene Sicht haben. Gleichzeitig muss jede Gruppe mit den den anderen Gruppen über die Gestaltung des Portals kommunizieren. Nachdem der Inhalt und die Struktur des Portals festgelegt wurde, verursacht die Implementierung des Portals häufig weitere enorme Aufwände.

1.1 Motivation

Bei der Gestaltung von Unternehmensportalen sind viele Probleme zu lösen. Bisherige Ansätze zur Modellierung und Implementierung von Portalen konzentrieren sich meistens auf die technischen Probleme, dazu zählen z.B.:

1. die Modellierung der im Portal zu präsentierenden oder zu manipulierenden Daten
2. die Konsistenz der Daten, falls diese aus verschiedenen Ursprungssystemen stammen
3. die angepasste Auslieferung der darzustellenden (HTML-)Seiten
4. die Performanz des Gesamtsystems (Architektur und Skalierbarkeit)
5. der technische Aufruf von Funktionen in entfernten Systemen.

Die technischen Probleme wurden dabei schon sehr weitgehend gelöst und in die Praxis umgesetzt, wie die Eigenschaften verschiedener kommerzieller Portal-Systeme verdeutlichen. Der bei der konkreten Lösung der Probleme entstehende Aufwand ist dabei nicht zu unterschätzen.

Weit weniger Beachtung gefunden hat bisher der Umstand, dass die durch die Integration mehrerer Applikationen entstandene, sehr komplexe „Meta-Applikation“ (das Portal mit seinen enthaltenen Diensten) für verschiedene Gruppen aus verschiedenen Blickwinkeln

heraus verständlich und anwendbar sein muss. Der traditionelle Ansatz, nämlich die explizite Erstellung von Dokumentation für die verschiedenen Gruppen, versagt bei Portalen noch mehr als bei herkömmlichen Applikationen, da Portale im höchsten Maße personalisierbar und in ihrem Inhalt und ihrer Struktur dynamisch sein sollen.

1.2 Ziele der Arbeit

Im Rahmen dieser Arbeit soll ein Modell entwickelt werden, das vorhandene Dienste und Personalisierungsmöglichkeiten als Bausteine eines Portals interpretiert und die Definition eines Portals basierend auf den Bausteinen erlaubt. Das Modell soll den Inhalt und die Struktur des Portals für verschiedene Gruppen verständlich beschreiben und muss dabei dennoch ausdrucksstark genug sein, damit aus dem Modell heraus das tatsächliche Portal automatisch erzeugt werden kann und somit der Implementierungsaufwand dramatisch sinkt.

Ein solches Modell für Unternehmensportale und der enthaltenen benutzerorientierten Dienste soll aber nicht nur vorgestellt werden, sondern es soll auch die praktische Anwendbarkeit und der Nutzen des Modells für die verschiedenen Gruppen anhand realer, nicht-trivialer Beispiele verdeutlicht werden. Um Aussagen über die Stärke des Modells und nicht über die Schwächen eines speziellen Systems zu erhalten, soll das Modell für zwei sehr unterschiedliche, kommerzielle Portalsysteme umgesetzt werden.

1.3 Vorgehen und Gliederung

Im ersten Teil der Arbeit (Kapitel 2) werden Portale und Portalsysteme als Integrationsmechanismus für betriebliche Informationssysteme vorgestellt. Nach einem kurzen Abriss über die Bedeutung der betrieblichen Informationssysteme werden verschiedene Typen von Portalen betrachtet und die Begriffe Web-Applikation und Portal voneinander abgegrenzt, soweit dies möglich ist. Anhand zentraler Anforderungen bei der Gestaltung von Portalen werden drei unterschiedliche Portalsysteme verglichen und Schwächen aufgezeigt.

In Kapitel 3 wird das Wesen von Unternemhensportalen untersucht. Zunächst werden die allgemeinen Eigenschaften von Diensten in Unternehmensportalen sowie die grundsätzliche Personalisierung von Diensten und Portalen beschrieben, um anschließend eine allgemeine Referenzarchitektur, die die Bausteine eines Portals und ihr Zusammenspiel umfasst, angeben zu können.

Im zweiten Teil der Arbeit (Kapitel 4) wird das Modell der Enterprise Portal Maps, kurz EPM genannt, vorgestellt, das die Bausteine eines Portals metaphorisch interpretiert. Das EPM-Modell verwendet die Stationen, Linien und Netzpläne des ÖPNV als Bilder für Portale und enthaltene Dienste. Nach einem kurzen Abriss über Karten als graphischer Kommunikationsprozess im Allgemeinen, wird das spezielle Design der Netzpläne im ÖPNV vorgestellt und die besonderen Eigenschaften hervorgehoben. Nach einem Überblick über das EPM-Modell im Ganzen werden die einzelnen Bestandteile der Enterprise

Portal Maps im Detail beschrieben. Es wird eine Grammatik angegeben, die die Syntax des Modells wiedergibt. Abschließend werden die Auswirkungen des EPM-Ansatzes auf die Referenzarchitektur betrachtet.

Der dritte Teil der Arbeit (Kapitel 5) befasst sich mit der Umsetzung des im vorhergehenden Teil beschriebenen EPM-Modells. Dabei werden zunächst ein allgemeines Definitionswerkzeug, sowie die notwendigen Schnittstellen zu einer möglichen Laufzeitkomponente vorgestellt. Es folgen zwei Implementierung der EPM-Laufzeitkomponente für zwei kommerzielle Portalsysteme. Für jedes System wird eine konkrete Enterprise Portal Maps vorgestellt, zum Einen ein Portal für das Wissensmanagement in Unternehmen und zum Anderen ein Portal für die Buchung von Flügen.

Die Arbeit schließt mit einer Diskussion darüber, in wie weit das Modell der Enterprise Portal Maps die im zweiten Kapitel aufgezeigten Anforderungen bei der Gestaltung von Portalen erfüllt und gibt Hinweise für zukünftige Aktivitäten und mögliche Forschungsrichtungen.

Danksagung

Mein erster Dank gilt meinem Doktorvater Professor Dr. Florian Matthes, der in zahlreichen Diskussionen wesentliche Impulse und Anregungen für diese Arbeit gegeben hat. Seine Betreuung war dabei nicht nur inhaltlicher Natur, auch organisatorisch gebührt ihm mein Dank. Nur aufgrund der Tatsache, dass ich eineinhalb Jahre lang in Teilzeit in der von ihm gegründeten Firma *infoAsset* arbeiten durfte, war mir das Verfassen einer Dissertation überhaupt möglich. Herrn Professor Dr. Helmut Krcmar danke ich für die intensive Diskussion, seine konstruktiven Anmerkungen und sein Interesse am Thema.

Für die anregenden Diskussionen rund um mein Thema und andere Hilfe, die mir zuteil wurde, danke ich in alphabetischer Reihenfolge: Thomas Büchner, Cornelia Jacobsen, Hans-Werner Sehring, Thomas Sidow, Michael Skusa, Holm Wegner, Axel Wienberg und André Wittenburg.

Mein ganz besonderer Dank gilt meinen Eltern, die mich bei allen meinen Entscheidungen stets unterstützt haben und die mir während der Zeit des Ausarbeitens und des Schreibens dieser Arbeit großzügige finanzielle Hilfe gewährt haben. Ihnen ist diese Arbeit gewidmet.

Meine Frau Purkin hatte sicherlich am meisten unter dieser Arbeit zu leiden. Meine „Arbeitszeiten“ waren unregelmäßig und sehr spontan, und oft war ich geistig mit meiner Dissertation beschäftigt, wenn andere Dinge wichtiger gewesen wären. Ihr widme ich diese Arbeit ganz besonders.

Kapitel 2

Portale und Portalsysteme

Im diesem Teil der Arbeit werden Portale und Portalsysteme als Integrationsmechanismus für betriebliche Informationssysteme vorgestellt. Nach einem kurzen Abriss über die Bedeutung der betrieblichen Informationssysteme werden verschiedene Typen von Portalen betrachtet und die Begriffe Web-Applikation und Portal voneinander abgegrenzt, soweit dies möglich ist. Anhand zentraler Anforderungen bei der Gestaltung von Portalen werden drei unterschiedliche Portalsysteme verglichen und Schwächen aufgezeigt.

2.1 Die Bedeutung Betrieblicher Informationssysteme

Betriebliche Informationssysteme dienen der Unterstützung von *Geschäftsprozessen*. Ein Geschäftsprozess wird in [Jab97, Glossar] wie folgt definiert: „Ein Geschäftsprozess ist ein Vorgang in Wirtschaftseinheiten (Unternehmen, Verwaltungseinheiten etc.), der einen wesentlichen Beitrag zu einem nicht notwendigerweise ökonomischen Unternehmenserfolg leistet. Dabei läuft er in der Regel funktions-, hierarchie- und standortübergreifend ab, kann die Unternehmensgrenzen überschreiten und erzeugt einen messbaren, direkten Kundennutzen."

Etwas allgemeiner definieren Hammer und Champy in [HaCh93]: „[A business process is] a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer".

Ein betriebliches Informationssystem wird durch ein ein betriebswirtschaftliches Fachkonzept (Daten-, Funktions- und Organisationsmodell) auf der Ebene des Geschäftsprozesses beschrieben und als Hard- und/oder Softwarelösung implementiert [Su02, Seite 14]. Suhl ([Su02, a.a.O.]) nennt folgende Gliederungsarten für betriebliche Informationssysteme:

Nach Erstellungsart Individual- oder Standardsoftware.

Nach Funktionen Forschung und Entwicklung, Beschaffung und Materialwirtschaft, Produktion, Vertrieb und Marketing, Finanz- und Rechnungswesen, Personalwesen, Controlling, etc..

Nach Unterstützungsebene Administrationssysteme, Dispositionssysteme, Berichts- und Kontrollsysteme, Analyse und Informationssysteme, Planungs- und Entscheidungssysteme.

Nach Integrationsart innerbetriebliche und zwischenbetriebliche Integration, Datenbankintegration, Funktionsintegration, Prozessintegration, EDI-Integration, etc..

Nach Branche industrielle Anwendungssysteme, Anwendungssysteme in Handel, Banken, Versicherungen, Verkehrs- und Transportwesen, etc..

Nach Kerntechnologien daten-, modellorientiert, Entscheidungsunterstützung, wissensbasiert.

In nahezu jedem Unternehmen werden mehrere betriebliche Informationssysteme verwendet. Hierfür kann es historische Gründe geben, etwa nach der Fusion zweier Unternehmen, fachliche Gründe, z.B. weil kein einzelnes System mit allen gewünschten Funktionen verfügbar ist, praktische Gründe, wie z.B. Erwägungen zur Performanz, oder auch rechtliche Gründe, weil z.B. das System für die Verarbeitung der Personaldaten aus Gründen des Datenschutzes physisch und technisch nur einem bestimmten Personenkreis zugänglich sein darf.

Historisch betrachtet hat sich die Systemarchitektur betrieblicher Informationssysteme von monolithischen Systemen zu mehrschichtigen Systemen gewandelt. Während bei monolithischen Systemen alle Funktionen, also etwa Datenhaltung, Verarbeitungslogik und Präsentation in einem Programm realisiert sind, werden die Funktionen bei mehrschichtigen Systemen auf einzelne Schichten aufgeteilt. Durch die Aufteilung auf mehrere Schichten lassen sich insbesondere die Wiederverwendbarkeit einzelner Schichten für andere Systeme und die Skalierbarkeit des Gesamtsystems verbessern. Derzeit ist die Aufteilung in eine Schicht für Datenhaltung, eine für die Geschäftslogik und eine für die Präsentation üblich. Schichtenarchitekturen haben zusätzlich den Vorteil, dass bei Bedarf weitere Schichten hinzugefügt oder bestehende Schichten modular verändert werden können, um beispielsweise den Zugriff über mobile Endgeräte zu erreichen[1].

Die Benutzungsoberfläche betrieblicher Informationssysteme ist traditionell auf die Bewältigung des konkreten Arbeitsschritts fokussiert. Während die ersten Systeme über keine bzw. fast keine Benutzungsoberfläche verfügten, also nur für reine Verarbeitungsschritte von Experten verwendet wurden, kamen seit den 70er Jahren des vorigen Jahrhunderts zunehmend Dialogsysteme zum Einsatz, die eine unmittelbare Kommunikation mit auch weniger versierten Anwendern erlauben und heutzutage an praktisch allen betrieblichen Arbeitsplätzen in der ein oder anderen Form zu finden sind. Ein bestimmte Dialogfolge stellt dabei einen Arbeitsschritt dar. Da moderne betriebliche Informationssysteme üblicherweise mehr als einen Arbeitsschritt unterstützen, werden die Einstiegspunkte zu den Dialogfolgen in hierarchischen Menüs organisiert, die selbst wieder in Menüs und Untermenüs gegliedert sein können.

[1] Vgl. z.B. [Bar01] oder SAPs *Mobile Engine*.

Aus Sicht des Management ist es aus verschiedenen Gründen heraus wünschenswert, Geschäftsprozesse transparent darzustellen und schnell und unkompliziert ändern zu können. In betrieblichen Informationssystemen ist jedoch eine große Menge speziellen Wissens über konkrete Arbeitsschritte und Geschäftsprozesse enthalten, so dass eine Ablösung oder Umgestaltung veralteter Systeme bzw. Abläufe durch neue Systeme bzw. Abläufe oft sehr schwierig und aufwändig ist.

Ein erster Schritt in Richtung Flexibilität ist die Modularisierung der Systeme bzw. der Funktionen, um die so entstandenen Module, die möglichst unabhängig voneinander sein sollen, zu neuen Systemen zusammensetzen zu können. Böhmann et al beschreiben in [BöJu+03] wie betriebliche Informationssysteme unter Berücksichtigung der Anforderungen von Service Providern für verschiedene Kunden methodisch in Module zerlegt werden können.

Einen weiter gehenden Ansatz verfolgen *Workflow-Management-Systeme*, die versuchen, die Kontrolle, welche Arbeitsschritte von wem zu welcher Zeit unter welchen Bedingungen vorgenommen werden, von den betrieblichen Informationssystemen auf sog. Workflow-Engines zu verlagern und damit die Umsetzung der Geschäftsprozesse dynamischer zu gestalten. Veraltete Systeme bzw. Abläufe sollen so schneller austauschbar oder anpassbar werden. Dem einzelnen Benutzer gegenüber präsentieren sich Workflow-gestützte Geschäftsprozesse meistens als Listen (Körbe) mit unerledigten, in Arbeit befindlichen, wartenden und abgearbeiteten Aufgaben. Eingehende Darstellungen des Workflow-Ansatzes finden sich z.B. bei [Jab97] oder [Zie98].

Für die Integration verschiedener betrieblicher Informationssysteme setzt sich zunehmend ein zweistufiger Ansatz durch: Einerseits die Integration auf der Ebene der Daten durch den Austausch von XML-Dokumenten. Der Austausch von XML-Dokumenten wird in dieser Arbeit nicht weiter betrachtet. Die oberflächliche Integration der Systeme findet zunehmend über *Portale* statt.

2.2 Portale und Portalsysteme

In diesem Abschnitt wird zunächst eine Übersicht über die Verwendung des Begriffs „Portal“ gegeben und aufgrund der Unterschiede der Begriff für diese Arbeit festgelegt. Nach einer Diskussion über die Anforderung bei der Gestaltung von Portalen wird die Erfüllung der Anforderungen bei drei sehr unterschiedlichen Portalsystemen überprüft.

2.2.1 Der Begriff des Portals

In der Literatur und in der Wirtschaft werden immer wieder sowohl der Begriff der Web-Applikation, als auch der Begriff des Portals gebraucht.

Portale verstehen sich als „Eingangstor“ zu verschiedenen Informationen und Funktionen [ZiWe+01, Seite 501]. Der Begriff „Portal“ wurde als *Buzzword* in den letzten Jahren häufig für verschiedenste Web-Sites ge- und miss-braucht [ZiWe+01, a.a.O.]. In Literatur und Praxis werden Portale entsprechend je nach Sichtweise in verschiedene Portal-Klassen

oder -Typen eingeteilt, die sich begrifflich zum Teil decken oder überschneiden. Die folgende Aufzählung verdeutlicht dies:

- In [ZiWe+01, Seiten 501-502] wird zwischen *horizontalen* und *vertikalen* Portalen mit Unterarten unterschieden.

  Horizontale Portale Diese Portale bilden Einstiegspunkte in das Internet [NeBa+99], indem sie in irgendeiner Form Verweise auf Internet-Seiten präsentieren. Sie Versuchen einen möglichst breiten Überblick über vorhandene Web-Seiten für eine nicht näher definierte Benutzergruppe zu geben. Ein typischer Vertreter dieser Gattung ist `Yahoo!`.

  Vertikale Portale Diese Portale fokussieren sich auf ganz bestimmte Benutzergruppen, denen eine möglichst passende Auswahl an Diensten präsentiert werden soll. Als Abkürzung für vertikale Portale wird der Begriff „Vortale“ verwendet.

  Enterprise Information Portals (EIP) Auch „Cooperate Portals“ (Unternehmensportale) genannt. Diese Vortale integrieren Daten und Dienste bezogen auf ein einzelnes Unternehmen.

  Business Portals Bieten gleiche der ähnlich eCommerce-Dienste verschiedener Unternehmen an, wie z.B. Einkaufsmeilen bestehend aus verschiedenen einzelnen Shops.

  Knowledge Portals Konzentrieren sich auf die Bereitstellung und den Austausch von Informationen.

  Intranet Portals Portale, die für die Angestellten einer Firma gedacht sind.

  B2B Portals Auch „Industry Portals“ genannt. Dienen der Abwicklung von Geschäftsprozessen zwischen verschiedenen Unternehmen.

- In [SchuSchw99, Seiten 7-10] wird nach dem Personalisierungsgrad und den eBusiness-Aktivitäten unterschieden. Bei der Unterscheidung nach den eBusiness-Aktivitäten werden folgenden Typen genannt:

  Consumer-Portal Consumer-Portale sind hoch frequentierte Web-Einstiegsseiten im Internet. Sie bieten Nutzern eine kostenlose Einstiegs- und Orientierungshilfe.

  Enterprise-Information-Portal Unterstützen die Benutzer (Kunden und Mitarbeiter), um unternehmensspezifische Informationen zu finden. Alternative Bezeichnungen sind auch „vertikale Portale“ oder „Cooperate Portals“.

  Extranet-Portal Die Zielgruppe des Extranet-Portals eines Unternehmens bilden die aktiv und potentiell kooperierenden Geschäftspartner in der erweiterten Wertschöpfungskette des Unternehmens. Die Untergruppe der „Developer-Portals“ dienen dem kooperativen Informationsaustausch zwischen Entwicklungspartnern.

  Intranet-Portal Consumer-Portale für die Angestellten eines Unternehmens.

- In [EbGu+02] werden drei grundsätzliche Kriterien für die Klassifizierung von Portalen benannt: Der Anwendungsfall, die Zielgruppe und die Funktionalität. In Bezug auf Zielgruppen werden folgende „Grundformen“ unterschieden:

  Consumer Portal Consumer Portale sind horizontale Portale als Einstiegspunkt in Web-Angebote. Sie bieten für den Kunden einen Katalog mit thematisch sortierten und zumeist ausgewählten Verweisen (Links)auf andere Web-Angebote.

  Business Portal Business Portale weisen im Gegensatz zu Consumer Portalen eine vertikale Ausrichtung auf. Sie bilden die zentrale Anlaufstelle für Kunden eines Unternehmens oder für Interessengruppen. Sie beschränken sich jedoch nicht auf die reine Information, wie zum Beispiel eine einfache Unternehmens-Website, sondern bieten eine Vielzahl von Diensten rund um ein Unternehmen und seine Produkte oder ein spezielles Interessengebiet an.

  Corporate Portal Auch „Enterprise Portal“ genannt. Sie stellen in erster Linie eine Plattform für Mitarbeiter eines Unternehmens dar, auf der mit entsprechenden Rollen und Rechten Informationen abgerufen, aber auch geschäftsrelevante Prozesse durchgeführt werden können.

  Enterprise Information Portals dienen der Aggregation und Verwaltung von unternehmensinternen Informationen.

  Enterprise Application Portals entstammen dem Zugriff auf ERP-Systeme und bieten eine Möglichkeit, Daten aus diesen Systemen zu visualisieren bzw. auf Informationen in diesen Systemen lesend oder schreibend zuzugreifen.

  Enterprise Knowledge Portals enthalten Knowledge Management-Werkzeuge zur Erfassung und Strukturierung von Wissen und dienen somit der effizienten Gestaltung des Wissenstransfers zwischen Mitarbeitern und der Unterstützung von Entscheidungen.

  Marktplatz Auf Elektronischen Marktplätzen steht in erster Linie der elektronische Handel sowie die Abwicklung von Transaktionen im Vordergrund. Charakteristisch für Marktplatzanwendungen ist, dass mehrere Anbieter über einen Marktplatz mehrere Kunden erreichen (Many to Many). Dies ist vor allem bei der Abgrenzung zu Shop-Systemen (One to Many)und E-Procurement-Systemen (Many to One).

Zusammenfassend lässt sich feststellen, dass die Verwendung des Begriffs „Portal“ sehr uneinheitlich verwendet wird. Dies ist damit zu begründen, dass die Begriffsbildung oft aus Gesichtspunkten des Marketings eines bestimmten Produkts durch den Hersteller heraus erfolgt ist und die o.g. Quellen die verschiedenen Begriffe aufgenommen und verwendet haben. Symptomatisch hierfür ist, dass der Begriff „Web-Applikation“ gerne synonym für „Portal“ verwendet wird, so z.B. [ZiWe+01, Seite 501] und [EbGu+02].

Die Abgrenzung der Begriffe „Portal“ und „Web-Applikation“ fällt schwer. So versteht z.B. [ZhKe+02] unter einer Web-Applikation eine Applikation, „die Infrastrukturen des

Webs verwendet, um ihre Funktionalität ausführen zu können ([Kri98]) und eine komplexe Anwendungslogik besitzt, im Gegensatz zu Hypermedia-Systemen oder rein Datenintensiven Anwendungen".

So einfach und klar die Unterscheidung zwischen Web-Applikation als Webseite mit „komplexer Anwendungslogik" und Portalen als einfachen „Eingangstoren" in der Theorie auch klingt, so schwierig ist in der Praxis oftmals die Abgrenzung. Dies hat mehrere Ursachen:

- Portalsysteme werden häufig mit umfangreicher Funktionalität ausgestattet, dies schließt insbesondere Funktionen für die Verwaltung von Benutzern, Rollen, Berechtigungen und - ein für Portale natürlich ganz wesentlicher Punkt - die Integration von externen Anwendungen ein (vgl. etwa die aufgezählten Funktionen in [Epi02], [Plum02], [SAPP02]). Da diese Funktionen typischerweise über ein Web-Interface bedient werden, sind sie nach der Definition von [Kri98] eindeutig als Web-Applikationen anzusehen. Ohne diese Funktionen wäre eine Portalsoftware wiederum nicht vollständig.

- Web-Applikationen, insbesondere wenn sie eine „komplexe Anwendungslogik" haben sollen, werden Informationen über den Benutzer und seine Berechtigungen brauchen, um die Ausführung der Funktionalität entsprechend anpassen zu können. Um eine Web-Applikation sinnvoll einsetzen zu können, wird sie entsprechende Funktionen zu Benutzer- und Berechtigungsverwaltung, die typische Dienste eines Portals sind, beinhalten müssen.

Portale selbst können als eine spezielle Art von Web-Applikation angesehen werden, deren Aufgabe es ist, andere Web-Applikationen aufzurufen und Informationen strukturiert anzubieten und ggf. auch aufzurufen. Die Abgrenzung, wann ein Softwaresystem als Web-Applikation und wann als Portal anzusehen ist, bleibt jedoch unscharf.

Im weiteren Verlauf dieser Arbeit wird deshalb von Portalen und Diensten anstelle von Portalen oder Web-Applikationen gesprochen, wobei folgendes gemeint ist:

Definition: Ein *Dienst* ist eine aufrufbare Funktion, die in Software realisiert ist und sich von externen Programmen aufrufen lässt.

Es ist dabei unerheblich, ob die Funktion eine komplexe Anwendungslogik enthält oder nicht. So kann beispielsweise der Dienst A einen Buchhaltungsbeleg in einem SAP R/3-System periodengerecht stornieren und der Dienst B nur den Text „Hello World!" zurück liefern.

Definition: Ein *Portal* ist ein Softwaresystem, das Dienste über eine Web-Oberfläche Anwendern zur Verfügung stellt.

Dienste können im Portalsystem selbst oder in externen Systemen realisiert sein. Ein *Unternehmensportal* fokussiert auf die Bedürfnisse eines bestimmten Unternehmens.

Eine Web-Applikation entspricht in diesem Sinne einem Portal, das einen oder mehrere Dienste anbietet. Der Begriff „Web-Applikation“ wird in dieser Arbeit nur noch dann verwendet werden, wenn sich dies auf Grund eines Zitats nicht vermeiden lässt.

Innerhalb eines Unternehmens kann es mehrere Portale geben, die teils unterschiedliche, teils gleiche Dienste anbieten. Jedes Portal ist dann auf eine Zielgruppe ausgelegt, es handelt sich hierbei um eine Form der Personalisierung, vgl. Kapitel 3.2.

Im folgenden Teil wird zunächst die typischen Anforderungen bei der Gestaltung eines Portals beschrieben, anschließend werden verschiedene Methoden und Systeme zur Erstellung von Portalen und Diensten, dort meist Web-Applikationen genannt, kurz vorgestellt.

2.2.2 Anforderungen bei der Gestaltung von Portalen

Die Gestaltung eines Portals umfasst alle Bereiche eines Portals, von der Konzeption, welche Dienste welchen Benutzern ggf. unter welchen Bedingungen an welchen Orten zur Verfügung gestellt werden sollen bis hin zur Festlegung des Layout der einzelnen Seiten.

Der Lebenszyklus eines Portals als Software entspricht dem klassischen Software-Lebenszyklus. Der Zyklus setzt sich aus den Prozessen Entwicklung (vor der Inbetriebnahme) und Pflege und Wartung (nach der Inbetriebnahme) zusammen.

Während des Entwicklungsprozesses sind aus Sicht der Gestaltung eines Portals folgende Schritte notwendig:

1. Festlegung der gewünschten Zielgruppe.
2. Festlegung der gewünschten Dienste (Funktionen) anhand der Vorstellungen der Zielgruppe.
3. Festlegung des Layouts anhand der Vorgaben der Zielgruppe.
4. Festlegung eines Berechtigungskonzepts (Rollen- und Objekt-spezifisch) anhand der organisatorischen Vorgaben.
5. Implementierung der Dienste und des Berechtigungskonzepts.

In der Praxis kann sich jeder einzelne Schritt als überaus schwierig erweisen, wie z.B. in [Ala01] beschrieben wird, wie sich aber auch in jedem Projekt zeigt, an dem mehr als eine Person beteiligt ist. An dieser Stelle soll nicht weiter auf die konkreten Probleme von Entwicklungsprozessen von Software im Allgemeinen[2] eingegangen werden, sondern nur auf die Spezifika bei der Entwicklung von Portalen.

Bei der Entstehung und dem Betrieb eines Portals sind folgende Gruppen entscheidend involviert:

- Die *Benutzer* (Endanwender): Für diese Gruppe ist das Portal gedacht. Die letztlich angebotenen Dienste und die benötigte Benutzungsoberfläche werden maßgeblich von den Anforderungen dieser Gruppe bestimmt.

[2]Eine sehr frühe Arbeit hierzu findet sich z.B. in [Roy70].

- Die *Entwickler*: Dieser Gruppe obliegt die unmittelbare technische Umsetzung auf Systemebene. Entwickler sind u.a. für die Integration der Systeme, die Einbindung der externen Dienste und die Umsetzung der Navigationsstruktur zuständig.

- Die *Web-Designer*: Diese Gruppe ist für die Umsetzung der Benutzungsoberfläche einschließlich des durchgängigen Aussehens der Seiten zuständig.

- Die *Portal-Designer*: Diese Gruppe legt fest, welche Dienste integriert werden sollen, wie die Navigationsstruktur beschaffen sein muss, welche Personalisierungen möglich sein sollen und welcher Form das Berechtigungskonzept ist. Diese Gruppe ist der (fachlichen) Entscheidungs- und Führungsebene zuzuordnen.

- Die *Administratoren*: Diese Gruppe ist für den reibungslosen Betrieb des Gesamtsystems zuständig. Zu den Aufgaben gehört auch die Umsetzung und die Pflege des Berechtigungskonzepts, wie z.B. die Einrichtung von Gruppen oder Rollen und die Zuordnung von Benutzern.

Jede der oben genannten Gruppen wird während der Entstehung und des Betriebs eines Portals mit unterschiedlichen Problemen konfrontiert:

- Benutzer müssen sich in ein System hineindenken, um die gebotenen Möglichkeiten in Ansätzen nutzen zu können. Eine Kernfrage ist dabei, wie ein Benutzer erkennen kann, auf welcher Seite des Portals ein Dienst angeboten wird und was er tun muss bzw. über welche Seiten er navigieren muss, um zu der gewünschten Seite zu gelangen.

- Portal-Designer müssen sich ein Portal zunächst abstrakt in Form von Modellen oder einfachen Prototypen vorstellen, um die Benutzeranforderungen zu Beginn der Entwicklung überhaupt formulieren zu können. Bei der Konzeption und späteren Änderungen der Dienste und ggf. der zugehörigen Berechtigungen muss der Überblick über schon vorhandene Dienste und bereits vergebene Rechte bewahrt werden. Wichtig ist später während des Betriebs auch die Abschätzbarkeit der Folgen von Veränderungen bezüglich der angebotenen Dienste, Rechte und Navigationsstruktur.

- Entwickler müssen zunächst eine gemeinsame Sprache mit der Gruppe der Portal-Designer finden, wobei es sehr hilfreich ist, wenn ein Entwickler sich bereits in der Anwendungsdomäne des beabsichtigten Portals gut auskennt. Dennoch bleibt ein hoher Aufwand vor allem für die Integration von Dienstleistungssystemen, der Implementierung des Berechtigungskonzepts und der Gestaltung des Kontrollflusses.

- Web-Designer müssen einen Kompromiss zwischen den Vorgaben und Wünschen der Portal-Designer, ggf. den unternehmensweiten Gestaltungsrichtlinien und den beschränkten Möglichkeiten HTML-basierter Oberflächen finden. Muss für jede Seite eines Portals einzeln ein Template erstellt werden, wie dies beispielsweise bei dem infoAsset Broker oder dem SAP Web Applikation Server der Fall ist, fällt zusätzlich ein hoher Aufwand für das manuelle Layout aller Seiten an.

- Alle Gruppen gemeinsam haben das Problem, die Sprache der anderen Gruppen zu verstehen, da jede Gruppe ihren eigenen Sprachgebrauch pflegt. Ohne ein gemeinsames Verständnis ist jede Entwicklung zum Scheitern verurteilt.

Aus diesen Problemen heraus lassen sich allgemeine Anforderungen an Modelle oder Systeme für die Gestaltung von Portalen ableiten:

Endbenutzertauglichkeit Benutzer benötigen ein einfaches, nicht-technisches Modell, das auch den gelegentlichen Benutzer nicht überfordert. Angebotene Dienste und Navigationsmöglichkeiten müssen klar und während der Benutzung kommuniziert werden.

Abteilungsleitertauglichkeit Portal-Designer benötigen ein verständliches Modell, das die angebotenen Dienste, die Navigationsstruktur, die Berechtigungen und die Personalisierungsmöglichkeiten auf der Ebene des Portalsystems, detailliert beschreibt, ohne gleich auf die Ebene der Implementierung zu wechseln.

Entwicklertauglichkeit Entwickler benötigen ein Modell, das klare Schnittstellen für die Integration von Diensten, Berechtigungskonzepten und Personalisierungsmöglichkeiten auf der Ebene des Portalsystems definiert, so dass sich die Entwickler auf die Probleme der Integration konzentrieren können und nicht die Aufgaben der Portal-Designer teilweise mit übernehmen müssen, wie dies so häufig der Fall ist.

Web-Designertauglichkeit Web-Designer benötigen ein Modell, das es erlaubt, aus einem allgemeinen Rahmen für die Benutzungsoberfläche die notwendigen Templates automatisch zu generieren. Die ermöglicht es den Web-Designern, sich auf die Definition und Implementierung des Rahmens zu konzentrieren.

Kommunikationstauglichkeit Alle Gruppen gemeinsam benötigen ein Modell, das ihnen hilft, die Probleme und Anliegen der anderen Gruppen besser zu verstehen.

Aus Sicht der Effizienz und der Akzeptanz des Modells gesellt sich eine weitere wichtige Anforderung hinzu:

Arbeitserleichterung Das Modell muss allen Beteiligen Arbeit abnehmen und nicht Zusätzliche aufbürden.

Im folgenden Abschnitt werden einige ausgewählte Entwicklungsmodelle und Portalsystem daraufhin untersucht, inwieweit sie den oben aufgestellten Anforderungen genügen.

2.2.3 Entwicklungsmethoden für Portale und Dienste

In der Literatur werden zahlreiche Entwicklungsmethoden und -systeme für Portale und Dienste vorgeschlagen.

Die Entwicklungsmethoden lassen sich grob in folgende Kategorien einordnen:

1. Daten-orientierte Methoden: Diese Methoden modellieren die darzustellenden oder zu ändernden Daten meist auf relationale Art und Weise. Stark vereinfacht gesagt werden dabei einzelne Datensätze auf einer Seite abgebildet, Tabellen werden als Aggregation in Listenform angezeigt. Navigationspfade (Verweise oder engl. Links) entstehen durch Aggregation, durch Fremdschlüsselbeziehungen und explizite Modellierung.

   In diese Klasse gehören z.B. autoweb ([FrPa98], [FrPa00]), HDM ([GaPa+93]), RMM ([IsSt+95]), WebML ([CrFr+00], [BrCo+02]) oder HTM ([Zol01]).

2. Objekt-Orientierte Methoden: Diese Methoden modellieren die Informationsobjekte, die in einem Portal präsentiert oder geändert werden sollen. Als Modellierungssprache kommt meistens UML ([FoSc98]) mit speziellen Erweiterungen zum Einsatz. Die Navigationspfade orientieren sich an den Beziehungen zwischen den Objekten.

   In diese Klasse gehören z.B. OO-Method ([PaIn+97], [PaPe+98]), OO-HMethod ([GoCa+01]), OO-HDM ([RoSc+99]) oder UWE ([KrKo02]).

3. Integrative Methoden: Diese Methoden konzentrieren sich auf die Steuerung der Abläufe und Kommunikation mit dem Benutzer. Viele dieser Ansätze stammen aus der Praxis wie z.B. struts[3], cocoon ([ZiLa02]) oder SAP Portals und der SAP Web Application Server.

4. Sonstige Methoden: Diese Methoden verfolgen jeweils ihren ganz eigenen Ansatz.

   So wird etwa bei STRUDEL ([FeFl+00]) ein Portal deklarativ mit Hilfe von Ausdrücken der Sprache StruQL beschrieben. Der oder die StruQL-Ausdrücke bestimmen die Struktur des Portals (Seiten und Navigationsmöglichkeiten), das Ergebnis eines Ausdrucks die konkret darzustellenden Daten bzw. Objekte.

   Bei dem SEAL-Ansatz ([MaSt+02]) werden Ontologien zur Beschreibung eingesetzt, ein Ansatz, der aus dem Bereich der künstlichen Intelligenz und des Wissensmanagements stammt.

   Der von Brandl ([Bra02]) beschriebene Ansatz verwendet ein hierarchisches Datenmodell, ein Aufgaben- und ein Benutzermodell, aus dem allgemeine Informationssysteme, also nicht nur solche mit Web-Oberflächen generiert werden können.

Das Vorgehen der meisten oben erwähnten Methoden beruht darauf, zunächst die gewünschten Daten bzw. Informationsobjekte zu modellieren, dann ein Navigationsmodell zu erstellen und letztlich die gewünschten Seiten ggf. mit Personalisierungsmöglichkeiten zu erstellen. In [MaFr02] findet sich eine relativ umfassende Gegenüberstellung verschiedener kommerzieller Werkzeuge für die Entwicklung von personalisierten Web-Applikationen.

Im Folgenden wird beispielhaft für Daten-orientierte Methoden die Entwicklungsmethode WebML näher untersucht, die dem oben angedeutetem Vorgehen folgt. Die Darstellung lehnt sich dabei an die in [CrFr+00] gegebene Darstellung an, ergänzt um die in [WebR02] beschriebenen Erweiterungen.

[3] http://jakarta.apache.org/struts, November 2002

2.2.4 Die Modellierungssprache WebML

Die Web Modeling Language (kurz WebML) ist eine abstrakte Beschreibung einer WebSite mit Hilfe orthogonaler Dimensionen (Modelle). Es werden folgende Modelle verwendet:

- Strukturmodell (Structural Model): Beschreibt die zugrunde liegenden Daten mit Hilfe eines Entity Relationship (ER)-Modells. Mit Hilfe einer vereinfachten Form der OQL (Object Query Language) können auch aus dem Modell abgeleitete Daten dargestellt werden.

- Hypertextmodell (Hypertext Model): Beschreibt Ausprägungen des Portals, sog. Site Views. Eine Ausprägung setzt sich wiederum aus zwei Modellen zusammen:

  - Kompositionsmodell (Composition Model): Beschreibt, welche Seiten das Portal enthalten soll und welche Einheiten (Units) auf welcher Seite enthalten sind.
  - Navigationsmodell (Navigation Model): Beschreibt die Verweise zwischen den Seiten und Einheiten. Es werden kontextabhängige (contextual) und kontextfreie (non-contextual) Verweise unterschieden. Kontextabhängige Verweise transportieren Daten von der Quelle des Verweises zum angegebenen Ziel, kontextfreie Verweise transportieren keine Daten.

- Präsentationsmodell (Presentation Model): Beschreibt das Layout und das grafische Erscheinungsbild der Seiten in XML unabhängig vom konkreten Ausgabegerät.

- Personalisierungsmodell (Personalization Model): Beschreibt mögliche Anpassungen des Portals. Die Entitäten „Benutzer“ und „Gruppe“ werden explizit im Strukturmodell mit den gewünschten Eigenschaften modelliert. In den Entitäten können Informationen gespeichert werden, aus denen mit Hilfe von OQL benutzerspezifische, neue Entitäten für das Strukturmodell gewonnen werden können. Als zweite Möglichkeit der Personalisierung können ECA-Regeln (Event-Condition-Action-Regeln) angegeben werden, die in Abhängigkeit des definierten Ereignisses Aktionen im Portal auslösen können.

Eine entsprechende Software, verfügbar unter www.webratio.com[4], übersetzt die Modelle in das gewünschte Ausgabeformat, wie z.B. HTML oder WML. Auf die Daten wird im Rahmen einer J2EE-Plattform über JSP zugegriffen.

Folgender, iterativer Entwurfsprozess wird als typisch für Entwicklungen mit WebML angegeben:

1. Aufnahme der Anforderungen: Was ist das Hauptziel des Portals, wer ist die Zielgruppe, welche Personalisierungen sind gewünscht?

2. Entwurf der Daten: Entwurf des Strukturmodells.

[4]Stand November 2002

3. Entwurf des Hypertextes „im Großen“: Welche Seiten mit welchen Einheiten und Verweisen.

4. Entwurf des Hypertextes „im Kleinen“: Betrachtung jeder einzelnen Seite, jeder einzelnen Einheit und jedes einzelnen Verweises.

5. Entwurf des Präsentationsmodells: Welches Aussehen sollen die Seiten erhalten?

6. Entwurf der Benutzer und Gruppen: Festlegung der Eigenschaften und ggf. Entwurf spezieller Portale (Site Views) für einzelne Gruppen.

7. Entwurf der Personalisierungen.

Der erste Schritt wird von allen Beteiligten durchgeführt, der zweite Schritt obliegt einem Datenexperten, die Schritte 3 und 4 nimmt ein Experte für Web-Applikationen vor, Schritt 5 benötigt einen Web-Designer, die übrigen Schritte 6 und 7 werden von einem Web-Administrator vorgenommen.

An einem Beispiel, entnommen aus [CrFr+00, Seiten 140ff.], sollen die Konzepte der WebML veranschaulicht werden. In dem Beispiel soll ein Portal entwickelt werden, in dem Informationen über Musiker und deren Alben angeboten werden. Zu einem Musiker werden zusätzlich auch Kritiken aufgeführt, zu den Alben sind die enthaltenen Lieder bekannt.

Strukturmodell

Das Fundament des Strukturmodells sind Entitäten und Beziehungen. Entitäten bestehen aus Datenelementen, die einen Namen und einen Typ haben. Mengenwertige Datenelemente sind in Komponenten organisiert, die den klassischen „ist Teil von“-Beziehungen entsprechen. Beziehungen können Einschränkungen bezüglich ihrer Kardinalitäten und einen Rollennamen haben.

Folgender Auszug aus dem Strukturmodell definiert eine Entität „Album“:

```
<ENTITY id="Album">
  <ATTRIBUTE id="title" type="String"/>
  <ATTRIBUTE id="cover" type="Image"/>
  <ATTRIBUTE id="year" type="Integer"/>
  <COMPONENT id="Support" minCard="1" maxCard="N">
    <ATTRIBUTE id="type" userType="SupportType"/>
    <ATTRIBUTE id="listPrice" type="Float"/>
    <ATTRIBUTE id="discountPercentage" type="Integer"/>
    <ATTRIBUTE id="currentPrice" type="Float"
              value="Self.listPrice*(1-(Self.discountPercentage /100))"/>
  </COMPONENT>
  <RELATIONSHIP id="Album2Artist" to="Artist" inverse="ArtistToAlbum"
                minCard="1" maxCard="1"/>
  <RELATIONSHIP id="Album2Track to="Track" inverse="Track2Album"
```

```
                minCard="1" maxCard="N"/>
</ENTITY>
```

Die definierte Entität „Album“ besteht aus einfachen Attributen, z.B. „title“, der mengenwertigen Komponente „Support“, die die verfügbaren Medien (CD, Kassette, etc.), den Listenpreis, den Rabatt und – als abgeleitetes Datenelement – den aktuellen Preis enthält. Es werden zusätzlich zwei gerichtete Beziehungen - spätere mögliche Navigationspfade - definiert, einmal vom Album zum Künstler und einmal vom Album zu den Tracks.

Das gesamte Strukturmodell des Beispiels besteht aus vier Entitäten (Artist, Album, Review, Track) und drei Beziehungen (Artist2Album, Artist2Review, Album2Track) und deren Inverse.

Kompositionsmodell

Das Kompositionsmodell beschreibt die Seiten und Einheiten des Portals. Es stehen 8 Typen von Einheiten (6 aus [CrFr+00], 2 aus [WebR02]) zur Verfügung, zu denen es jeweils eine eigene grafische Repräsentation existiert, mit deren Hilfe ein grafisches Gesamtmodell des Portals entsteht.

Dateneinheiten Dateneinheiten bilden einfache Sichten auf Entitäten ab. Für die Definition einer Dateneinheit müssen die zu Grunde liegende Entität und die eingebundenen Attribute benannt werden.

Im Beispiel wird u.a. folgende Dateneinheiten definiert:

```
<DATAUNIT id="ShortArtist" entity="Artist">
  <INCLUDE attribute="firstName"/>
  <INCLUDE attribute="lastName"/>
  <INCLUDE attribute="photo"/>
</DATAUNIT>
```

Abbildung 2.1 zeigt die Visualisierung in WebML und eine HTML-basierte Implementierung.

Mengenwertige Dateneinheiten Mengenwertige Dateneinheiten repräsentieren viele Dateneinheiten der identischen Struktur. Für die Definition werden die zu Grunde liegende Komponente, aus der die Daten stammen, und die Dateneinheit, nach der die Daten präsentiert werden sollen, benötigt. Im Beispiel wird folgende mengenwertige Dateneinheit definiert:

```
<MULTIDATAUNIT id="MultiAlbumUnit" entity="Album">
  <DATAUNIT id="AlbumUnit" entity="Album">
    <INCLUDEALL/>
  </DATAUNIT>
</MULTIDATAUNIT>
```

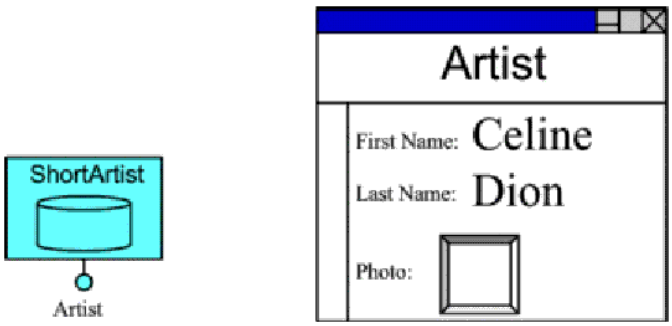

Abbildung 2.1: Grafische Notation für Dateneinheiten und eine mögliche Umsetzung in HTML.

Die grafische Notation für mengenwertige Einheiten lehnt sich eng an die Notation für einfache Dateneinheiten an.

Indexeinheiten Indexeinheiten repräsentieren Listen von Entitäten oder Komponenten. Die Definition umfasst zum Einen die Entität oder Komponente, aus der die Daten kommen, und zum Anderen die Attribute, die als Schlüssel für den Index dienen. Im Beispiel wird folgende Indexeinheit definiert:

```
<INDEXUNIT id="AlbumIndex" entity="Album">
  <DESCRIPTION Key="title"/>
</INDEXUNIT>
```

Abbildung 2.2 gibt die grafische Repräsentation und eine mögliche Implementierung in HTML wieder.

Abbildung 2.2: Grafische Notation von Indexeinheiten und Umsetzung in HTML.

Scroll-Einheiten Scroll-Einheiten definieren folgende Elemente zur Navigation in Mengen von Entitäten oder Komponenten: Erster, voriger, nächste, letzter. Scroll-Einheiten

werden zumeist in Verbindung mit einer Dateneinheit verwendet, die zur Darstellung der aktuellen Entität oder der aktuellen Komponente dient. Im Beispiel wird folgende Scroll-Einheit definiert:

```
<SCROLLERUNIT id="AlbumScroll" entity="Album" first="yes"
              last="yes" previous="yes" next="yes"/>
```

Die Visualisierung wird in Abbildung 2.3 gezeigt.

Abbildung 2.3: Grafische Notation von Scroll-Einheiten und Umsetzung in HTML.

Filtereinheiten Filtereinheiten dienen der Eingabe von Suchkriterien und werden in Verbindung mit Indexeinheiten oder mengenwertigen Dateneinheiten verwendet. Filtereinheiten beziehen sich auf eine Entität, eine Beziehung oder eine Komponente und definieren Suchkriterien. Im Beispiel wird folgende Filtereinheit definiert:

```
<FILTERUNIT id="AlbumFilter" entity="Album"/>
  </SEARCHATTRIBUTE name="title" predicate="like">
  </SEARCHATTRIBUTE name="year" predicate="between">
</FILTERUNIT>
```

Abbildung 2.4 gibt die grafische Darstellung in WebML und eine mögliche Umsetzung in HTML wieder.

Direkte Einheiten Direkte Einheiten sind spezielle Indexeinheiten, die eine eins-zu-eins Beziehung zwischen zwei Objekten ausdrücken. Im Gegensatz zu Indexeinheiten werden direkte Einheiten nicht angezeigt. Im Beispiel wird folgende direkte Einheit definiert:

```
<DIRECTUNIT id="ToArtist" relation="Album2Artist"/>
```

Die grafische Darstellung von direkten Einheiten zeigt Abbildung 2.5.

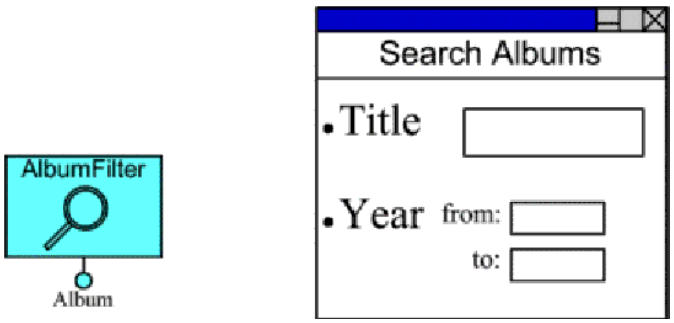

Abbildung 2.4: Grafische Notation von Filtereinheiten und Umsetzung in HTML.

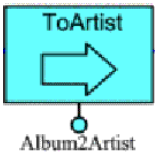

Abbildung 2.5: Grafische Darstellung einer direkten Einheit.

Dateneingabeeinheiten und Operationseinheiten Dateneingabe- und Operationseinheiten dienen der Eingabe und Änderung von Daten. Eine Dateneingabeeinheit ist eine sichtbare Einheit, die Felder zur Eingabe zur Verfügung stellt. Eine Dateneingabeeinheit hat einen oder mehrere Verweise, die die Eingaben des Benutzer als Kontext zu dem jeweiligen Ziel transportieren. Die Verweise entsprechen den SUBMIT-Schaltflächen in HTML.

Operationseinheiten werden verwendet, um beliebige externe Operationen aufzurufen, es gibt jedoch die vordefinierten Einheiten zum Erstellen, Ändern und Löschen von Entitäten und Beziehungen. Sie verfügen über einen aktivierenden Eingangsverweis, der in der grafischen Notation mit einem „A“ markiert wird. Die Ausführung der Operation kann eine beliebig lange Interaktion mit externen Systemen einschließen. Das Ergebnis einer Operation kann über die ausgehenden Verweise als Kontext weiter transportiert werden. Ausgehende Verweise können in Abhängigkeit des Ausgangs der Operation definiert werden, um eine Fehlerbehandlung zu ermöglichen. Im Gegensatz zu allen anderen Einheiten gehören Operationseinheiten nicht zu Seiten, d.h. sie können dort auch nicht platziert werden.

Im Beispiel ([WebML02, Seiten 67-68]):

```
<ENTRYUNIT id="ArtistEntryUnit">
  <FIELD id="fName" preloaded="no" modifiable="yes" hidden="no"
         type="String" contentType="text/plain" />
  <FIELD id="lName" preloaded="no" modifiable="yes" hidden="no"
         type="String" contentType="text/plain" />
  <LINK id="ArtistCreatUnit" activating="yes" />
```

```
  <LINKPARAMETER id="FName" source="Name" dest="firstName"
                 type="String" />
  <LINKPARAMETER id="LName" source="lName" dest="lastName"
                 type="String" />
</ENTRYUNIT>

<CREATEUNIT id="ArtistCreateUnit" entity="Artist">
  <OK-LINK id="link2" to="ArtistDetails" />
  <KO-LINK id="link3" to="ArtistCreationPage" />
</CREATEUNIT>
```

Die Daten werden über die Dateneingabeeinheit „ArtistEntryUnit“ erfasst, die Operationseinheit „ArtistCreateUnit“ versucht den neuen Datensatz anzulegen. Ist die Anlage erfolgreich, so werden die neuen Daten angezeigt, schlägt die Anlage fehl, so wird zurück zur Eingabe verzweigt. Abbildung 2.6 zeigt die Anlage eines neuen Musikers.

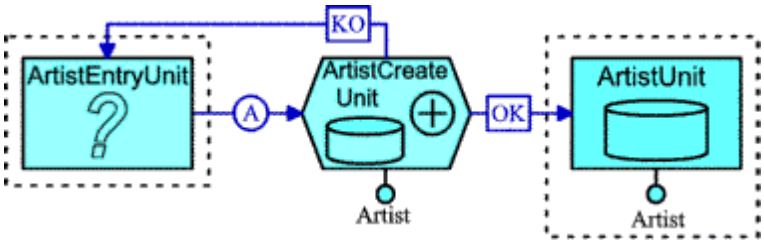

Abbildung 2.6: Anlage eines neuen Musikers in WebML.

Seiten Die Anforderungen an ein Portal können es erfordern, dass mehrere Einheiten zur gleichen Zeit sichtbar sind. Eine Seite in WebML ist eine Abstraktion einer abgeschlossenen Region auf dem Ausgabegerät, die einen Block in der Benutzeroberfläche darstellt.

Seiten können Einheiten und/oder andere Seiten enthalten. In Seiten enthaltene Seiten, sog. Unterseiten, können konjunktiv (AND) oder disjunktiv (OR) enthalten sein. Konjunktiv enthaltene Seiten werden gleichzeitig, disjunktive Seiten werden alternativ zueinander angezeigt. Insbesondere die Disjunktion erlaubt die Personalisierung einer Ausgabe.

Im Beispiel wird folgende Seite mit zwei Unterseiten definiert:

```
<PAGE id="outermost">
  <PAGE id="leftmost">
    <UNIT id="pastIndex"/>
    <UNIT id="thisYearIndex"/>
   </PAGE >
   <PAGE id="rightmost">
     <UNIT id="AlbumInfo"/>
   </PAGE >
```

```
</PAGE>
```

In der Visualisierung werden Seiten durch gestrichelte Vierecke symbolisiert, in deren linken oberen Ecke der Name der Seite steht, vgl. Abbildung 2.7.

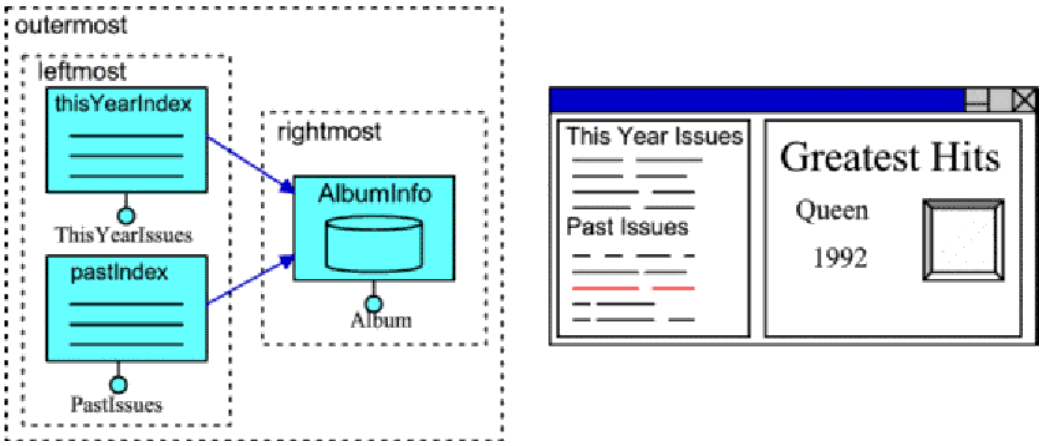

Abbildung 2.7: Grafische Darstellung von Seiten in WebML und eine Umsetzung in HTML.

Mit Hilfe des nun vorliegenden Kompositionsmodells kann nun das Navigationsmodell erstellt werden.

Navigationsmodell

Das Navigationsmodell legt fest, auf welche Weise die Einheiten und Seiten verbunden sind, d.h. welche Verweise gewünscht sind. Bei Verweisen werden zwei Arten unterschieden:

1.Kontextabhängige Verweise: Diese Verweise verbinden Einheiten auf eine Art, die mit dem Strukturmodell überein stimmt. Hierbei werden Daten als Kontext von der Quelle des Verweises zu seinem Ziel transportiert.

2.Kontextunabhängige Verweise: Diese Verweise verbinden nicht Einheiten, sondern Seiten und das auf völlig freie Art und Weise. Es werden keine Daten als Kontext transportiert.

Kontextabhängige Verweise werden durch das Schlüsselwort INFOLINK definiert, kontextunabhängige Verweise durch HYPERLINK.

Das folgende Beispiel zeigt die Verwendung von kontextabhängigen Verweisen:

```
<DATAUNIT id="ArtistUnit" entity="Artist">
  <INCLUDEALL/>
  <INFOLINK id="link1" to="AlbumIndex"/>
</DATAUNIT>

<INDEXUNIT id="AlbumIndex" relation="Artist2Album">
  <DESCRIPTION key="title"/>
```

```
  <INFOLINK id="link2" to="AlbumUnit"/>
</INDEXUNIT>

<DATAUNIT id="AlbumUnit" entity="Album">
  <INCLUDEALL/>
</DATAUNIT>
```

Die Verweise ermöglichen eine Navigation vom Musiker über eine Liste seiner Alben zu einem einzelnen Album. Abbildung 2.8 stellt den Navigationspfad grafisch dar.

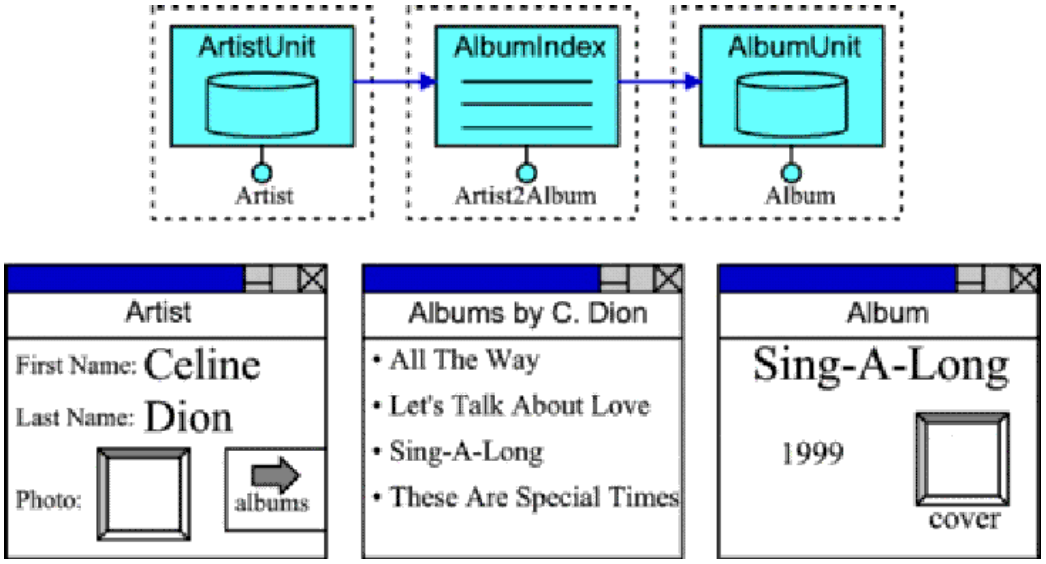

Abbildung 2.8: Navigation mit Hilfe kontextabhängiger Verweise in WebML und HTML.

Folgende Informationen werden in Abhängigkeit der Quell-Einheit durch einen kontextabhängigen Verweis transportiert:

- Dateneinheit und Mengenwertige Dateneinheit: Der Schlüssel (Id) des Objektes, das aktuell angezeigt wird.
- Indexeinheit: Der aus der Liste ausgewählte Schlüssel.
- Scroll-Einheit: Der Schlüssel des durch das Kommando ausgewählten Objekts.
- Filtereinheit: Die vom Benutzer eingegeben Suchwerte.
- Direkte Einheit: Der Schlüssel des Objekts.
- Dateneingabeeinheit: Die eingegebenen Werte.
- Operationseinheit: Implizit erzeugter Kontext.

Zusätzlich können globale Parameter definiert werden, die jeder Seite zur Verfügung stehen und in denen beliebige Werte abgelegt werden können. Die globalen Parameter sind vom aktuellen Benutzer abhängig, sie werden typischerweise in der Session abgelegt ([WebML02, Seite 52]).

Portale

Portale (Site Views) werden durch die zugeordneten Seiten definiert. Im Beispiel wird die Seite „Homepage“ definiert, die durch kontextfreie Verweise auf die Seiten „AllReviewsPage“, „AllArtistsPage“, „AllAlbumsPage“ und „AllTracksPage“ verweist. von diesen Seiten aus wird dann durch kontextabhängige Verweise weiter verzweigt. Die grafische Darstellung des gesamten Portals zeigt Abbildung 2.9.

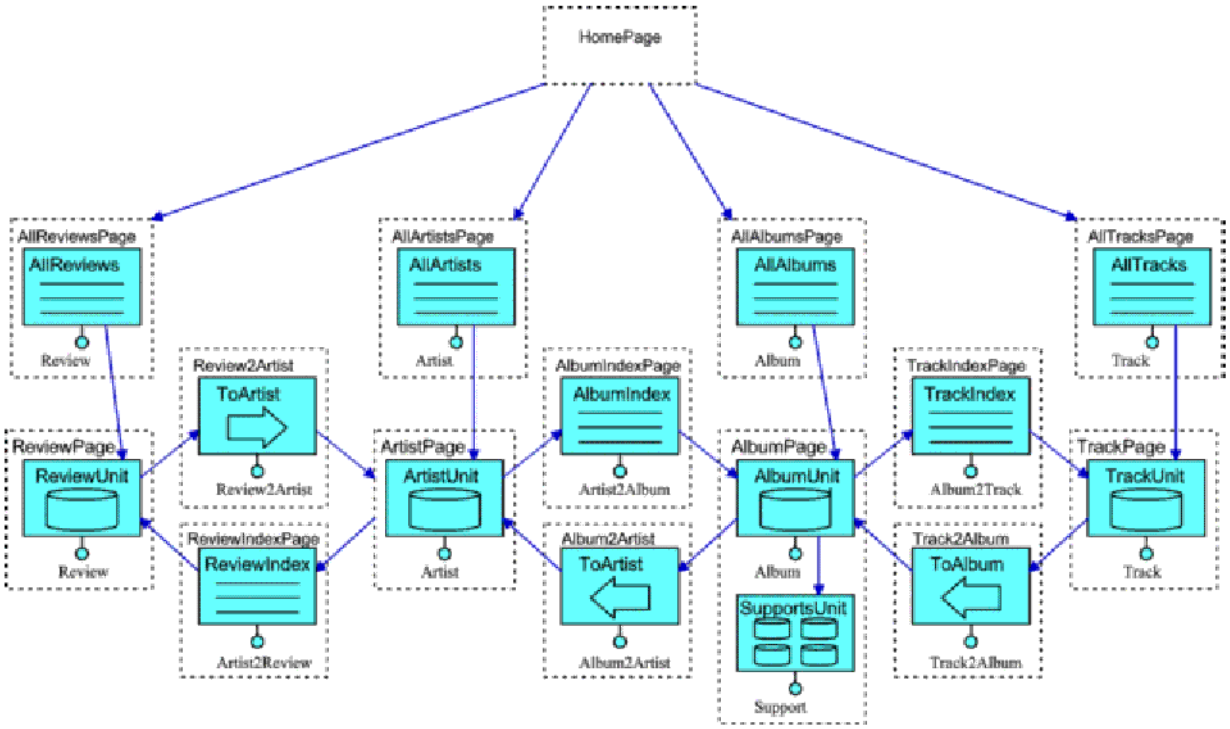

Abbildung 2.9: Darstellung des Beispiel-Portals in WebML.

Das resultierende Portal besteht aus 12 sichtbaren Seiten; die drei direkten Einheiten, die im Portalmodell enthalten sind, bilden keine eigenen Seiten, sondern nur Navigationspfade. Sie Seite „Hompage“ besteht aus 4 Verweisen, die als Einstieg zu den eigentlichen Seiten des Portals dient.

Bewertung

WebML erfüllt die in Kapitel 2.2.2 (Seite 19) benannten Anforderungen nur sehr wenig:

Endbenutzertauglichkeit WebML ist nicht für die Kommunikation mit dem Endbenutzer gedacht und auch nicht geeignet, da sehr viele, recht technische Konzepte enthält.

Abteilungsleitertauglichkeit Es ist zu unterstellen, dass ein etwas überdurchschnittlich technisch ausgerichteter Abteilungsleiter in der Lage sein könnte, WebML oberflächlich zu verstehen. Seine Anliegen hinsichtlich der Berechtigungen und Personalisierungen werden jedoch völlig ignoriert, da sich dies auf der Ebene der Implementierung abspielt.

Entwicklertauglichkeit WebML ist nur sehr bedingt für Entwickler geeignet, die nicht nur neue, d.h. mit WebML erstellte, Dienste nutzen wollen, sondern auch bereits bestehende Dienste aus betrieblichen Informationssystemen nutzen wollen. Die Integration bestehender Dienste geschieht einfach über eine „black box“, Darstellung oder Inhalt der Dienste sind nicht in WebML darstellbar.

Web-Designertauglichkeit Da Portale aus WebML heraus generiert werden können, können Web-Designer sich um das allgemeine Design kümmern, das dann bei der Generierung einer Seite verwendet werden kann. WebML erfüllt die Anforderung der Web-Designertauglichkeit.

Kommunikationstauglichkeit WebML ist für die gemeinsame Kommunikation zwischen den verschiedenen Gruppen nicht geeignet, da viele Belange der einzelnen Gruppen nicht berücksichtigt werden.

Arbeitserleichterung Eine Arbeitserleichterung ist WebML nur in wenigen Bereichen, da viele Dinge eines Portals, z.B. die bestehenden Dienste oder die Personalisierungsmöglichkeiten, „von Hand“ hinzuprogrammiert werden müssen.

Nachdem mit WebML eine Modellierungssprache für Dienste und gleichzeitig Portale vorgestellt wurde, soll nun mit der von SAP entwickelten mySAP Technologie eine Produktfamilie untersucht werden, in der Portale und Dienste getrennt betrachtet werden.

2.2.5 mySAP Portals und Web Application Server

Die von SAP entwickelte mySAP Technologie dient dem Ziel, unternehmensübergreifende Geschäftsprozesse zu ermöglichen. Um solche Prozesse aus Sicht der IT zu ermöglichen, müssen vorhandene, heterogene Systemlandschaften integriert werden können. Die mySAP Technologie versucht eine solche Integration auf folgende Weise zu unterstützen ([SAPW02, Seite 5]:

- Integration von Anwendungen: mySAP Technologie ist eine Web-Infrastruktur, die auf offenen Standards basiert. Sie kann im Prinzip jede beliebige Anwendung integrieren.

- (Pseudo-)Integration von Daten: Daten, die redundant in verschiedenen Systemen abgelegt sind, werden über eine gemeinsame, nachrichtenbasierte Infrastruktur abgeglichen und damit konsistent gehalten.

- Syndizierung von Web-Services: Von verschiedenen Systemen angebotenen Web-Services (Dienste) können gemeinsam genutzt werden.

Zu den zentralen Elementen der mySAP Technologie gehören:

- Die Portal-Infrastruktur, die für eine benutzerorientierte Integration und Zusammenarbeit sorgt, realisiert als Komponente mySAP Portals.

- Der Web Application Server für Anwendungskomponenten und Web-Services.

- Die Exchange-Infrastruktur, die eine prozessorientierte Integration und Zusammenarbeit sowie durchgängige Geschäftsprozesse gewährleistet. Sie bewerkstelligt die Datenkonsistenz.

- Die Infrastruktur-Services, die u.a. die Aspekte Sicherheit, Benutzerverwaltung und Systemverwaltung abdecken.

In diesem Kapitel wird hauptsächlich auf die Portals-Komponente und auf den Web Application Server eingegangen, die aus Sicht der Portale und Dienste die Hauptkomponenten sind. Die Funktion der anderen Bestandteile der mySAP Technologie werden dort erwähnt, wo sie für die Hauptkomponenten von Belang sind.

Abbildung 2.10 stellt die Architektur der mySAP Technologie dar. Dabei interagieren zwei Komponenten mit dem Benutzer bzw. Client: Die Portals-Komponente und der Web Application Server. Die Kommunikation findet über das HTTP(S)-Protokoll statt. Der Web Applikation Server kann über die Integration Engine zusätzlich die in ihm enthaltenen Enterprise JavaBeans (EJBs) und Business Objects (BOs) automatisch als Web-Services ([ChCu+01]) über das SOAP-Protokoll ([W3C01]) zur Verfügung stellen. Die Portals-Komponente integriert über den Unification-Mechanismus Daten aus verschiedenen Applikation mit Hilfe von Metadaten und stellt diese zur Visualisierung über iViews zur Verfügung. Der Datenzugriff ist dabei rein lesender Natur. Die Web Dynpros des Web Application Server ermöglichen eine dialogorientierte Darstellung und Änderung von Daten in Applikationen. Dabei geschieht die Kommunikation mit den Applikationen über geeignete EJBs und/oder Business Objects, einer objekt-orientierten Erweiterung der ABAP-Sprache. Der Exchange-Server erfüllt die elementare, nicht triviale Aufgabe, die Datenbestände in den verschiedenen Applikationen konsistent zueinander zu halten. Dies geschieht über sog. Adaptoren, die einerseits Änderungen im Datenbestand einer Anwendung an den Exchange-Server melden, andererseits Änderungen, die über andere Adaptoren gemeldet wurden, an „ihre“ Anwendung weitergeben können. Nähere Einzelheiten zum Exchange-Server finden sich z.B. in [SAPE02]. Das Portal Content Directory (PCD) speichert u.a. Benutzerdaten, Rollen und verfügbare Dienste und stellt diese für andere Komponenten und Applikationen z.B. über die Protokolle LDAP bzw. UDDI zur Verfügung.

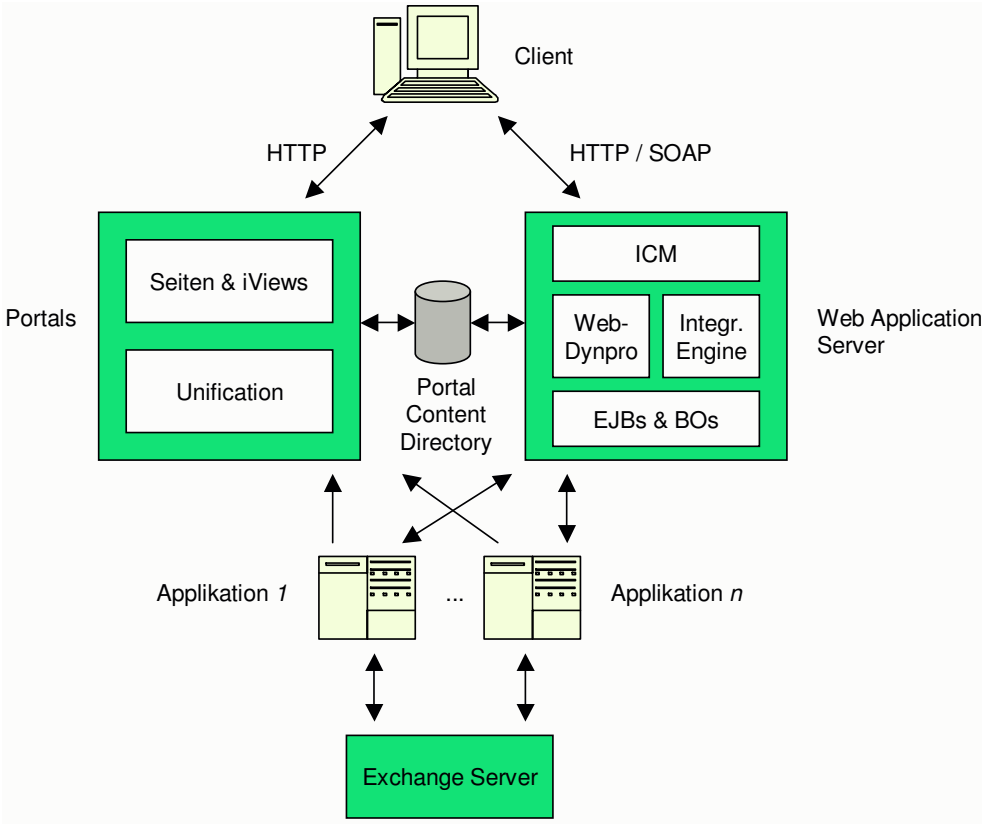

Abbildung 2.10: Überblick über die Architektur der mySAP Technologie.

Im Folgenden wird zunächst die Portals-Komponente als zentrales Integrations- und Personalisierungselement der Benutzungsoberfläche näher beschrieben. Anschließend werden die wesentlichen Konzepte des Web Application Servers in Bezug auf Dienste und die Navigation zwischen Seiten und Diensten vorgestellt.

mySAP Portals

SAP sieht vier Gruppen von Informationsquellen in einem Unternehmensportal ([SAPP02, Seite 7]):

- Transaktionssysteme und konventionelle Datenbanken
- Data Warehouses und Analysen
- Unstrukturierte Dokumente
- Internet

Das Portal sollte zusätzlich „den Benutzer, sein Profil, seine Zuständigkeiten und Erfordernisse kennen, um eine Funktion oder Anfrage bearbeiten zu können“ ([SAPP02, a.a.O.]).

Für jede der vier Gruppen, von SAP die „vier Säulen des Unternehmensportals“ genannt, werden Lösungen für den Zugriff angeboten: Unification und iViews dienen dem Zugriff auf Transaktionssysteme und Datenbanken, die Business Intelligence Services stellen Data Warehousing-Funktionen zur Verfügung, das Knowledge Management (zutreffender wäre die Bezeichnung Dokumentenmanagement) behandelt die Verwaltung von unstrukturierten Dokumenten und das Content und Services Management bietet die Möglichkeit, Informationsdienste der Firma *Yahoo!* in das Portal zu integrieren. Abbildung 2.11 stellt Portal-Architektur im Zusammenhang dar. Ein mit „R“ beschrifteter Pfeil zeigt dabei die Bearbeitungsrichtung eines Requests an.

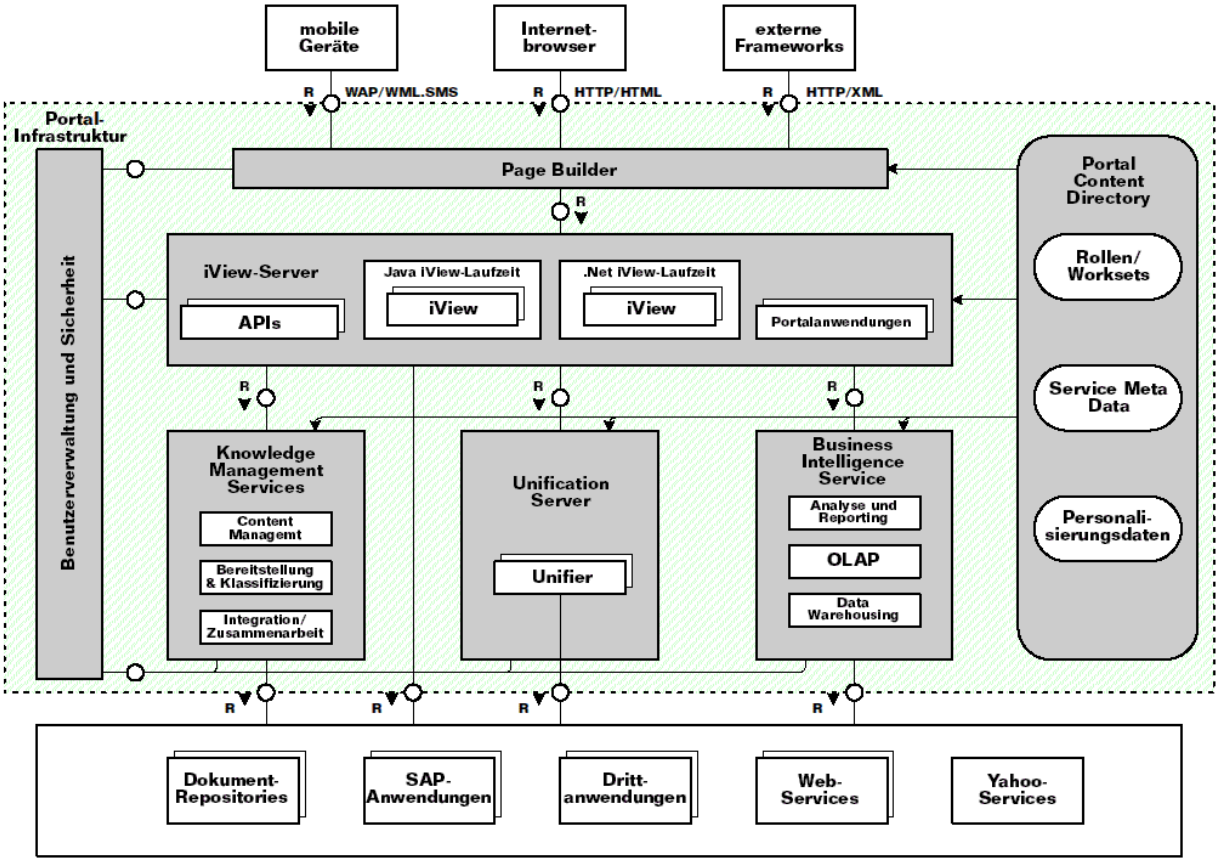

Abbildung 2.11: Die SAP-Portals-Architektur ([SAPP02, Seite 8])

Die Erzeugung von (Antwort-)Seiten erfolgt über einen sog. Page Builder, der für das einheitliche Aussehen der aus verschiedenen zu Grunde liegenden Anwendungen verantwortlich ist. Eine Seite hat dabei folgenden, von SAP vorgeschlagenen, prinzipiellen Aufbau (vgl. Abbildung 13): Am oberen Rand befinden sich Kopf- und Titelfeld sowie eine

Top-Level-Navigationsleiste. Auf der linken Seite wird ein iPanel angezeigt, zumeist ein Menü, das gemäß der verschiedenen Rollen eines Benutzers aufgebaut ist und hierarchisch verschachtelt sein kann. Der Hauptteil der Seite wird vom Workset eingenommen, der seinerseits verschiedene iViews enthalten kann.

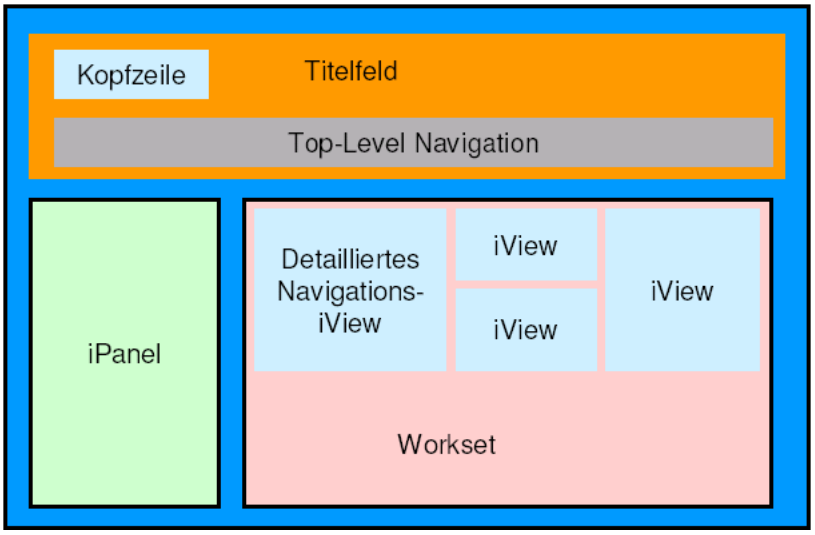

Abbildung 2.12: Grundsätzlicher Aufbau einer SAP-Portals-Seite ([Zah02, Seite 19]).

Unter einer Rolle wird eine Zusammenstellung von Worksets verstanden, die für eine bestimmte Aufgabenstellung in ihrer Zusammenstellung optimiert sind. So könnten z.B. für die Rolle „Projektleiter“ ein Workset mit den aktuellen Kennzahlen des Projektes angezeigt werden. Worksets enthalten dabei iViews, die die eigentliche Sicht auf die Daten ermöglichen. Rollen, Worksets und zulässige iViews werden dabei von einem Administrator festgelegt, der auch die Zuordnung von Benutzern zu Rollen vornimmt. Dabei kann ein Benutzer auch mehrere Rollen gleichzeitig einnehmen. iViews, die für eine Rolle zugelassen, aber nicht in einem der Workset enthalten sind, werden in sog. Channels platziert, aus denen der Benutzer sie dann bei Bedarf seinen persönlichen Worksets hinzufügen kann. Weiterhin kann ein Benutzer das Aussehen der Seiten auf seine persönlichen Bedürfnisse anpassen, indem er z.B. die Anordnung oder Farben ändert.

Der Unification-Server bietet den iViews die Möglichkeit, transparent auf Daten verschiedener Herkunft zuzugreifen. Dies ermöglicht eine Sicht auf Informationsobjekte und die Navigation zwischen Informationsobjekten, die es so in keinem System des Unternehmens gibt. Der Unification-Server modelliert hierfür die (virtuellen) Informationsobjekte als normale Objekte und merkt sich, welche (Informations-)Teile von welchem Unifier geliefert werden. Die Unifier selbst greifen spezialisiert auf die eigentliche Datenquelle zu, die z.B. ein SAP-System oder eine Oracle-Datenbank sein kann. „Die Unification-Metadaten enthalten den Kontext der Informationen aus einer Quelle und Informationen darüber, wie dieser Kontext auf Daten in einer anderen Quelle angewendet werden kann, damit der Benutzer die wichtigsten Informationen aus dieser Quelle erhält“ ([SAPP02], Seite

17). Die Unification-Technologie kann mit Views in Datenbanken verglichen werden, die ebenfalls Daten aus verschiedenen Quellen – Tabellen und ggf. andere Views – als Einheit präsentieren. Die Problematik der Änderung von Daten über Views, die bei Datenbanken dazu führt, dass solche Änderungen im Allgemeinen nur dann zugelassen werden, wenn sich die Daten nur aus genau einer Tabelle zusammensetzen, entsteht bei der Unification-Technologie nicht, da Änderungen der Daten hier schlicht nicht zulässig sind. Die von iViews angezeigten Daten können nur in den Ursprungssystemen verändert werden, wobei der Exchange-Server dafür Sorge zu tragen hat, dass redundant gespeicherte Daten zwischen verschiedenen Systemen konsistent gehalten werden. Der Exchange-Server garantiert also auch die Konsistenz der virtuellen Informationsobjekte des Unification-Servers.

Die einheitliche Sicht auf die Daten verschiedener Systeme durch den Unification-Server ist auch die Grundlage der sog. Drag and Relate-Technik. Hierbei kann ein Kontext vom Benutzer von einem iView, der Quelle, mit Hilfe der Maus auf einen anderen iView, dem Ziel, übertragen werden. Der Ziel-iView stellt dann seine Informationen im Rahmen des ihm übergebenen neuen Kontextes dar. Beispielsweise kann so von einem iView, der die List aller offenen Lieferungen in Unternehmen anzeigt, die Kundenbezeichnung zu einem anderen iView übertragen werden, der nähere Details zu einem Kunden anzeigt. Dabei muss die Kundenbezeichnung im Auftragssystem, aus der die Liste der offenen Lieferungen stammt, nicht mit der Kundenbezeichnung im Stammdatensystem, aus dem die Detailinformationen über den Kunden stammen, übereinstimmen. Nur der Unification-Server muss in der Lage sein, von einer Kundenidentität im Auftragssystem auf die Kundenidentität im Stammdatensystem zu schließen. Den iViews bleibt dies verborgen: Für sie existiert nur ein einheitliches Kundeninformationsobjekt.

Web Application Server

Im Gegensatz zum Portals-Server dient der Web Application Server zur Änderung von Daten in Transaktionssystemen und Datenbanken. Hierbei wird ein dialogorientierter Ansatz gewählt, wie er auch im SAP R/3-System zu finden ist (vgl. z.B. [Zie97]). Speziell der Ansatz der Seiten mit Ablauflogik erinnert sehr stark an das Konzept der Dynpros im R/3, erst das in neueren Versionen des Web Application Servers eingeführte MVC-Pattern löst sich etwas von dieser Vorlage.

In Abbildung 2.13 wird die Architektur des Web Applikation Servers gezeigt. Mit „R“ bezeichnete Pfeile zeigen die Bearbeitungsrichtung eines Requests. Der Internet Communication Server ist für die Kommunikation mit den Benutzern zuständig, wobei die Protokolle HTTP, HTTPS, SMTP, XML/SOAP und RFC unterstützt werden. Eine Anfrage wird entweder an die Laufzeitumgebung der Web-Dynpros oder an die Integration Engine weitergegeben. Hierbei werden ggf. Session-Informationen aus dem Request extrahiert und an die relevante Umgebung weitergegeben, damit diese den richtigen Kontext der Anfrage herstellen kann. Die Objekte der Geschäftslogik laufen innerhalb eines Applikationsservers, der derzeit (SAP) Business Obejcts unterstützt und später auch dem J2EE-Standard (Java 2 Enterprise Edition) entsprechen soll. „Der SAP Java Connector (SAPJCo) ermöglicht Methodenaufrufe zwischen Java-Anwendungen und ABAP-Anwendungen (und umgekehrt)“

([SAPW02, Seite16]). Die Datenbankschnittstelle stellt eine von der tatsächlich verwendeten Datenbank unabhängige Schnittstelle zur Verfügung, wie dies mit den Data Dictionary im SAP R/3-System ebenfalls der Fall ist. Die Integration Engine stellt Methoden von Objekten der Geschäftslogik anderen Applikationen als Web-Services bereit.

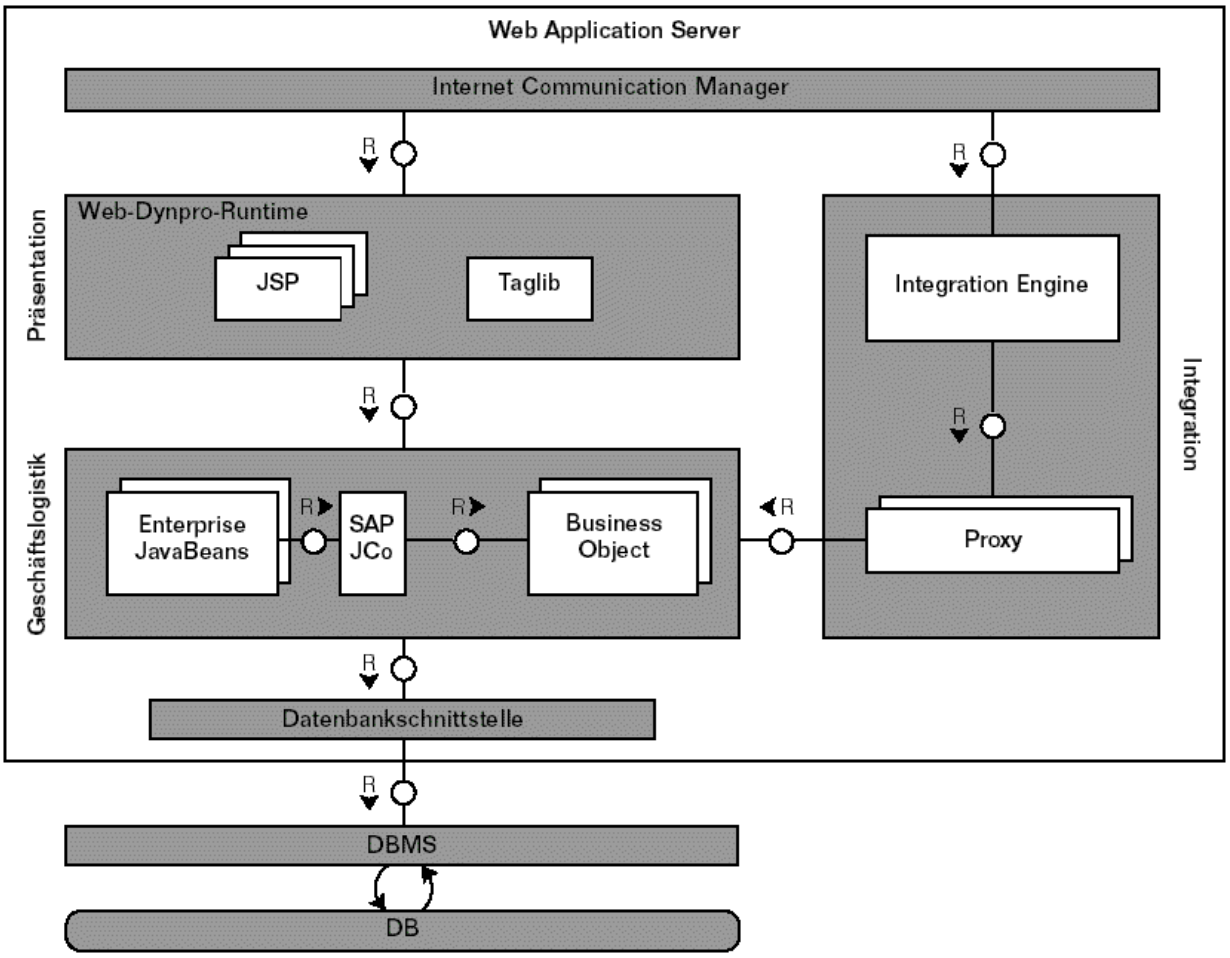

Abbildung 2.13: Architektur des Web Application Servers ([SAPW02, Seite 18]).

BSP-Applikationen Eine SAP-Web-Applikation, auch BSP-Applikation genannt, besteht sowohl aus der Benutzungsoberfläche, als auch aus der Geschäftslogik, soweit diese nicht durch externe Funktionen gekapselt ist.

Die Benutzungsoberfläche einer BSP-Applikation wird gebildet aus ([SAPH02]):

- statischen Webseiten.
- dynamisch erzeugten Webseiten, so genannten Business Server Pages (BSPs) oder Templates, die serverseitiges Scripting enthalten und erst zur Laufzeit zu einer fer-

tigen, „statischen“ Webseite expandiert werden. BSPs können entweder Seiten mit Ablauflogik oder Views sein.

- Zusätzlich können Controller existieren, wenn das Model-View-Controller Design Pattern, kurz MVC-Pattern, verwendet wird. SAP empfiehlt die Verwendung des MVC-Patterns, es war jedoch in älteren Versionen (vor Version 6.20) nicht vorgesehen.

- diversen MIME-Objekten wie Bilder, Symbole, Sound-Dateien, Stylesheets etc., die Teil einer typischen Web-Anwendung sind.

Alle diese Objekte sind als Teil der BSP-Applikation in das Korrektur- und Transportwesen des Web Applikation Servers integriert und werden als logische Einheit behandelt.

Als Besonderheit stellt SAP eine eigene Tag-Library zur Verfügung, mit deren Hilfe sich für Geschäftsanwendungen so wichtige Dinge wie Eingabehilfen, Eingabetabellen oder Tastatursteuerung relativ gut nachahmen lassen. Die Tags sind dabei unabhängig vom verwendeten Endgerät des Benutzers; erst bei der Erzeugung einer (Antwort-)Seite wird zur Laufzeit Client-seitiges Javascript o.ä. hinzugefügt.

Die Geschäftslogik kann in Form von BAPIs, Funktionsbausteinen oder Klassenbibliotheken aus einer BSP-Applikation heraus angesprochen werden. Zusätzlich hierzu bietet das BSP-Programmiermodell ein Strukturierungshilfsmittel - die BSP-Applikationsklasse - an, die zur Kapselung der in der BSP-Applikation aus der Business Logik verwendeten Funktionalität genutzt werden kann.

Abbildung 2.14 stellt die einzelnen Bestandteile einer BSP-Applikationen dar. Dabei werten Controller einen Request gemäß des ihnen zugrunde liegenden Modells aus und wählen einen passenden View, um das Ergebnis darzustellen. Die Business Server Pages (BSPs) sind (HTML-)Seiten, die statischen Code und dynamischen Code (Javascript oder ABAP) enthalten können. Nach der Auswertung des dynamischen Codes wird eine Seite als normaler HTML-Code an den Client gesendet. Eine Seite kann eine Seite mit Ablauflogik, ein View oder ein Seitenfragment sein. Bei einer Seite mit Ablauflogik kann zu den jeweiligen Zeitpunkten (Events) über die Eventhandler beliebiger Code auf dem Server ausgeführt werden. Seiten mit Ablauflogik können über eine URL angesprochen werden, wenn die Applikation nicht dem MVC-Pattern folgt. Views dienen hingegen der reinen „Befüllung“ einer BSP-Seite mit den Ergebnissen eines Controller-Requests. Ein Seitenfragment wiederum ist eine BSP-Seite, die in andere BSP-Seiten eingefügt werden kann. Dabei kann ein Seitenfragment in beliebig vielen anderen Seiten verwendet werden. In der Navigationsstruktur wird festgelegt, mit welchem Request von welcher Seite auf welche Folgeseite verzweigt wird. Die Applikationsklasse bündelt die Geschäftslogik einer Applikation an einer Stelle und bietet die Möglichkeit, Daten über mehrere Requests zwischenzuspeichern. Die MIME-Objekte, z.B. Grafiken, Style Sheets, Video- oder Audio-Daten dienen der Attraktivitätssteigerung der Web-Applikation.

Eine BSP-Applikation kann *zustandsbehaftet* (stateful) oder *zustandslos* (stateless) sein. Bei einer stateful Applikation lebt das Applikationsobjekt, eine Instanz der Applikationsklasse, für die Dauer der gesamten Session. Das Applikationsobjekt wird einmalig beim ersten Request auf die BSP-Applikation mit einer Session erzeugt, es ist ein Singleton

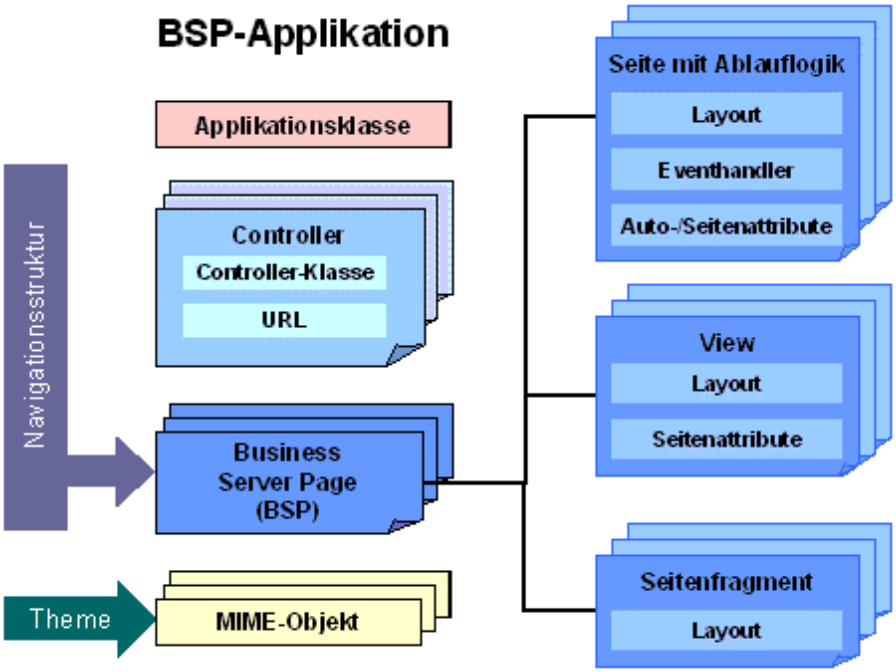

Abbildung 2.14: Bestandteile einer BSP-Applikation ([SAPH02]).

bezüglich der Session. Damit bietet sich bei stateful Applikationen die Applikationsklasse als Puffer für aufwändig zu bestimmende oder längerlebige Daten an. Bei stateless Applikationen wird das Applikationsobjekt bei jedem Seitenaufruf neu initialisiert, es eignet sich in diesem Falle also nicht zur Speicherung von Daten, sondern nur zur Kapselung der Geschäftslogik. Eine Applikationsklasse kann von mehr als einer BSP-Applikation verwendet werden, eine Applikation kann aber auch ganz auf die Verwendung eines Applikationsobjekts verzichten.

Business Server Pages Die Business Server Pages (BSPs) sind die HTML-Seiten, die die eigentliche Anwendungslogik und die Präsentationslogik beinhalten. BSPs bestehen aus folgenden Komponenten ([SAPH02]):

- Layoutverarbeitung (Scripting)
- Preview
- Typdefinitionen
- Seitenattribute
- Eventhandler

- Verwaltungsattribute

Im Rahmen der Layoutverarbeitung wird die Präsentationslogik durch Server-seitiges Scripting festgelegt. Im Preview lässt sich das Aussehen einer Seite kontrollieren, ohne einen Browser aufrufen zu müssen.

In den Typdefinitionen können lokale Datentypen festgelegt werden. Die Seitenattribute sind Variable, die sowohl für die Layoutverarbeitung, als auch für die Eventhandler sichtbar sind. Seitenattribute können vom Typ „automatisch“ sein, sie werden dann automatisch mit Werten aus der aufrufenden URL versorgt. Die Namensgleichheit von URL-Parameter und Seitenattribut ist entscheidend. Automatische Seitenattribute gibt es nur bei Seiten mit Ablauflogik.

Die Eventhandler bilden die Geschäftslogik einer BSP. Sie können Seitenattribute manipulieren und beliebigen (ABAP-)Code ausführen. Sie werden in Abhängigkeit des Zustands zu verschiedenen Zeitpunkten (Events) aufgerufen. Ist die Seite ein View, wie dies bei der Anwendung des von SAP empfohlenen MVC-Patterns der Fall ist, so werden keine Eventhandler aufgerufen, da die gesamte Geschäftslogik im Modell, also in der Applikationsklasse, realisiert wird.

Zu den Verwaltungsattributen gehören u.a. der Typ der Seite, die URL, die ggf. anzuzeigende Fehlerseite, der Zustand (stateful oder stateless), Caching-Informationen und Übertragungsoptionen.

Beispiel eines Views mit Controller Zur Verdeutlichung des BSP-Konzepts sollen nun Ausschnitte aus dem Model-View-Controller-Tutorial aus [SAPH02] wiedergegeben werden. Bei diesem Beispiel wird mit Hilfe eines Controllers und eines Views der Name des angemeldeten Benutzers angezeigt.

Zunächst wird eine Controller-Klasse definiert, die von der Klasse `CL_BSP_CONTROLLER` abgeleitet wird. Anschließend wird die Seite `view_test.htm` als View angelegt, die das Seitenattribut „name“ erhält, vgl. Abbildung 2.15.

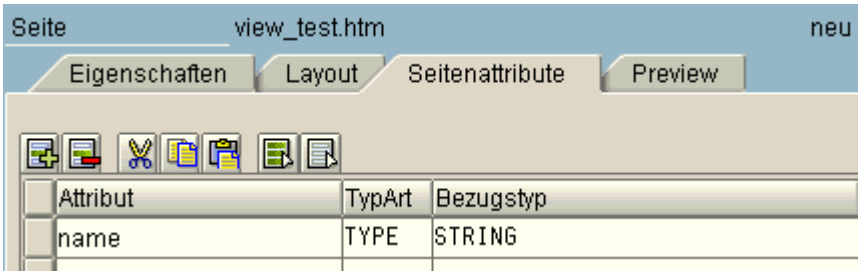

Abbildung 2.15: Definition eines Seitenattributs.

Die Layoutverarbeitung lautet wie folgt:

```
<@page language="abap">
<html>
```

```
  <head>
     <link rel="stylesheet"
           href="../../sap/public/bc/bsp/styles/sapbsp.css">
     <title> Layout zum Controller </title>
  </head>
  <body class="bspBody1">
     <h1>View Beispiel</h1>
     <h3>Hallo, Benutzer <%=name%></h3>
  </body>
</html>
```

Damit ist der View definiert. Um den View zu verwenden, wird die Methode `DO_REQUEST` des Controllers wie folgt überschrieben:

```
method DO_REQUEST.
  data: myview type ref to if_bsp_page.
  myview = create_view( view_name = 'view_test.htm' ).
  myview->set_attribute( name = 'name' value = sy-uname ).
  call_view( myview ).
endmethod.
```

In der Methode wird also zunächst die Variable `myview` definiert, die eine Referenz auf eine BSP-Seite enthalten kann. Nach der Erzeugung eines Views des Typs `view_test.htm` wird dem Seitenattribut „name“ der Wert des aktuellen Benutzers zugeordnet. Anschließend wird der View aufgerufen. Die an den Benutzer ausgelieferte Seite könnte so aussehen:

```
<html>
  <head>
     <link rel="stylesheet"
           href="../../sap/public/bc/bsp/styles/sapbsp.css">
     <title> Layout zum Controller </title>
  </head>
  <body class="bspBody1">
     <h1>View Beispiel</h1>
     <h3>Hallo, Benutzer ZIEMER</h3>
  </body>
</html>
```

Anmerkung: Wie leicht ersichtlich ist, hätte in der Methode `DO_REQUEST` auch das Applikationsobjekt als Modell aufgerufen werden können, um die Daten für die Seitenattribute des Views zu besorgen. Ebenso könnte in Abhängigkeit des Zustands des Modells zwischen verschiedenen Views ausgewählt werden. Dem Programmierer sind hier keine Grenzen gesetzt.

Navigationsstruktur Die Navigation zwischen den Seiten einer BSP-Applikation kann entweder direkt im Code oder indirekt über die Navigationsstruktur geschehen. Die Navigation findet im Allgemeinen über Methoden des Objekts `navigation` statt.

Bei der direkten Navigation wird unmittelbar die Ziel-Seite angegeben, z.B. durch die Anweisung `navigation->goto_page('error.htm')`. Hierbei wird ein Redirect durchgeführt, d.h. dass der Browser des Anwenders den Hinweis erhält, die neue Seite anzufordern. Alternativ kann in der Layoutverarbeitung das `<bsp:goto>`-Tag verwendet werden, hierbei wird das Ergebnis der aufgerufenen Seite als Ergebnis des ursprünglichen Requests zurückgegeben. Es findet im zweiten Falle also kein Redirect statt.

Bei der indirekten Navigation wird das Ziel der Navigation anhand einer Tabelle bestimmt, die zu jeder Startseite den Namen des Navigationsrequests und die Zielseite kennt. Abbildung 2.16 zeigt ein Beispiel aus [SAPH02]. Die indirekte Navigation geschieht durch die Methode `navigation->next_page`, auf der Seite `default.htm` z.B. durch `navigation->next_page('TOAUTHORS')` und führt auf die Seite `authors.htm`.

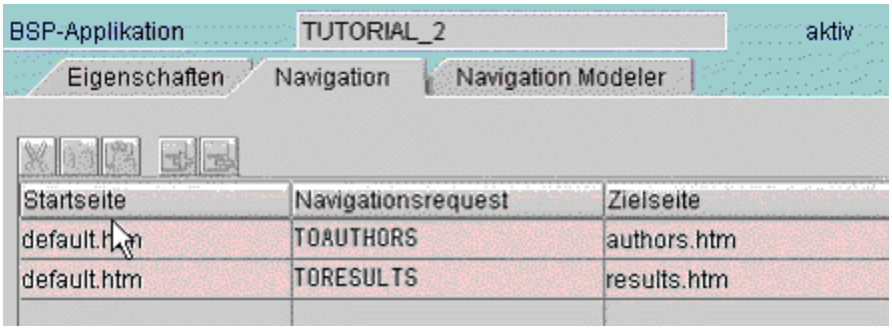

Abbildung 2.16: Navigationsstruktur einer BSP-Applikation.

Für die Bearbeitung der Navigationsstruktur stellt SAP einen einfachen grafischen Editor zur Verfügung, so dass die Seitenfolge theoretisch auch ohne Programmierkenntnisse verändert werden könnte.[5]

Bewertung

Die mySAP-Technologie erfüllt die in Kapitel 2.2.2 (Seite 19) benannten Anforderungen bei der Gestaltung eines Portals teilweise:

Endbenutzertauglichkeit Hier muss zwischen der Portals-Komonente und dem Web Application Server unterschieden werden. Während mySAP Portals durch das Konzept der Rollen und Worksets ein gewisses Modell der angebotenen Dienste gegenüber dem Benutzer beinhaltet, liegt die Präsentation eines Dienstes über den Web Application Server völlig in der Hand des Programmierers. Die mySAP-Technologie erfüllt

[5] Dieser Editor scheint den Versionen 6.20 und 6.30 nicht mehr vorhanden zu sein.

also nur bei der Portals-Komponente teilweise die Anforderung der Endbenutzertauglichkeit.

Abteilungsleitertauglichkeit Wie bei der Endbenutzertauglichkeit bietet die Portals-Kompontente mit den Rollen und Worksets wenigstens ansatzweise eine Lösung, der Web Application Server ist auch hier auf reine Programmierung beschränkt.

Entwicklertauglichkeit Die Portals-Komponente erfüllt nur sehr bedingt die Anforderungen von Entwicklern. Sie bietet zwar eine Abstraktion über die Datenquellen und damit gute Voraussetzungen für die Ansicht von Daten, eine Veränderung von Daten ist aber nicht möglich. Der SAP Web Application Server ist hingegen völlig auf Programmierer zugeschnitten, auch wenn die Wiederverwendbarkeit von entwickelten Komponenten im Einzelfall sehr komplex und undurchsichtig sein kann. Eine höhere Abstraktion über Dienste bietet der Web Application Server nur auf der Ebene der Web Applikationen selbst.

Web-Designertauglichkeit Weder die Portals-Komponente noch der Web Application Server erfüllen die Anforderung der Web-Designertauglichkeit. Bei den Portals fehlt die Kontrolle über die MiniApps und letztlich auch die iViews, wenn diese entsprechend implementiert werden. Beim Web Application Server kann zwar zwischen verschiedenen vorgegebenen Designs für die HTMLB-Tag-Bibliothek ausgewählt werden, ein Konzept für generelle Vorlagen oberhalb der Tag-Ebene fehlt jedoch.

Kommunikationstauglichkeit Die Vielzahl von Konzepten, die sich z.T. überlappen und z.T. auch gegensätzlich sind, ist der Kommunikation zwischen den beteiligten Gruppen eindeutig abträglich und nicht förderlich.

Arbeitserleichterung Die einzelnen Konzepte und Vorgehensmodelle können für die jeweiligen Gruppen als Arbeitserleichterung gesehen werden. Diese Arbeitserleichterungen nivellieren sich etwas durch die große Anzahl von Konzepten und Modellen.

Als drittes und letztes System für Portale und Dienste soll nun der infoAsset Broker der Firma infoAsset AG[6] untersucht werden, der die Problematik der Portale und Dienste durch eine konsequente Schichtenarchitektur zu lösen versucht.

2.2.6 infoAsset Broker

Der infoAsset Broker ist eine Portalsoftware für das Wissensmanagement in Unternehmen. Das Wissensmanagement wird durch sog. *information assets* realisiert, die es in verschiedenen Ausprägungen gibt und die in vielfältiger Weise miteinander in Beziehung stehen können ([inf01]).

In ([Weg02, Seite 193]) werden vier implementierte Typen von Assets genannt, die wesentlich für das Wissensmanagement sind:

[6] www.infoasset.de, Stand November 2002

- Personen
- Dinge
- Orte
- Begriffe

Die Idee dabei ist, dass Personen, die Träger des impliziten Wissens, Dinge, das explizite Wissen, an Orten ablegen und dass sowohl Personen wie auch Dinge über eine Taxonomie (Begriffe) miteinander in Beziehung gesetzt werden.

Die Architektur des infoAsset Brokers wird in Abbildung 2.17 dargestellt. Sie umfasst die sechs Schichten Presentation, Communication, Interaction, Services, Store und Persistence/Integration. Auf der Ebene der Presentation wird entweder ein Web-Browser oder ein Mail-Client verwendet, kurzzeitig wurde auch WAP als Protokoll unterstützt. Die Interaktionsebene besteht aus Templates, Handlern und Session-Informationen. Templates sind HTML-Seiten, die durch eine einfache, Server-seitige Scriptsprache um dynamische Aspekte erweitert werden können. Handler stoßen die Geschäftslogik in der Diensteschicht an und versorgen die dynamischen Aspekte der HTML-Seite mit den notwendigen Daten. Die Services-Ebene stellt die Geschäftsobjekte zur Verfügung; die Abstraktion über die konkrete Form der Datenhaltung geschieht auf der Ebene der Stores. Im Sinne des MVC-Patterns entsprechen die Handler den Controllern, die Templates den Views und die Geschäftsobjekte dem Model. In der Persistence/Integration-Ebene werden verschiedene Datenbanken, Content-Management-Systeme und sonstige Systeme angebunden ([MaSt02]).

Im Folgenden werden Dienste, Handler und Templates diskutiert und an einem Beispiel illustriert, da hier alle für Portale und Dienste wesentlichen Elemente angesiedelt sind. Beschreibungen weiterer Eigenschaften des infoAsset Brokers, wie z.B. Synchronisation zwischen verschiedenen Instanzen oder Anwendung in konkreten Projekten finden sich u.a. in [MaLe02], [MaLe02a], [RaMue+02] und [Weg02].

Dienste

Die Dienste des infoAsset Brokers werden mit Hilfe von (Java-)Interfaces beschrieben und durch entsprechende Klassen implementiert. Dabei bleibt die Implementierung eines Dienstes für darüber liegende Schichten unsichtbar, lediglich die Schnittstelle ist bekannt.

Dienste, die persistente Objekte verwalten, teilen sich zumeist in zwei Dienste auf: Zum Einen in den Dienst, der die eigentlichen Objekte darstellt und zum Anderen in den Dienst, der die Menge der Objekte verwaltet, auch Container genannt. Ein typisches Beispiel ist das Asset und der dazugehörige AssetContainer. Aus diesem Grundtyp leiten sich auch die anderen Asset-Typen wie z.B. Person oder Document ab.

Das Interface der Assets lautet wie folgt:

```
public interface Asset
{
  /**
```

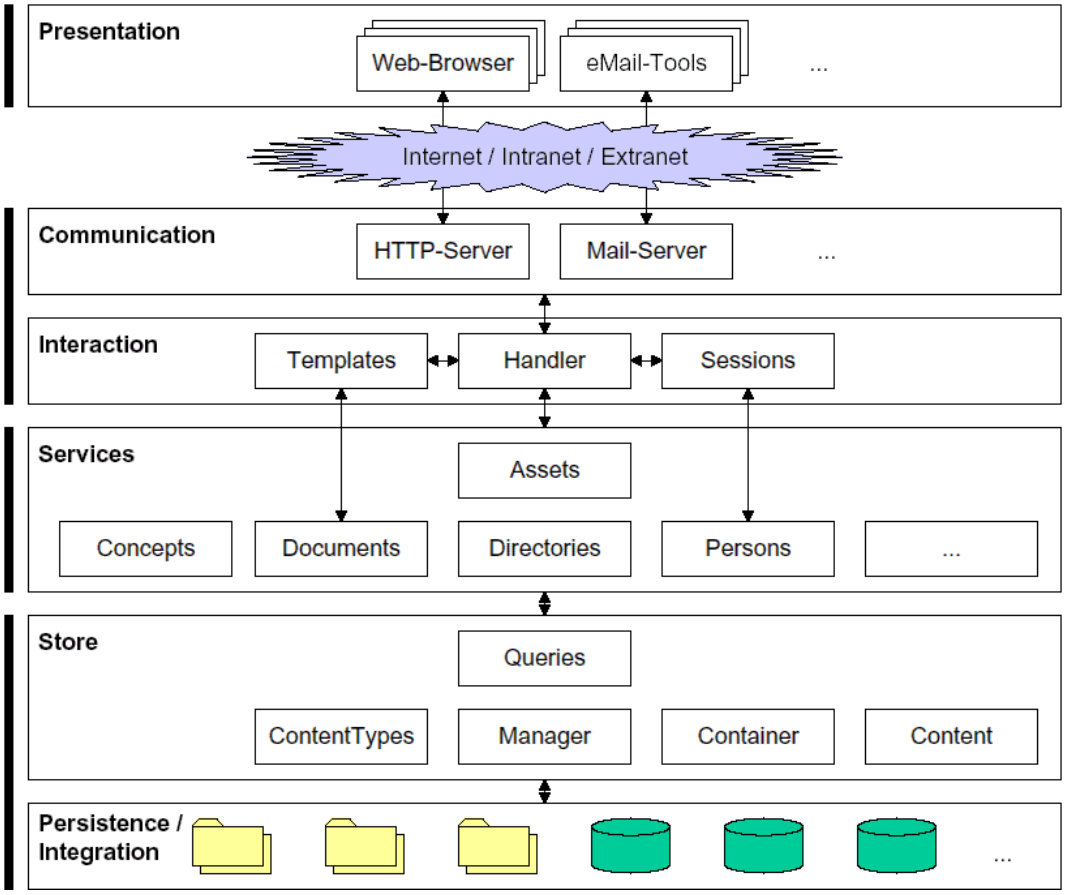

Abbildung 2.17: Architektur des infoAsset Brokers ([Weg02, Seite 142]).

```
 * Returns the unchangable asset id of this asset.
 */
public String getId();

/**
 * Return the name of this asset.
 */
public String getName();

/**
 * Change the name of this asset.
 */
public void setName(String name);

/**
```

```
   * Return the date of the last modification of this asset.
   */
  public Date getLastModification();

  /**
   * Make all changes to this asset persistent.
   */
  public boolean write();

} // Asset
```

Assets haben also einen eindeutigen Identifier, einen Namen und einen Zeitstempel, wann die letzte Änderung durchgeführt wurde. Assets können zusätzlich abgespeichert werden.

Der dazugehörige Container für Assets bietet folgende Funktionen:

```
public interface AssetContainer
{

  /**
   * Return the name (or display title) of a given asset id
   */
  public String getName(String assetId);

  /**
   * Return true if an asset with the given asset id exists.
   */
  public boolean exists(String assetId);

  /**
   * Execute a Search and return an iterator returning SearchResults.
   */
  public Iterator search(Search search) throws SearchException;

  /**
   * Return the supported predicate names.
   */
  public Iterator supportedPredicateNames(String kind);

  /**
   * Return the supported ranking names of this asset container.
   */
  public Iterator supportedRankingNames(String kind);
```

```
} // interface AssetContainer
```

Konkrete Dienste erweitern diese Schnittstellen dann für ihre spezifischen Bedürfnisse. Beispielsweise definiert die Schnittstelle für Personen, dass Personen einen Vornamen haben können und der entsprechende Container für Personen bietet Suchmöglichkeiten nach Namen und Vornamen an.

Templates

Templates sind (zumeist) HTML-Seiten, die mit speziellen Befehlen versehen sind. Die Templates bilden die Benutzungsoberfläche des infoAsset Brokers.

In der Scriptsprache des infoAsset Brokers gibt es nur drei Elemente ([Weg02, Seiten 177-179]):

- Skalare Platzhalter: Diese Platzhalter können überall und beliebig oft in einem Template verwendet werden. Jeder Platzhalter beginnt und endet mit dem Dollarzeichen `$` und darf lediglich Buchstaben und Ziffern enthalten. Ein Beispiel wäre `$name$` . Ein Platzhalter wird vor der Auslieferung durch ein beliebiges Textfragment ersetzt.

- Bedingte Platzhalter: Bedingte Platzhalter bestehen im Unterschied zu skalaren Platzhaltern jeweils aus einem Paar einander zugeordneter Platzhaltern, die den bedingten Bereich einschließen. Um einen bedingten Platzhalter von einem skalaren Platzhalter unterscheiden zu können, beginnt der erste Platzhalter des Paares mit `$[`, während der zweite Platzhalter des Paares mit `]$` endet. Ein bedingter Platzhalter könnte also z.B. `$[istAdmin$...$istAdmin]$` lauten. Zu Laufzeit würde der enthaltene Bereich nur dann ausgeliefert, wenn dem Platzhalter `istAdmin` der Wert „true“ zugeordnet wurde.

  Eine Variation des bedingten Platzhalters sieht auch die Negation der Bedingung vor. Dazu wird ein dritter Platzhalter zwischen dem ersten und dem zweiten Platzhalter eingefügt, der mit einem `$]` eingeleitet und mit einem `[$` beendet wird. Ein solcher Platzhalter könnte z.B. wie folgt lauten:

  ```
  $[istAdmin$
    ...
  $]istAdmin[$
    Sie sind nicht Administrator.
  $istAdmin]$
  ```

  Diese Variation entspricht dem klassischen IF...THEN...ELSE in Programmiersprachen.

- Listen-Platzhalter: Listen-Platzhalter kennzeichnen Bereiche, die wiederholt in Form einer Liste ausgegeben werden sollen. Ähnlich wie die bedingten Platzhalter, umschließen sie den jeweiligen Bereich durch ein Paar von zusammengehörigen Platzhaltern mit bestimmter Syntax. Da der wiederholte Bereich natürlich nicht immer

identisch repliziert werden soll, muss er weitere Platzhalter enthalten, die in jeder Wiederholung (Iteration) mit neuen Werten gefüllt werden. Die enthaltenen Platzhalter sind dabei so an den Listen-Platzhalter gebunden, dass sie bei jeder Iteration neue Werte liefern.

Syntaktisch wird die Bindung durch Qualifikation der Namen ausgedrückt: Jeder eröffnende Listen-Platzhalter führt einen neuen Namensbereich ein, indem er einen Präfix angibt, z.B. durch `$[persons p$`. Dies unterscheidet die Syntax zugleich von der der bedingten Platzhalter. Alle in der Liste enthaltene Platzhalter, die an der Iteration teilnehmen, müssen mit dem Präfix qualifiziert werden, also z.B. durch `$p.name$` statt `$name$`. Da alle Konstrukte orthogonal zueinander verwendet werden können, also beispielsweise Listen in Listen geschachtelt werden können, ist die Qualifikation erforderlich, um die Bindung an die korrekte Liste ausdrücken zu können.

Folgendes Beispiel definiert eine Liste mit den Namen von Personen, die durch Komma getrennt werden, falls es mehr als eine Person in der Liste gibt:

```
$[persons p$
  Personen in der Liste: $p.vorname$  $p.name$, $p.vorname$ $p.nachname$
$]persons[$
  Es sind keine Personen in der Liste.
$persons]$
```

Gibt es keine Person in der Liste, so wird der entsprechende Text ausgegeben.

Wie bereits erwähnt, lassen sich alle drei Elemente beliebig miteinander kombinieren. Dies ermöglicht auch ausgefeilte Layouts und Personalisierung durch Aus- bzw. Einblendungen von HTML-Code.

Folgendes Beispiel soll die Verwendung der Elemente verdeutlichen. Das Template zeigt einen kurzen Text aus der Online-Hilfe an:

```
<html>
<head></head>
<body onLoad="executeLoaded()" onUnload="executeExit()">
  $[hasParent$
    <p>
      <a href="javascript:s.show('helpTopic/$parentId$.htm')">
        <b>$parentTitle$ </b>
      </a>
    </p>
  $hasParent]$

  <p>Der infoAsset Broker beinhaltet eine Reihe von Community
     -Funktionen.</p>
```

```
  $[children c$ <br>
    <table border="0" cellspacing="2" cellpadding="2">
      <tr align="left">
        <td valign="middle">
          <img src="/de/skin/images/bullet.gif">
        </td>
        <td valign="middle">
           <a href="javascript:s.show('helpTopic/$c.id$')">
             $c.title$
           </a>
        </td>
      </tr>
      <tr align="left">
        <td valign="middle">
          <img src="/de/skin/images/bullet.gif">
        </td>
        <td valign="middle">
           <a href="javascript:s.show('helpTopic/$c.id$')">
             $c.title$
           </a>
        </td>
      </tr>
    </table>
  $children]$
</body>
</html>
```

Instantiiert könnte ein Browser folgenden HTML-Code erhalten:

```
<html>
<head></head>
<body onLoad="executeLoaded()" onUnload="executeExit()">

  <p>Der infoAsset Broker beinhaltet eine Reihe von Community
     -Funktionen.</p>

  <br>
    <table border="0" cellspacing="2" cellpadding="2">
      <tr align="left">
        <td valign="middle">
          <img src="/de/skin/images/bullet.gif">
        </td>
        <td valign="middle">
```

```
          <a href="javascript:s.show('helpTopic/forum')">
            Forum
          </a>
        </td>
      </tr>
    </table>
</body>
</html>
```

Wie zu sehen ist, wurde der HTML-Code im bedingte Platzhalter `hasParent` nicht mit ausgeliefert. Der Listen-Platzhalter `children` mit dem Namensraum `c` ist eine Iteration durchlaufen, in der der skalare Platzhalter `id` durch den Text `Forum` ersetzt wurde. Die Javascript-Methode `s.show` ist eine Broker-spezifische Methode, die die aufzurufende URL, inklusive Session-ID, erstellt.

Handler

Handler sind das Bindeglied zwischen der Geschäftslogik in den Diensten einerseits und der Präsentation in Form der Templates andererseits. Sie entsprechen den Controllern im MVC-Pattern. Es werden sichtbare und unsichtbare Handler unterschieden. Sichtbare Handler produzieren eine Ausgabe, indem sie ein Template füllen und zurückgeben. Unsichtbare Handler rufen Methoden der Geschäftslogik auf, z.B. nehmen sie Änderungen an einem Objekt vor, und leiten dann die Kontrolle dann an einen anderen Handler weiter. Der Handler, an den weitergeleitet wird, kann ein sichtbarer oder ein unsichtbarer Handler sein. Die Weiterleitung geschieht nur Server-seitig, d.h. dass keine Aufforderung (Redirect) an den Browser des Benutzers gesendet wird, eine neue Seite anzufordern.

Alle Handler erben von der Klasse `GenericHandler`, die viele Methoden für das Füllen eines Templates bereitstellt. Ein Handler muss dann lediglich die Methode `handleRequest` überschreiben, um die konkreten Werte für die Platzhalter im Template zu bestimmen oder Geschäftslogik aufzurufen.

Ein typischer, sichtbarer Handler hat folgenden Aufbau:

```
public class VisibleHandler extends GenericHandler {
  public void handleRequest(Session session,
                            OutputStream output,
                            String documentId)
  {
    // get some values
    final String parentTitle = ...
    final HelpTopic helpTopic = ...
    ...
    // get template
    Template template = new Template(documentId);
    // fill template
```

```
    template.put("name", new PrintSubstitution() {
      public void print(Session session, PrintStream p) {
        p.print(parentTitle);
    }});
    template.put("hasParent", new ConditionalTemplate(){
      public boolean test(Session session) {
        return !helpTopic.isMostGeneralTopic();
      }
    });
    ...
    // instantiate template
    template.instantiate(session, output);
  }
}
```

Zunächst werden die Werte ermittelt, die später den Platzhaltern zugewiesen werden sollen. Sind alle Werte bekannt, so wird das Template, besser gesagt eine Objekt-Repräsentation des Templates, beschafft und über die `put`-Methode mit den benötigten Werden versorgt. Schließlich wird das Template instantiiert, d.h., dass die Ersetzung der Platzhalter durch die zugewiesenen Werte vorgenommen wird. Der `session`-Parameter wird vom System automatisch zur Verfügung gestellt, eine Session enthält z.B. Informationen über den aktuell eingeloggten Benutzer oder beim Aufruf in der URL mit übergebene Parameter.

Unsichtbare Handler folgen meistens diesem Aufbau:

```
public class InVisibleHandler extends GenericHandler {
  public void handleRequest(Session session,
                            OutputStream output,
                            String documentId)
  {
    // call business logic
    Person p = session.getServices().getPersons().getPerson(...);
    p.setName(session.getRequestParameter("name");
    ...
    boolean success = p.write();
    // navigate to next page
    if (success) {
      session.forwardRequest(output, "persons/show");
    } else {
      session.forwardRequest(output, "persons/error");
    }
  }
}
```

Unsichtbare Handler rufen zunächst Geschäftslogik auf, indem sie sich die entsprechenden Objekte und Methoden aus der Diensteschicht des infoAsset Brokers ansprechen.

Anschließend wird der Request, und damit die Kontrolle, an einen anderen Handler mit Hilfe der Methode `forwardRequest` weitergeleitet.

Um zu bestimmen, welcher Handler welchen Request bearbeitet, wird eine Indirektion ähnlich der Navigationsstruktur im SAP Web Applikation Server verwendet. In der Konfigurationsdatei `handler.txt` wird einem Request ein Handler zugeordnet, der für diesen Request zuständig ist. Der Eintrag

`persons/edit=de.infoasset.broker.handler.persons.EditHandler`

legt z.B. fest, dass Requests der Form

`http://<brokerserver>:<port>/de/persons/edit.htm?id=4711`

an den Handler `de.infoasset.broker.handler.persons.EditHandler` gesendet und von diesem abgearbeitet werden. Das Präfix `/de` in der URL bestimmt lediglich die vom Anwender gewünschte Sprache der Benutzungsoberfläche und der multilingualen Werte.

Bewertung

Der infoAsset Broker erfüllt die in Kapitel 2.2.2 (Seite 19) benannten Anforderungen bei der Gestaltung eines Portals teilweise:

Endbenutzertauglichkeit Der infoAsset Broker bietet kein Konzept oder Modell, dem Benutzer ein erstelltes Portal zu kommunizieren. Der Programmierer ist hierbei auf sich gestellt.

Abteilungsleitertauglichkeit Das Gruppenkonzept als Grundlage des Berechtigungskonzepts ist zwar abteilungsleitertauglich, die Umsetztung geschieht jedoch mit Hilfe bedingter Platzhalter, also abteilungsleiteruntauglich auf der Ebene der Implementierung.

Entwicklertauglichkeit Die Abstraktionsschichten für die Datenhaltung und die Geschäftslogik ermöglichn eine relativ problemlose Wiederverwendung bestehender Geschäftslogik und die Erstellung neuer Portale auf Basis bestehender Dienste. Die Handler, also die Komponente, die die Verbindung von Oberfläche und Geschäftslogik herstellt, müssen jedoch für jedes Portal im Wesentlichen neu erstellt werden. Eine Wiederverwendung von Handlern mit anderen Templates ist nur dann möglich, wenn sich die Templates (sehr) wenig ändern.

Web-Designertauglichkeit Durch die einfache Script-Sprache in den Templates können für die Seitengestaltung auch grafische Tools, wie z.B. der Makromedia Dreamweaver, verwendet werden. Die Web-Designertauglichkeit kann also unterstellt werden.

Kommunikationstauglichkeit Die Templates des infoAsset Brokers können für ein *Rapid Prototyping* verwendet werden, was die Kommunikation zwischen den Gruppen sicherlich fördert. Ein abstrakteres Modell, das auch etwas über den Inhalt der Dienste oder das Berechtigungskonzept aussagen wüde, existiert jedoch nicht.

Arbeitserleichterung Das der infoAsset Broker keine Tools zur Generierung enthält, müssen, bis auf die Seitengestaltung, alle Schritte bei der Umsetzung eines Portals von Hand vorgenommen werden. Der infoAsset Broker erfüllt die Anforderung der Arbeitserleichterung nicht.

2.3 Zusammenfassung

In diesem Kapitel wurde zunächst die Bedeutung der betrieblichen Informationssysteme als motivierender Faktor für die Entwicklung von Portalen erörtert. Wesentlich waren dabei die flexible Zusammensetzung als Resultat veränderter Geschäftsprozesse und die Integration über verschiedene betriebliche Informationssysteme hinweg.

Es wurden für die Begriffe „Portal“ und „Web-Applikation“ Definitionen aus der Literatur zitiert und kritisch beleuchtet. Die dabei auftauchenden Widersprüche und Probleme führten zu der Definition der Begriffe (Unternehmens-)Portal und Dienst. Ein Portal wurde als Softwaresystem definiert, das Endanwendern (Benutzern) Dienste über eine Web-Oberfläche zur Verfügung stellt. Ein Dienst ist jede in Software geschriebene Funktion, die sich von anderen Systemen aus aufrufen lässt.

Anschließend wurden Anforderungen an Entwicklungsmethoden und Portalsysteme bei der Gestaltung von Portalen aufgestellt und verschiedene Entwicklungsmethoden und -Systeme kategoriesiert. Exemplarisch drei verschiedene Ansätze für Portale und Dienste vorgestellt, die jeweils einen eigenen Ansatz verfolgen. Die Sprache WebML beschreibt ein Portal und die angebotenen Dienste mit Hilfe orthogonaler Modelle für einzelne Aspekte, wie z.B. dem Datenmodell, dem Seitenmodell oder dem Navigationsmodell. Die Kombination der einzelnen Modelle ergibt dann das eigentliche Portal. Der Ansatz der SAP mit der mySAP-Technologie beruht auf dem Einsatz spezieller Server für die Aspekte der (virtuellen) Integration von Daten für lesende Zugriffe, der Konsistenz redundant gehaltener Daten in unterschiedlichen Datenquellen und der Änderung von Daten über eine Web-Oberfläche. Der infoAsset Broker verfolgt dem gegenüber einen Schichten-Ansatz, der speziell auf das Wissensmanagement ausgerichtet ist und bei dem die Integration von Daten auf einer speziellen Schicht stattfindet.

Keines der aufgeführten Systeme und Modelle für Portale und Dienste berücksichtigen mehr als nur Teilaspekte der oben aufgeführten Probleme. Portalmodelle, die einfach, ausdrucksstark und integrativ sind, fehlen, obwohl schon länger solche Modelle gefordert werden ([TrDe00]). Ein konzeptionelles Schema für ein Portal muss weit mehr sein, als eine reine Dokumentation für Endanwender. Damit die übrigen im Entwicklungsprozess beteiligten Gruppen die Erstellung und Pflege eines Portal-Modells nicht nur als zusätzliche Arbeit betrachten, müssen auch diese Vorteile durch die Modellierung erhalten. Bei einem derartigen Modell für Portale und Dienste muss es sich selbstverständlich um *ein gemeinsames Modell* handeln und nicht um ein Modell für jede einzelne Gruppe.

Im nächsten Kapitel wird aus den Diensten und Personalisierungmöglichkeiten in Portalen die grundsätzlichen Bausteine eines Portals und ihr Zusammenspiel in Form einer Referenzarchitektur abgeleitet.

Kapitel 3

Bausteine eines Portals

Nachdem im vorigen Kapitel drei konkrete, verschiedene Ansätze zur Lösung der Problematik der Dienste und Portale aufgezeigt wurden, werden im Folgenden die allgemeinen Eigenschaften von Diensten in Unternehmensportalen sowie die grundsätzliche Personalisierung von Diensten und Portalen beschrieben, um anschließend eine allgemeine Referenzarchitektur, die die Bausteine eines Portals und ihr Zusammenspiel umfasst, angeben zu können.

3.1 Dienste in Unternehmensportalen

In Unternehmensportalen lassen sich zwei Arten von Diensten unterscheiden: Benutzerorientierte Dienste und technische Dienste.

Benutzerorientierte Dienste besitzen eine für den Benutzer sichtbare Repräsentation. Typische Beispiele für benutzerorientierte Dienste sind u. a. die Anzeige von Informationsobjekten, Dialoge für die Bearbeitung von Informationsobjekten, Börsenticker, eine eingebundene Textverarbeitung oder auch die Möglichkeit der individuellen Anpassung des Layouts.

Ein benutzerorientierter Dienst kann entweder dialogorientiert oder direkt manipulierend sein. Bei einem dialogorientierten Dienst erhält der Benutzer eine Art Datenmaske, in der er Daten eintragen oder verändern kann. Hat der Benutzer die Eingabe beendet, so werden die Daten vom System, typischerweise einem entfernten Server, ausgewertet. Diese Form der benutzerorientierten Dienste findet sich vor allem im Umfeld von Datenbankanwendungen, wie es die klassischen Informationssysteme in Unternehmen darstellen.

Die Idee der direkten Manipulation wurde von Schneidermann 1983 ([Sch83]) erstmals wissenschaftlich beschrieben. Bei der direkten Manipulation verändert der Benutzer unmittelbar das gewünschte Objekt, ohne eine Eingabemaske für Daten zu verwenden. Beispiele für die direkte Manipulation sind u.a. grafische Dateimanager, moderne Textverarbeitungen oder auch die iViews des SAP Portals Server, auch wenn bei iViews lediglich die Darstellung der Daten direkt verändert werden kann.

Web-Browser sind aufgrund ihrer historischen Entwicklung als Anzeigewerkzeug für ver-

knüpfte Texte vor allem für dialogorientierte Dienste geeignet, auch wenn es erste Ansätze für die Integration direkt manipulierender Dienste gibt ([Re02]). Aber selbst für dialogorientierte Dienste sind derzeitige Web-Browser nicht optimal, wie die Bemühungen der SAP mit der Tag-Bibliothek der Web-Dynpros zeigt. Problematisch bleibt vor allem der fehlende Komfort für die Steuerung von Dialogen über die Tastatur und die Anzeige einer Auswahl von zulässigen Eingabewerten für bestimmte Felder, wie dies SAP mit der „F4-Hilfe“ (vgl. [Zie97, Seite 22]) im R/3-System aufwändig realisiert hat.

Die technischen Dienste in Portalen bleiben für den Benutzer unsichtbar. Unter technischen Diensten sind all jene Dienste zu verstehen, die Funktionalität im Portal anbieten, dies aber für den Benutzer auf transparente Art und Weise tun, also keine Repräsentation für den Benutzer besitzen. Ein typisches Beispiel für einen technischen Dienst ist der Unification-Server als Teil des SAP Portals Servers oder auch die Store-Ebene des infoAsset Brokers. Technische Dienste werden von den benutzerorientierten Diensten verwendet, um (evtl. zusätzliche) Funktionalität bereitzustellen oder um überhaupt gewissen Funktionen des Portals zu ermöglichen.

Nach diesem kurzen Überblick über die verschiedenen Arten von Diensten in Unternehmensportalen, soll nun die Personalisierung von Diensten und Portalen diskutiert werden.

3.2 Personalisierung in Unternehmensportalen

Das Verb „personalisieren“ definiert ([Du91, Seite 542]) als „auf eine Person beziehen oder ausrichten“. Im Sinne der Informationstechnologie bedeutet Personalisierung die Anpassung der Benutzungsoberfläche und des Verhaltens eines Systems in Abhängigkeit des aktuellen Benutzers.

Der Begriff der Personalisierung wird im Umfeld von Portalen und Diensten sehr häufig und oft aus verschiedenen Blickwinkeln heraus gebraucht. Im Folgenden sollen die Gründe hierfür beleuchtet und die Voraussetzungen und Objekte der Personalisierung in Unternehmensportalen beschrieben werden. Die Darstellung lehnt sich an die Darstellung in [Jac02, Seiten 5-36] an.

3.2.1 Ziele der Personalisierung

Es gibt vier Ziele, die mit Personalisierung verfolgt werden ([Jac02, Seiten 6-9]):

- Marketing
- Umsetzen von Zugriffsrechten
- Bedienungsfreundlichkeit
- Kooperationsunterstützung

Unter Marketing sind all jene Aktivitäten eines Unternehmens zu verstehen, die der Kundenbindung oder der Kundengewinnung dienen. Personalisierung zum Zwecken des

Marketings findet man vor allem in Szenarien des eCommerce, wo über ein Portal Produkte oder Dienstleistungen entgeltlich angeboten werden. Typische Beispiele sind etwa die automatischen Vorschläge ähnlicher Produkte wie jenes Produkt, das man gerade betrachtet oder kauft oder auch Hinweise auf der Einstiegseite, die über Neuigkeiten oder besondere Angebote informieren.

Die Umsetzung von Zugriffsrechten ist immer dann wichtig, wenn Daten oder Funktionen für unberechtigtem Zugriff geschützt werden sollen. Die Identifizierung des Benutzers geschieht meistens über die Eingabe der Benutzerkennung und eines Passworts. Zu dem Profil des Benutzers gehören nicht nur die persönlichen Einstellungen, die ein Benutzer vorgenommen hat, sondern z.B. auch seine Rollen, die bestimmen, welche Daten oder Funktionen für den Benutzer zugreifbar sein sollen. Das System personalisiert sich dann so für den Benutzer, dass ihm nur erlaubte Daten und Funktionen angeboten werden. Beispiele für die Umsetzung von Zugriffsrechten sind der bedingte Platzhalter `isAdmin` in Kapitel 2.2.6 im infoAsset Broker oder die Zuordnung von iViews zu Rollen und Channels im SAP Portals Server, wie in Kapitel 2.2.5 beschrieben.

Die Personalisierung eines Systems kann zur Steigerung der Bedienungsfreundlichkeit beitragen. Individuelle Anpassungen der Benutzungsoberfläche, automatische Anpassungen des Inhalts an den Wissensstand bei intelligenten, tutoriellen Systemen oder auch individuelle Sammelmappen sind Beispiele, wie Personalisierung die Bedienungsfreundlichkeit eines Systems erhöhen kann.

Kooperationsunterstützung durch Personalisierung erscheint im ersten Moment ein Widerspruch in sich zu sein, da die Kooperationsunterstützung viele Benutzer voraussetzt, die Personalisierung aber auf den Einzelnen gerichtet ist. Die Personalisierung kann jedoch wertvolle Hinweise auf die sog. *Communities of Practice* ([HuHo95]) liefern. Communities of Practice werden (virtuelle) Gruppen genannt, deren Mitglieder ähnliche Interessen oder Aufgabengebiete haben, die aber nicht unbedingt auch organisatorisch eine Gruppe bilden müssen. Ähneln sich nun die Personalisierungen zweier Benutzer unter bestimmten Aspekten, so kann es sich um zwei Benutzer mit ähnlichen Interessen oder Aufgabengebieten handeln – mithin also um zwei Mitglieder einer Community, worauf das System dann die Benutzer auch hinweisen kann.

3.2.2 Der Prozess der Personalisierung

Das Konzept der Personalisierung soll dem richtigen Anwender zur richtigen Zeit die richtigen Inhalte in einer auf die Bedürfnisse des Anwenders abgestimmten Art und Weise präsentieren. Der Prozess der Personalisierung setzt sich hierbei aus den drei Schritten Tracking, Profiling und Matching zusammen.

Im Prozessschritt des Trackings werden Benutzeraktionen und -eingaben protokolliert und für das Profiling gesichert.

Im Schritt Profiling werden die durch das Tracking gewonnenen Daten bereinigt und geordnet, so dass sie für das anschließende Matching zur Verfügung stehen. Jeder Benutzer erhält auf diese Weise ein Profil für die Personalisierung. Oftmals wird auch eine Segmentierung der Profile durchgeführt, bei der nach bestimmten Kriterien gleichartige Profile

zu Gruppen zusammengefasst werden. Die Schritte Tracking und Profiling werden oftmals vereinfachend unter dem Begriff Profiling zusammengefasst.

Durch das Matching werden anhand von festgelegten Personalisierungsmodellen, auch Business- oder Anpassungsregeln genannt, individuelle personalisierte Web-Oberflächen generiert. Die Personalisierungsmodelle definieren, welcher Inhalt anhand welcher Eigenschaften im Profil angezeigt werden soll.

Nachdem der Personalisierungsprozess durchlaufen wurde, wird dem Anwender die angepasste Oberfläche, im Falle der Dienst und Portale eine Web-Seite, präsentiert. Die erneute Protokollierung des Anwenderverhaltens in der anschließenden Interaktion mit dem personalisierten System kann wiederum Anlass zu neuen Personalisierungen geben. Es entsteht ein Personalisierungskreislauf, wobei die Qualität der Daten immer weiter steigt. Diese Tatsache wird gelegentlich auch als Learning Relationship oder Feed-Back-Loop Profiling bezeichnet.

3.2.3 Profiling

Die Grundlage der Personalisierung sind die zu einem Benutzer bekannten Daten, das Profil. Der Erfolg der Personalisierung hängt entscheidend von der Aussagekraft des Profils ab.

Um an die Daten zu gelangen kann eine aktive Strategie, auch direktes Verfahren genannt, angewandt werden oder eine passive Strategie, auch indirektes Verfahren genannt. Bei einer aktiven Strategie gibt der Benutzer die Daten, oder Teile davon, seines Profil selbst ein, bei der passiven Strategie wird aus dem Verhalten des Benutzers mit Hilfe entsprechender Regeln auf die Daten geschlossen. Möglich ist auch der Zukauf von Daten, was aus datenschutzrechtlichen Gründen nicht unproblematisch, im kommerziellen Umfeld aber nicht unüblich ist.

Das Ziel von Profiling ist die geordnete Sammlung von Daten, die die Zuordnung von Diensten, Layout und Inhalten zu Benutzern oder Benutzergruppen erleichtern soll. Gesammelt werden kann alles, was über den Benutzer zur Verfügung steht oder aus seinem Verhalten geschlossen werden kann. Entscheidend für die konkrete Auswahl der Profilattribute ist die Art des Systems, das die Profilinformationen nutzen soll. Abbildung 3.1 gibt eine Klassifizierung der Profiling-Daten wieder. Die Profiling-Daten werden dabei in drei Hauptkategorien eingeteilt: Benutzerdaten, Benutzungsdaten und Umgebungsdaten (nach [KoKoe+01, Seiten 115-121]).

Zu den Benutzerdaten gehören die demographischen Daten, Informationen über die Kenntnisse, Fertigkeiten und Fähigkeiten des Benutzers, Interessen, Vorlieben sowie Ziele und Pläne.

Demographische Daten bestehen aus den „objektiven“ Tatsachen, wie z.B.

- die Daten einer Visitenkarte, wie Name, Adresse Telefonnummer etc.,
- geographische Daten, wie Land, Stadt, Stadtteil etc.,
- Benutzercharakteristika, wie Alter, Geschlecht, Schulbildung, Einkommen etc.,

Benutzerdaten	Benutzungsdaten	Umgebungsdaten
Demographische Daten	Beobachtbare Nutzungsdaten	Software des Rechners
Benutzerkenntnis	Temporäres Beobachtungsverhalten	Hardware des Rechners
Benutzerfertigkeit/ -fähigkeit	Auswahlaktionen	Standort
Interessen/Vorlieben	Ratings	Umfeld
Daten über Ziele/Pläne	Kaufinformationen	
	Regelmäßigkeit der Benutzung	
	Häufigkeit	
	Serie von Aktionen	

Abbildung 3.1: Klassifizierung der Profiling-Daten.(Abbildung aus [Jac02, Seite 11]).

- psychographische Daten, darunter fallen solche Daten, die etwas über die Lebensart des Benutzers preisgeben,
- Kundendaten, wie die Häufigkeit der Inanspruchnahme von Dienstleistungen und
- Registrierungsdaten für Informationsangebote.

Informationen über die Kenntnisse eines Benutzers, seine Fertigkeiten und Fähigkeiten können verwendet werden, um das System entsprechend zu präsentieren. So kann einem Experten mehr an Inhalt und Befehlen angeboten werden, als einem Anfänger. Insbesondere im Bereich des E-Learning wird eine solche Art der Personalisierung immer wieder gerne gefordert, aber schon bei den intelligenten, tutoriellen Systemen (vgl. z.B. [Zie94]) haben sich in der Praxis vielfältige Probleme ergeben.

Benutzungsdaten sind solche Daten, die sich entweder unmittelbar aus der Verwendung des Systems durch einen Benutzer ergeben, oder daraus direkt abgeleitet werden können.

Zu den beobachtbaren Benutzungsdaten zählen:

- Auswahlaktionen (selective actions): Bei Auswahlaktionen wird festgehalten, wann und für welchen angebotenen Dienst sich ein Benutzer entschieden hat. Einhergehend ist die negative Auswahl, wofür sich der Benutzer eben nicht entschieden hat.
- Temporäres Beobachtungsverhalten (temporal viewing behavior): Hier wird versucht festzustellen, wie lange ein Benutzer sich eine Seite angesehen hat. Dabei ist eine negative Aussage („der Benutzer hat sehr schnell einen anderen Link auf der Seite gewählt“) leichter zu treffen, als die positive Aussage, dass ein Benutzer sich die Seite länger angesehen hat, da der Benutzer sich nebenher auch mit anderen Dingen beschäftigt oder den Browser geschlossen haben könnte.

- Ratings: Bei der Technik der Ratings geben Benutzer Bewertungen für Informationsobjekte oder Dienste ab. Die Bewertungen richten sich typischerweise nach einer festen Skala, so dass die Bewertungen gut maschinell auswertbar sind.

- Kaufinformationen und kaufbezogene Informationen: wie z.B. welche Produkte gekauft wurden oder an welchen Gewinnspielen teilgenommen wurde.

Unter der Regelmäßigkeit der Benutzung werden Informationen über die Häufigkeit der Verwendung des Systems durch einen Benutzer verstanden. Diese Informationen sind insbesondere im Rahmen des eCommerce interessant, da u.U. gelegentliche Benutzer anders beworben werden sollen als regelmäßige Benutzer. Doch auch in Unternehmensportalen können solche Informationen genutzt werden, um gelegentlichen Benutzern mehr automatische Hilfestellung zu geben, als häufigen Benutzern. Die Serien von Aktionen, die ein Benutzer durchführt, können dazu genutzt werden, die für den Benutzer typischen Aktionen, deutlicher herauszustellen oder „Abkürzungen“ anzubieten.

Zu den Umgebungsdaten gehören Daten über die Software des Benutzerrechners (Browser Versionen, Plug-Ins, Java- oder Java-Script-Fähigkeit etc.) und über die Hardware des Rechners (Bandbreite der Datenleitung, Prozessorgeschwindigkeit, Bildschirmauflösung und Eingabegeräte). Außerdem fallen in diese Kategorie die Informationen, die über den Standort oder das Umfeld des Benutzers Auskunft geben.

Zur Sammlung der Profiling-Daten werden direkte und indirekte Verfahren angewandt und externe Sekundärdaten hinzugezogen. Abbildung 3.2 gibt einen Überblick über die Datengewinnungsmethoden. Bei den direkten Verfahren geben die Benutzer freiwillig ihre Informationen preis. Die aus den direkten Verfahren gewonnenen Daten sind häufig zuverlässiger als die der anderen Methoden. Wie auch bei den anderen Verfahren kann allerdings nie ihre Richtigkeit vollständig nachgeprüft werden. Ein klassisches direktes Verfahren ist die Eingabe einer Benutzerkennung, personenbezogene Angaben werden häufig über Fragebögen o.ä. realisiert. Zu den indirekten Verfahren zählen vor allem technische Auswertungsmöglichkeiten, wie Logfiles oder Application Server Logging. Externe Sekundärdaten stammen „von außerhalb“, d.h. nicht direkt oder indirekt aus dem System.

3.2.4 Objekte und Methoden der Anpassung

In HTML-basierten Anwendungen werden in der Literatur zwei Ebenen der Anpassung unterschieden: die Anpassung des Inhalts (content-level adaption) und die Anpassung der Navigation (link-level adaption) ([Bru96, Seite 96]).

Bei der Anpassung des Inhalts wird nicht nur verschiedenen Benutzern auf ein und derselben Seite unterschiedlicher Text oder anderer multi-medialer Inhalt, angezeigt. Die Anpassung des Inhalts bezieht sich vielmehr auf die gesamte Präsentation, das sog. Look&Feel, also z.B. auch auf die Farbgebung oder die Anordnung der Elemente. Die grundlegende Methode für die Anpassung des Inhalts ist sog. *bedingter Text* ([Bru96, Seite 101]) der nur dann einem Benutzer angezeigt wird, wenn die assoziierte Bedingung wahr ist. Andere Methoden, wie *stretchtext* oder *explanation variants* können leicht auf bedingten Text zurückgeführt werden.

Direkte Verfahren	Indirekte Verfahren	Externe Sekundärdaten
User ID	Logfiles	Zukauf
Personenbezogene Angaben	Packet Sniffing	Fusion/Kooperation
Feedback	Application Server Logging	
	Cookies, OPS, P3P	
	Session ID	

Abbildung 3.2: Datengewinnungsmethoden der Personalisierung (Abbildung aus [Jac02, Seite 11]).

Die Anpassung der Navigation bezieht sich auf die in HTML-Seiten enthaltenen Verweise, über die man zu anderen Seiten gelangen oder Funktionen ausführen kann. Es können fünf Methoden der Anpassung der Navigation unterschieden werden:

- Direktes Leiten (direct guidance): Das System empfiehlt explizit, einem bestimmten Verweis zu folgen, in dem es ihn besonders hervorhebt oder kennzeichnet.Anpassung der Anordnung (adaptive ordering): Verweise, die das System empfiehlt werden auf der Seite weiter oben angezeigt. Der am meisten empfohlene Verweis steht ganz oben, der am wenigsten empfohlene Verweis ganz unten. Diese Methode der Anpassungen findet vor allem bei Listen mit Empfehlungen Anwendung, wie sie gerade in Shop-Systemen oder Systemen des Wissensmanagements häufig vorkommen.

- Verstecken (hiding): Nicht empfohlene oder erlaubte Verweise werden nicht angezeigt. Diese Methode wird sehr häufig zum Ziel der Umsetzung von Zugriffsrechten angewendet.

- Kommentierung (adaptive annotation): Verweise werden mit Hinweisen, häufig „Tool-Tips“ oder nur „Tips“ genannt, versehen. Diese Methode ist Standard in jeder modernen Anwendung.

- Anpassung der Karte (map adaption): Spezielle Index-Seiten (sog. Site Maps), die einen Überblick über die im System vorhanden Seiten geben betonen oder verbergen Verweise und kommentieren sie.

3.2.5 Dienste für die Personalisierung

Um die Personalisierung eines Unternehmensportals tatsächlich durchzuführen, werden technische und benutzerorientierte Dienste verwendet, die auf der Grundlage der oben

beschriebenen Profile und Methoden, die Objekte und Navigationsmöglichkeiten im Portal anpassen.

Die Dienste zur Personalisierung lassen sich in folgende Gruppen einteilen:

- Dienste für die Mehrsprachigkeit: Hierunter fallen die Dienste, die die Mehrsprachigkeit der Oberfläche und der Informationsobjekte bewerkstelligen.

- Benachrichtigungsdienste (Altering): Dienste, die den Benutzer auf verschiedene Art und Weise auf Neuigkeiten, Änderungen o.ä. aufmerksam machen.

- Empfehlungsdienste (Recommendations): Dienste, die dem Benutzer auf Grund seines bisherigen Verhaltens andere für ihn interessante Informationsobjekte oder Dienste vorschlagen.

- Persönliche und allgemeine Sichten: Diese Dienste ermöglichen es einem Benutzer oder einer Benutzergruppe, seine bzw. ihre eigene Sicht auf die Informationsobjekte und Dienste eines Portals zu definieren. Derartige Dienste bieten z.B. eine freie Farbgestaltung oder die Umordnung von Oberflächenelementen an. Aber auch Sammelmappen oder Shopping-Carts, d.h. die persönliche Ordnung von Informationsobjekten fallen in diese Dienstgruppe.

- Sicherheitsdienste: Diese Dienste setzen das Sicherheitskonzept eines Portals um, also insbesondere die Frage der Zugriffsrechte.

Tabelle 3.1 fasst die Dienste, die verfolgten Ziele, die Informationsquellen im Profil und die verwendeten Techniken zusammen (nach [Jac02, Seite 36]).

Nachdem in diesem Kapitel Aspekte der Personalisierung beleuchtet wurden, soll nun eine Referenzarchitektur für Portale vorgestellt werden, die die bisherigen Ergebnisse vereint.

3.3 Eine Referenzarchitektur für Portale und Dienste

Eine Systemarchitektur für Portale und Dienste muss Systeme für die Bereiche Verteilung, Generierung der Oberfläche, spezielle Portaldienste sowie benutzerorientierte und technische Dienste beinhalten. In Abbildung 3.3 wird eine derartige Systemarchitektur für Portale und Dienste mit typischen Komponenten dargestellt.

Die Architektur gliedert sich in drei Teile:

- Kommunikationssysteme

- Portalsystem

- Dienstleistungssysteme

Dienst	**Ziele**	**Beispiele Informationsquellen**	**Beispiele Techniken**
Mehrsprachigkeit	M, B	Demographische Daten, Daten über den verwendeten Client oder den Standort	HTML: RFC2070, Java: Resource Bundles
Benachrichtigung	B, M, K	Kontaktdaten, Interessen und Vorlieben	Name Recognition, Notifikation, Empfehlungen
Empfehlungen	B, K	Interessen und Vorlieben	Feature Based Filtering, Collaborative Filtering
Persönliche und allgemeine Sichten	B, M, K	Interessen und Vorlieben, Kenntnisse, Fähigkeiten, Fertigkeiten, Informationen über Layout und Funktionen	Lesezeichen, Sammelmappen, Shopping-Carts, Topic-Maps, Annotationen
Sicherheit	Z	Benutzerkennung, Rollen, Mitgliedschaften in Gruppen	Zugriffslisten (ACLs), Rollen und Worksets (SAP Portals)

Tabelle 3.1: (B=Bedienungsfreundlichkeit, K=Kooperationsunterstützung, M=Marketing, Z=Zugriffskontrolle)

Die Kommunikationssysteme sind für die Kommunikation mit den Benutzern des Systems verantwortlich. Sie nehmen die Anfragen entgegen und versenden entsprechende Antworten. Üblicherweise wird das HTTP-Protokoll zur Kommunikation verwendet, eine Kommunikation über SMTP, WAP oder ein anderen Protokoll ist aber genauso möglich. Der Inhalt, der über das Protokoll versendet wird, hängt vom Anwendungsprogramm des Benutzers, meist einem Web-Browser, und von der generierten Oberfläche ab.

Die eigentliche Portalsystem gliedert sich in zwei Teile: Die Oberflächengenerierung und die Portaldienste. Die Oberflächengenerierung muss mehrere Faktoren berücksichtigen:

- Die vom Benutzer verwendete Applikation zur Kommunikation. Entsprechend muss HTML, WML, XML oder anderes erzeugt werden. Im Idealfall wird die erzeugte Oberfläche noch jeweils auf das Endgerät gesondert abgestimmt.

- Die vergebenen Zugriffsrechte bestimmen, welche Elemente der Oberfläche ein Benutzer sehen oder verwenden darf, und welche nicht.

- Ein vorgegebenes (Seiten-)Layout und evtl. vorgenommene individuelle Anpassungen. Zum Layout einer Seite gehören der Inhalt, d.h. welche Informationen oder Funktionen und welche Navigationsmöglichkeiten auf der Seite enthalten sind und das Erscheinungsbild der Seite, d.h. wie der Inhalt angeordnet ist, welche Farben verwendet werden etc..

- Die Sprache des Benutzers, um die Oberfläche in der richtigen Sprache zu erzeugen.

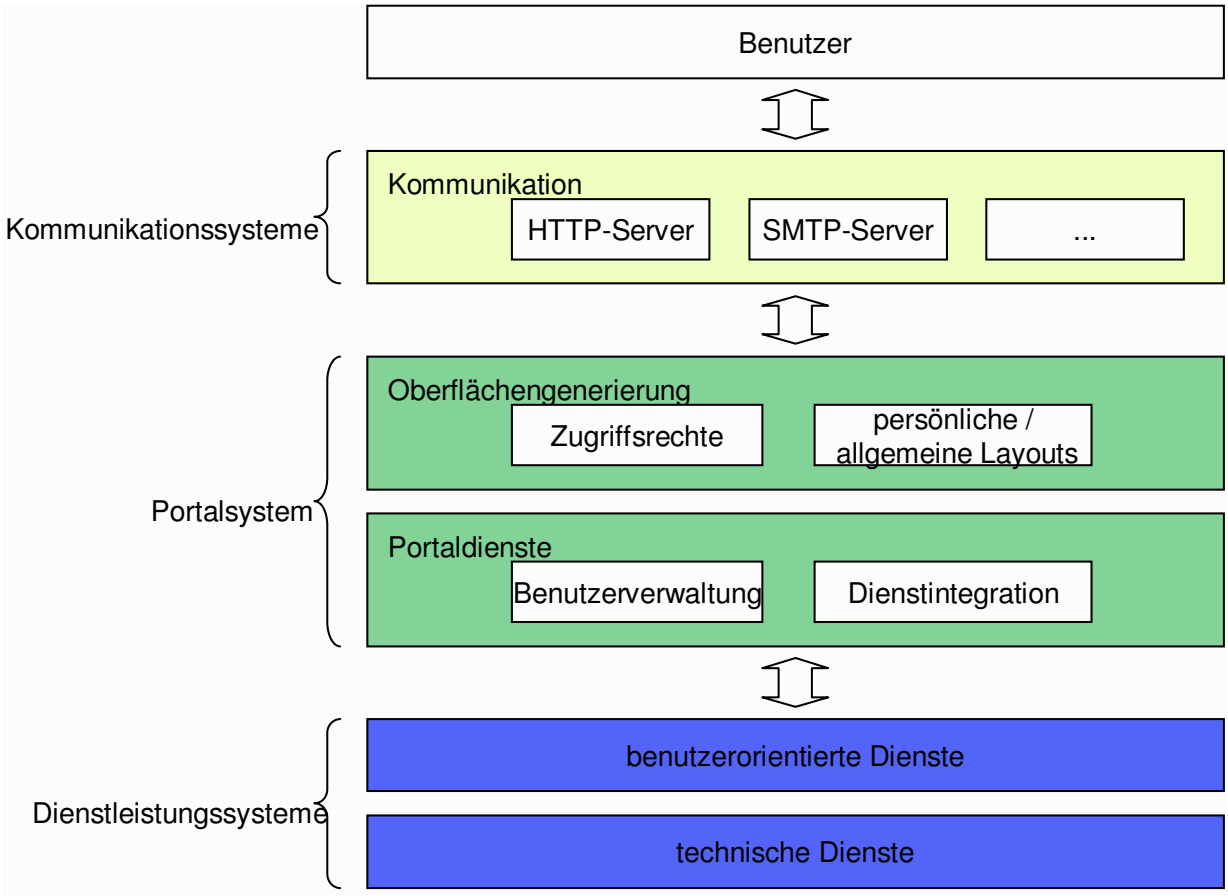

Abbildung 3.3: Eine Referenzarchitektur für Portale und Dienste.

- Ggf. Hinweise für die Kommunikationssysteme und die Applikation des Anwenders, welche Teile einer Seite gepuffert werden können und welche nicht.

Unter Portaldiensten sind die für Portale in Kapitel 2.2.1 genannten typischen Dienste zu verstehen. Eine zentrale Rolle spielen hier die Benutzerverwaltung und die Integration externer Dienste. Das Profil eines Benutzers in der Benutzerverwaltung enthält zumindest solche Daten, die für das Portalsystem unmittelbar relevant sind. Beispielsweise müssen auf der Grundlage des Profils die Berechtigungen eines Benutzers innerhalb des Portals bestimmbar sein oder seine Identitäten in externen Dienstleistungssystemen, soweit seine Berechtigungen im Portal einen Zugriff auf diese Dienstleistungssysteme gestatten. Die Benutzerverwaltung muss nicht ein Bestandteil des Portalsystems sein, es genügt, wenn eine externe Benutzerverwaltung entsprechend integrierbar ist. Häufig verwendete externe Benutzerverwaltungen sind z.B. LDAP-Server, da sie für verschiedene Systeme innerhalb eines Unternehmens die Benutzerdaten verwalten können und somit die oft aufwändige Abgleichung redundant gespeicherter Benutzerdaten entfällt.

Mit Hilfe der Dienstintegration stellt ein Portal – im Sinne der in Kapitel 2.2.1 gege-

benen Definition – die eigentlichen Dienste für die Anwender zur Verfügung. Die Dienstintegration sollte eine uniforme Sicht auf die im Portal möglichen Dienste anbieten. Die Dienstintegration kann sich in der Praxis als sehr komplex erweisen, insbesondere wenn Dienste aus verschiedenen, voneinander unabhängigen Dienstleistungssystemen integriert werden sollen. SAP beispielsweise widmet im Rahmen der mySAP-Technologie der Integration von Diensten allein drei Server, nämlich den Unification-Server, den Exchange Server und den Web Application Server, vgl. Kapitel 2.2.5.

Die Dienstleistungssysteme sind die eigentlichen Leistungserbringer innerhalb eines Portals. Sie stellen die Funktionen zur Verfügung, die im Portal angeboten werden. Ein Dienstleistungssystem kann nur benutzerorientierte Dienste dem Anwender im Portal zur Verfügung stellen, technische Dienste besitzen per Definition keine Repräsentation für einen Anwender, vgl. Kapitel 3.1.

Auf allen Ebenen der Referenzarchitektur kann eine Personalisierung erfolgen. Dies beginnt auf der Kommunikationsebene, wo die Art und Weise der Auslieferung speziell auf den Benutzer bzw. den verwendeten Kommunikationskanal abgestimmt werden kann. Innerhalb des Portalsystems kann bei der Generierung der Oberfläche auf das Profil des Benutzers eingegangen werden, indem etwa die Oberfläche in der gewünschten Sprache erzeugt wird oder Informationen oder Funktionen in Abhängigkeit der Zugriffsrechte des Benutzers ein- oder ausgeblendet werden. Auch benutzerorientierte Dienste können personalisiert sein, d.h. sie können sich in Abhängigkeit des jeweiligen Benutzers anders verhalten, indem sie z.B. andere Ergebnisse liefern. Typische benutzerorientierte Dienste, die personalisiert sind, sind etwa Benachrichtigungsdienste, Empfehlungsdienste oder Einkaufskörbe. Bei technischen Diensten ist die Bezeichnung „personalisiert“ nicht ganz angebracht, da die „Person“, also der Benutzer, sie nicht zu sehen bekommt; technische Dienste können jedoch sehr wohl mit Daten aus dem Benutzerprofil parametrisiert sein.

Für ein konkretes Portal muss festgelegt werden, welche Dienste auf welchen Seiten erscheinen, welche Zugriffsrechte für welche Benutzer gelten und welche Layouts und damit einhergehen welche Navigationspfade Anwendung finden sollen. Dienste, Zugriffsrechte und Layouts dienen quasi als Bausteine des „Hauses“ Portal. Die eigentliche Definition des Portals geschieht mit Hilfe eines Controllers, der die Bausteine und ihr Verwendung kennt und zur Laufzeit in der gewünschten Form zusammensetzt und an die Kommunikationssysteme weiterreicht. Abbildung 3.4 stellt den Zusammenhang graphisch dar.

3.4 Zusammenfassung

In diesem Kapitel wurde das Wesen eines Portals näher beleuchtet. Dazu wurden zunächst die verschiedenen Arten von Diensten untersucht, die in einem Portal anzutreffen sind. Es wurde dabei zwischen *technischen* und *benutzerorientierten* Diensten unterschieden.

Anschließend wurden die verschiedenen Beweggründe diskutiert, aus denen heraus das Schlagwort der Personalisierung verwendet wird. Danach wurde der eigentliche Prozess der Personalisierung beleuchtet, der sich aus den Schritten Tracking, Profiling und Matching zusammensetzt. Nach der Erörterung des Trackings und des Profilings, bei dem

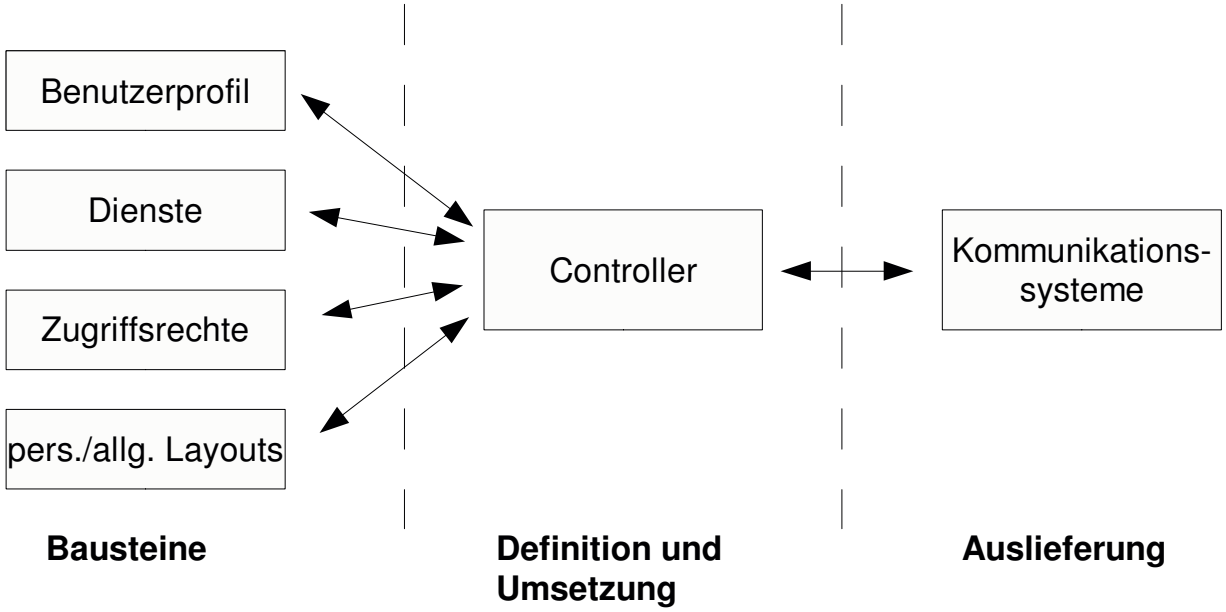

Abbildung 3.4: Konkrete Portale als Zusammensetzung von Bausteinen mit dem Controller als verbindendes Element.

Daten über den Benutzer gesammelt und aufbereitet werden, wurde auf die verschiedenen Objekte und Methoden des Matching eingegangen. Matching bedeutet im Wesentlichen das Anpassen des Inhalts und der Navigation auf den jeweiligen Benutzer. Die Personalisierung selbst wird dann von Diensten vorgenommen, hierunter fallen Dienste für die Mehrsprachigkeit, Benachrichtigungsdienste, Empfehlungsdienste, Dienste für persönliche und allgemeine Sichten und Sicherheitsdienste.

Die als Ergebnis der vorherigen Diskussionen und Erörterungen vorgestellte Referenzarchitektur für Portale und Dienste gliederte sich in drei Hauptbestandteile: Die Kommunikationssysteme, das eigentliche Portalsystem und die Dienstleistungssysteme. Die Hauptbestandteile bestanden ihrerseits aus mehreren Unterbestandteilen, die für bestimmte Aspekte zuständig waren. Möglichkeiten zur Personalisierung fanden sich über die gesamte Referenzarchitektur verteilt.

Im folgenden Kapitel wird das Modell der Enterprise Portal Maps definiert, das die Bausteine eines Portals metaphorisch interpretiert und ihnen konkrete Eigenschaften zuordnet.

Kapitel 4

Das Modell der Enterprise Portal Maps

> „Much of how we think in later life is based on what we learn in early life about the world of space.“ ([Min85, Seite 288])

In diesem Kapitel wird das Modell der Enterprise Portal Maps, kurz das EPM-Modell genannt, vorgestellt, das die Bausteine eines Portals metaphorisch betrachtet. Zunächst wird die Grundidee erläutert und Attraktivität von (ÖPNV-)Netzplänen als Kommunikationsmittel veranschaulicht. Anschließend werden die Konzepte Bedingung, Dienst, Station, Linie und Netzplan im Einzelnen vorgestellt.

4.1 Portale und Dienste als Streckennetze und Stationen

Dem Modell der Enterprise Portal Maps liegt der Gedanke zugrunde, Seiten und Navigationsmöglichkeiten eines Portals als Stationen und Linien zu interpretieren, wie sie im Bereich des öffentlichen Personennahverkehrs (ÖPNV) vorkommen. Der Benutzer eines Portals „fährt“ so über das „Liniennetz“ (die Navigationsmöglichkeiten) von „Station“ zu „Station“ (von Seite zu Seite). Aus dieser Sichtweise auf Portale leitet sich die Möglichkeit ab, die angebotenen Seiten, Dienste und Navigationsmöglichkeiten im Portal in Form der bekannten Karten des ÖPNV, auch Netzpläne genannt, darzustellen. Der große Vorteil der üblichen Darstellungsart von Netzplänen im ÖPNV besteht in ihrer erwiesenen Übersichtlichkeit und Verständlichkeit auch bei großen und komplexen Netzen selbst für nur gelegentliche Benutzer.

Die allgemeine Attraktivität der Darstellung von Netzplänen angewandt auf Abläufe und Zusammenhänge im Bereich der Informatik, wurde bereits mehrfach untersucht. So wird etwa in [SaGr+01] beschrieben, wie Benutzern einer Website geführte Touren (guided tours) mit Hilfe der Netzpläne des ÖPNV angeboten werden. Diese Touren verstehen sich

als geordnete Sammlung beliebiger HTML-Seiten, die ohne Vor- oder Nachbedingung aufgerufen werden können. Während einer solchen Tour werden in einem separaten Fenster (Frame) die aktuelle Station und die unmittelbar benachbarten Stationen dem Benutzer angezeigt. Ein anderer Ansatz wird in [Zie98] verfolgt, wo gezeigt wird, wie mit Hilfe der Metapher des ÖPNV die Arbeitsabläufe (Workflows) eines Unternehmens vollständig beschrieben und visualisiert werden können. Hierbei werden die einzelnen Arbeitsschritte eines Workflows als Dienste (Services) an Stationen interpretiert, der Kontrollfluss des Workflows findet sich in den Fahrplänen der Stationen wieder. Jedes Workflow-Schema wird als Linie oder Linienvariante interpretiert. Über die gemeinsame Nutzung von Arbeitsschritten an Stationen entstehen Umsteigestationen, die von mehreren Linien angefahren werden können, d.h. in mehreren Workflows verwendet werden.

Um die Attraktivität graphischer Notationen deutlich zu machen, wird im Folgenden kurz auf Karten als Kommunikationsmittel im Allgemeinen eingegangen, um anschließend einen Überblick über die Notation der Netzpläne im ÖPNV zu geben, die seit über siebzig Jahren sehr erfolgreich und nur mit wenigen Modifikationen verwendet wird. Die Ausführungen lehnen sich an die in [Zie98, Seiten 44-49] an.

4.1.1 Karten als graphischer Kommunikationsprozess

Karten werden schon sehr lange als Mittel zur Kommunikation von Sachverhalten eingesetzt, beispielsweise verfasste Ptolemäus im 2 Jh. n. Chr. 8 Bände mit dem geographischen Weltbild der damaligen Zeit ([dtv90], Stichwort „Ptolemäus“). Mitte des letzten Jahrhunderts kritisierte Robinson ([Rob52]), dass das Künstlerische im Vordergrund der Kartengestaltung stünde und die Behandlung von Karten als Kunst zu zufälligen und launischen Entscheidungen führe. Als Ausweg betrachtete Robinson entweder die vollständige Standardisierung aller Kartensymbole oder die Erforschung des Wahrnehmens von Karteninhalten, so dass Entscheidungen über Symbole und Design auf der Basis von „objektiven“ Regeln getroffen werden könnten.

Ende der sechziger Jahre setzte sich – durch Robinsons Arbeit beeinflusst – die Ansicht durch, dass Kartographie ein graphischer Kommunikationsprozess sei ([Mac95, Seite 3]). Am Anfang des Prozesses entscheidet der Kartograph, welche der ihm zur Verfügung stehenden Informationen auf der Karte abgebildet werden (enthalten sein) sollen und erstellt anschließend die Karte. Die Karte wird von Benutzern gelesen, die ihrerseits die Informationen, die sie der Karte entnehmen, zu ihrem vorhandenen Wissen hinzufügen. Offensichtlich beinhaltet dieser Kommunikationsprozess die gleichen Fehlerquellen, wie allgemeine menschliche Kommunikationsprozesse:

- Der „Sender“ der Information, in diesem Falle der Kartograph, verfügt selbst nur über lückenhafte oder falsche Informationen über das, was er kommunizieren möchte.

- Die vom Kartograph verwendete Symbolik wird vom Benutzer der Karte nicht in der vom Kartographen gewünschten Weise interpretiert, da der Benutzer über ein anderes Wissen verfügt als der Kartograph. Auf diese Weise kann sich der Benutzer

der Karte Wissen aneignen, das entweder falsch ist oder über das der Kartograph selbst nicht verfügte.

Gegen diese Auffassung von Kartographie erhebt McEachren große Einwände ([Mac95, Seite 6ff]). So gibt er zu bedenken, dass es nur eine kleine Menge von Karten gebe, die tatsächlich über so etwas wie eine „vordefinierte Botschaft“ verfügen, die kommuniziert werden kann. Der Kartograph könne bei allgemeinen Karten nicht wissen, welche Informationen der Leser aus einer Karte gewinnen wolle. Weiterhin bemängelt er, das innerhalb der Kartographie der menschliche Leser zu sehr als „reaktives System“ betrachtet werde, dass, wenn man es nur mit den richtigen Eingaben füttere, auch das richtige Wissen erlange. MacEachern vertritt die Auffassung, dass es keinen einzelnen richtigen wissenschaftlichen oder unwissenschaftlichen Ansatz dazu gibt, wie Karten funktionierten ([Mac95, Seite 12]).

Die eben wiedergegebene Argumentation trifft für Netzpläne im ÖPNV jedoch nur zum Teil zu, da es sich um sehr spezielle Karten handelt. Sie dienen dazu, den Reisenden die wichtigsten Fragen vor und während der Fahrt zu beantworten: „Wo bin ich im System? Wo ist meine Endstation? Muss ich umsteigen? Wenn ja, wo und welche Linie? In welche Richtung muss ich fahren? Was ist der Stationsname am Ende der Linie? Wie viele Stationen muss ich fahren, bevor ich aussteige?“ ([Mon91, Seite 35]). Entsprechend muss ein Netzplan also insbesondere die Stationsfolgen und die Umsteigemöglichkeiten besonders hervorheben. Zusätzlich ist ein guter Netzplan immer auch ein Stück Werbung für die angebotene (Transport-) Dienstleistung, so nennt Monmonier ([Mon91, Seite 58]) zwei Gründe für eine Symbiose aus Karte und Werbeanzeige:

- Die Notwendigkeit, graphische Überschneidungen bei Karten zu vermeiden, kann dazu verwendet werden, bei Werbeanzeigen Auslassungen und Übertreibungen so zu gestalten, dass der Betrachter dies kaum bemerkt.

- Werbeanzeigen müssen Aufmerksamkeit erregen und Karten sind erwiesene Aufmerksamkeitserreger.

Für Netzpläne scheint sich überdies der erste Ausweg von Robinson durchzusetzen: Es gibt eine Notation für Netzpläne, die so erfolgreich ist, dass sie sich – von den jeweiligen Verkehrsunternehmen bzw. -verbünden stets nur wenig variiert – als Quasi-Standard für Netzpläne immer mehr durchsetzt. Dieser Standard beruht auf dem Design des Netzplans der London Underground und wurde durch H.C. Beck in den frühen dreißiger Jahren des letzten Jahrhundert erfunden.

4.1.2 Londons Netzplan: Ein historischer Abriss

Im Jahre 1908 erschien erstmals eine Karte, die die Verbindungen der konkurrierenden elektronischen Bahnen Londons gemeinsam auf einer Karte zeigte (Abbildung 4.1). Schon diese Karte sollte besser als Diagramm bezeichnet werden, da sie geographische Begebenheiten aus Gestaltungsgesichtspunkten heraus ignoriert. So deutet sie zwar schwach Straßen und

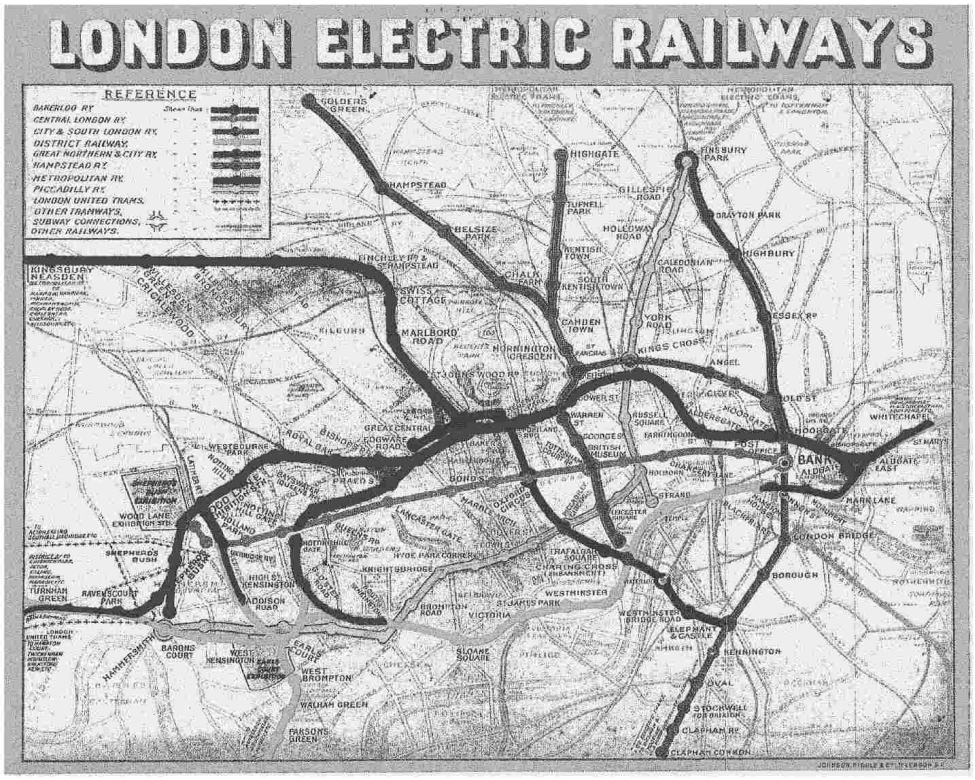

Abbildung 4.1: Frühe Darstellung der elektrischen Bahnen in London.

Flüsse an, der Verlauf der Metropolitan Railway ist im nordwestlichen Teilstück aber falsch wiedergegeben, um die Legende in der linken oberen Ecke platzieren zu können.

Die Verfälschung geographischer Tatsachen ist typisch für Netzpläne und zielt auf das Wesentliche ab: Die eindeutige Darstellung des Linienverlaufs im Sinne von aufeinander folgenden Stationen. Die geographische Anordnung der Stationen tritt dahinter zurück. Dieses Prinzip wurde schon sehr früh bei angewandt: In ([BuMc81, Seite 104]) wird eine Darstellung aus dem Jahre 1846 erwähnt, die die Eisenbahnlinie zwischen London und Turnbridge Wells allein als Folge von Stationen beschreibt.

Im Jahre 1931 entwickelte der technische Zeichner Henry C. Beck die ersten Entwürfe für einen neuen Netzplan der London Underground. Seine initiale Idee beschreibt Beck so ([Gar94, Seite 17]):

> „Während ich die alte Karte der Underground Railways betrachtete, kam mir die Idee, dass es möglich sein könnte, die Karte aufzuräumen, indem Linien begradigt werden; dabei experimentierte ich mit Diagonalen und glich die Abstände zwischen den Stationen aus. Je mehr ich darüber nachdachte, desto überzeugter war ich, dass es einen Versuch wert sei und machte einen ersten Entwurf, bei dem ich die Central London Railway als horizontale Basislinie auswählte. Ich versuchte mir vorzustellen, dass ich eine konvexe Linse oder

Spiegel benutze, um so die zentrale Region in einem größeren Maßstab darzustellen. Dadurch, so dachte ich, würden den Umsteigeinformationen die nötige Klarheit gegeben."

Vor dem Druck des ersten Netzplans, damals Diagramm[1] genannt, ersetze Beck noch die damals üblichen Kreise für Nicht-Umsteigestationen durch Striche, die sog. Ticks. Die Ticks, haben folgende Vorteile ([Gar94, Seite 18]):

- Das Diagramm wirkt als ganzes leichter und deshalb eleganter.
- Ticks beheben die falsche Gewichtung der mit gefüllten Kreisen dargestellten
- Nicht-Umsteigestationen mit den mit ungefüllten Kreisen dargestellten Umsteigestationen.
- Jeder Tick deutet unzweideutig auf den zugehörigen Stationsnamen, was es ermöglicht, die Stationsnamen auf jeweils abwechselnde Seiten der Linie zu schreiben und dadurch eine größere Kompaktheit zu erreichen.

Abbildung 4.2 zeigt das erste Diagramm nach Beck.
Die Gestaltung beruht auf folgenden Prinzipien ([Gar94, Seite 50]):

- Ausschließliche Nutzung von Horizontalen, Vertikalen und 45°-Diagonalen.
- Vergrößerte Darstellung der zentralen Region im Verhältnis zu weiter außen liegenden Teilen.
- Verwendung von „Ticks" zur Kennzeichnung von Stationen, die nur von einer Linie angefahren werden.
- Unterscheidung der Linien durch Farbkodierung.
- Eliminierung aller Oberflächendetails bis auf den Fluss, dessen Verlauf nur stark vereinfacht wiedergegeben wird.
- Die Central Line ist die horizontale Basis des Designs.
- Die Linienführung ist geschwungen und nicht kantig.

Die ersten vier Prinzipien haben sich als defacto-Standard bei der Gestaltung von Netzplänen etabliert, die international Anwendung finden (vgl. [Zie98, S. 76-81]).

[1] Beck lehnte den Begriff „map" für die Darstellung ab und verwendete statt dessen „diagram".

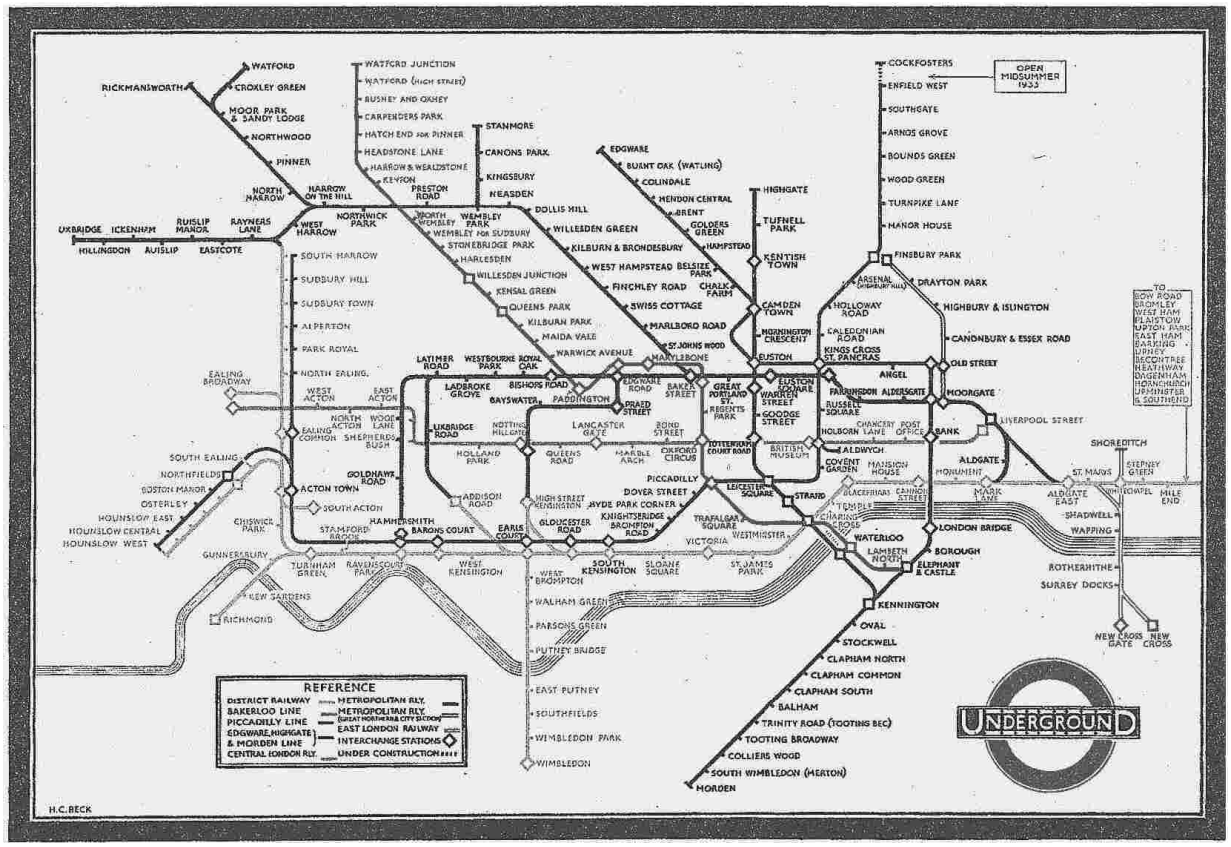

Abbildung 4.2: Das erste Diagramm von H. C. Beck.

4.1.3 Die Idee der Enterprise Portal Maps

Das Modell der Enterprise Portal Maps verwendet Netzpläne, Stationen und Linien metaphorisch für Portale, Dienste und Verweise (Links).

Jede *Station* bietet personalisiert einen *Dienst* oder den Teil eines Dienstes an, den ein Benutzer durch das Besuchen einer Station in Anspruch nimmt. Stationen sind untereinander durch *Linien* verbunden; jede Linie hat eine eindeutige Farbe und kann mehrere Stationen miteinander verbinden. Eine Station besitzt zwei Verzeichnisse (Pläne): den *Dienstplan* und den *Fahrplan*. Der Dienstplan beschreibt den angebotenen Dienst, die *Bedingung*, die erfüllt sein muss, damit dieser Dienst in Anspruch genommen werden kann und eine angepasste Schnittstelle des Dienstes durch *Interaktionselemente*. Ein Eintrag im Fahrplan bestimmt die nächste Station, die Linie dorthin, das *Ereignis*, bei dem dieser Eintrag ausgewählt wird, und ob es sich um einen *direkten* oder *indirekten Transport* handelt.

Stationen und Linien werden auf einem *Netzplan*, auch Karte genannt, verzeichnet, der so die in einem Unternehmensportal angebotenen Dienste verzeichnet und damit eine *Enterprise Portal Map* darstellt.

Im übrigen Teil dieses Kapitels werden die einzelnen Elemente des EPM-Modells detailliert beschrieben.

4.2 Kontexte

Kontexte in Unternehmensportalen beschreiben das Umfeld, in denen sich ein Benutzer bewegt bzw. in denen ein Benutzer Dienste in Anspruch nimmt. Der Kontext eines Benutzers im Rahmen des EPM-Modells ist eine Sammlung von Daten, über die Bedingungen und Dienste des Modells die näheren Einzelheiten ihrer Instanziierung oder ihres Aufrufs entnehmen können. Ein Kontext besitzt eine einfache Typbezeichnung, die Aufschluss über die enthaltenen Daten zulässt. Die Ausgestaltung der Kontexte in einem konkreten System hängt selbstverständlich von der jeweiligen Laufzeitkomponente ab. In Kapitel 5 werden zwei sehr unterschiedliche Implementierungen für Kontexte vorstellt, die auf den besonderen Eigenheiten der Systeme beruhen.

4.3 Bedingungen

Bedingungen bilden die Grundlage der Personalisierung im EPM-Modell. Über Bedingungen werden die unternehmensspezifischen, organisatorischen Vorgaben bezüglich der angebotenen Dienste und ihrer Verknüpfungen im Portal umgesetzt.

Eine Bedingung verfügt über folgende Merkmale:

- Sie besitzt einen systemweit eindeutigen Namen, über den sie identifiziert wird.
- Sie ist zu einem bestimmten Zeitpunkt erfüllt (wahr) oder nicht erfüllt (falsch). Hierfür besitzt sie eine Methode *check*.

  Ob sie erfüllt oder nicht erfüllt ist, kann vom aktuellen Kontext, vom aktuellen Benutzer, von sonstigen externen Faktoren (z.B. Zeit) oder einer beliebigen Kombination aus allen dreien abhängen.

Typische Bedingungen in Portalen lassen sich wie folgt klassifizieren:

1. Benutzer-bezogene Bedingungen, wie z.B. „Benutzer ist Administrator“, „Benutzer ist Buchhalter“ oder „Benutzer ist Mitglied der Gruppe X“.
2. Kontext-bezogene Bedingungen, wie z.B. „Dokument ist nicht zum Löschen vorgemerkt“, „Person ist angemeldet“ oder „Buchungsperiode ist offen“. Welches konkrete Objekt (Dokument, Person, Buchungsperiode, ...) dabei gemeint ist, ergibt sich aus dem jeweiligen Kontext bei Auswertung der Bedingung.
3. Benutzer- und Kontext-spezifische Bedingungen, wie z.B. „Benutzer darf in Buchungsperiode buchen“ oder „Benutzer darf Dokument ändern“.

4. Sonstige Bedingungen, wie z.B. „Es ist Montag“ oder „Es sind nicht mehr als 10 Benutzer angemeldet“.

Durch die wenigen Merkmale der EPM-Bedingungen wird eine große Allgemeinheit des Konzepts erreicht, so dass eine Vielzahl verschiedener Berechtigungs- und/oder Evaluationssysteme bei einer konkreten Implementierung verwendet werden können. Die tatsächliche Form der Repräsentation von Benutzer und Kontext hängt bei einer Implementierung selbstverständlich von der verwendeten Portalsoftware, der Benutzerverwaltung, dem System, aus dem das konkrete Objekt stammt, etc. ab. Es ist Aufgabe der Laufzeitumgebung, die über ihren Namen identifizierte Bedingung zu instantiieren und mit den entsprechenden Parametern (Benutzer und Kontext) zu versorgen (vgl. Kapitel 5.2.2 und Kapitel 5.3.2).

EPM-Bedingungen enthalten keinerlei Hinweise auf eine aufzurufende Methode eines Objektes, eine aufzurufende Funktion oder einen EPM-Dienst. Wenn beispielsweise für zwei verschiedene Methodenaufrufe eines Objekts zwei verschiedene Bedingungen geprüft werden sollen, so müssen hierfür auch zwei EPM-Bedingungen definiert werden.

EPM-Bedingungen sind einfach, d.h. dass sie sich nicht mit Hilfe logischer Operatoren wie *und*, *oder* oder *nicht* kombinieren lassen. Bei der Implementierung einer Bedingung können selbstverständlich andere Bedingungen innerhalb der Methode check kombiniert werden.

Bedingungen an Stationen werden auf Netzplänen durch Farben kodiert, wobei jede Bedingung ihre eigene Farbe erhält.

4.4 Dienste

Die Dienste in Enterprise Portal Maps stellen eine allgemeine, personalisierbare Abstraktion der benutzerorientierten Dienste (vgl. Kapitel 3.1) dar. Sie dienen der Gewinnung einer einheitlichen Modellierungssicht auf die Dienste der Dienstleistungssysteme eines Unternehmensportals.

Ein EPM-Dienst umfasst:

- eine Datenschnittstelle in Form einer Liste von Interaktionselementen,
- die Anforderungsmethode *request*,
- die Ausführungsmethode *submit*,
- die Destruktormethode *release*,
- einen Status,
- Typinformationen über einen ggf. benötigten Kontext für die Anforderung,
- Typinformationen über den Kontext, der nach der Anforderung gültig ist,
- Typinformationen über den Kontext, der nach der Ausführung gesetzt wird,

- die Methode *acceptType*.

Benötigt ein Dienst für seine Anforderung einen Kontext, so wird er als *kontextabhängig* bezeichnet, benötigt ein Dienst keinen Kontext, so heißt er *kontextfrei*.

Der Status eines Dienstes beinhaltet eine reine Statusinformation, z.B. angefordert oder ausgeführt, den Kontext des Dienstes und eine mögliche Nachricht für den Benutzer.

Im Folgenden werden die einzelnen Elemente eines EPM-Dienstes näher beschrieben.

4.4.1 Interaktionselemente

Mit Interaktionselementen wird die Datenschnittstelle eines Dienstes spezifiziert und der Datenaustausch realisiert.

Ein Interaktionselement besitzt einen Namen, über das es identifiziert wird, und eine Bezeichnung. Über *Anpassungen* wird gesteuert, ob ein Interaktionselement zur Laufzeit *sichtbar*, *änderbar* oder *obligatorisch* ist. Ist es keines davon, so gilt es als *unsichtbar*. Jedes Interaktionselement besitzt eine Standardanpassung, sowie möglicherweise weitere Anpassungen, die jeweils unter einer bestimmten Bedingung (vgl. Kapitel 4.3 eintreten. Sind bei der Anforderung eines Dienstes zur Laufzeit mehrere Bedingungen erfüllt, so müssen entsprechende Strategien zur Konfliktlösung angewendet werden, um zu bestimmen, welche Anpassung die entscheidende sein soll. Die Standardanpassung ist gegenüber anderen Anpassungen nachrangig.

Ein Interaktionselement kann

- ein Einzelwert,
- ein Verweis,
- eine Auswahl oder
- eine Liste

sein.

Ein *Einzelwert* enthält einen Wert (ein Objekt), eine für Menschen lesbare Bezeichnung des Wertes und eine Typbezeichnung. Einzelwerte entsprechen einfachen Datentypen in Programmiersprachen bzw. Text- oder Eingabefeldern bei Benutzerapplikationen. Anhand der Typbezeichnung kann dem Einzelwert eine Domäne zugeordnet werden, aus dem der Wert stammen muss. Die Typbezeichnung ist von der jeweiligen Laufzeitumgebung bzw. den unterliegenden Dienstleistungssystemen abhängig und muss entsprechend interpretiert werden, vgl. Kapitel 5.2.1 und 5.3.1.

Ein *Verweis* ist ein Interaktionselement, das die Übertragung eines Kontextes an ein Ziel erlaubt. Ein Verweis enthält den zu übertragenen Kontext, die Typbezeichnung des Kontextes, eine EPM-Station als Ziel und eine EPM-Linie über die die Navigation erfolgt. Die Kontextübertragung kann ggf. auch unterdrückt werden, dies bietet sich für die Navigation zu einer Station mit einem kontextfreiem Dienst an. Ein Verweise können nicht änderbar ober obligatorisch sein.

Alternativ zur Angabe einer EPM-Station kann ein Verweis bei Angabe einer URL auch auf beliebige Ziele, insbesondere Ziele, die außerhalb des Netzplans liegen, weisen. Was bei der Navigation über die URL passiert, ist im Modell undefiniert und muss von der Laufzeitumgebung bzw. Applikation des Benutzer bestimmt werden.

Eine *Auswahl* nimmt einen oder mehrere Werte einer festgelegten Menge von Einzelwerten an. Eine Auswahl ist also eine Teilmenge einer Wertemenge und wird in Benutzerapplikationen typischerweise mit sog. Drop-Down-Listen bei einfacher Auswahl oder als Listen bei mehrfacher Auswahl dargestellt.

Eine *Liste* enthält eine feste, geordnete Anzahl beliebiger Interaktionselemente, die die Struktur der Liste festlegen. Wie zu jedem Interaktionselement kann es auch zu einer Liste Anpassungen geben, wie es auch zu den in der Liste enthaltenen Interaktionselementen eigene Anpassungen geben kann. Eine Liste in ihrer Grundform ist mit der Zeile in einer Tabelle vergleichbar. Wie in einer Tabellenzeile getrennte Werte nebeneinander stehen, so stehen die getrennten Interaktionselemente einer Liste „nebeneinander“. Ist eine Liste als Iterator gekennzeichnet, so entspricht die Liste einer Tabelle, die zeilenweise gelesen oder geschrieben werden kann. Jede Zeile entspricht dabei einer Iteration. Im ersten Iterationsschritt enthält die Liste die Werte der ersten Tabellenzeile, im zweiten Schritt die Werte der zweiten Zeile und so fort.

Abbildung 4.3 stellt die verschiedenen Interaktionselemente und ihrer Zusammenhänge als Klassendiagramm in UML dar.

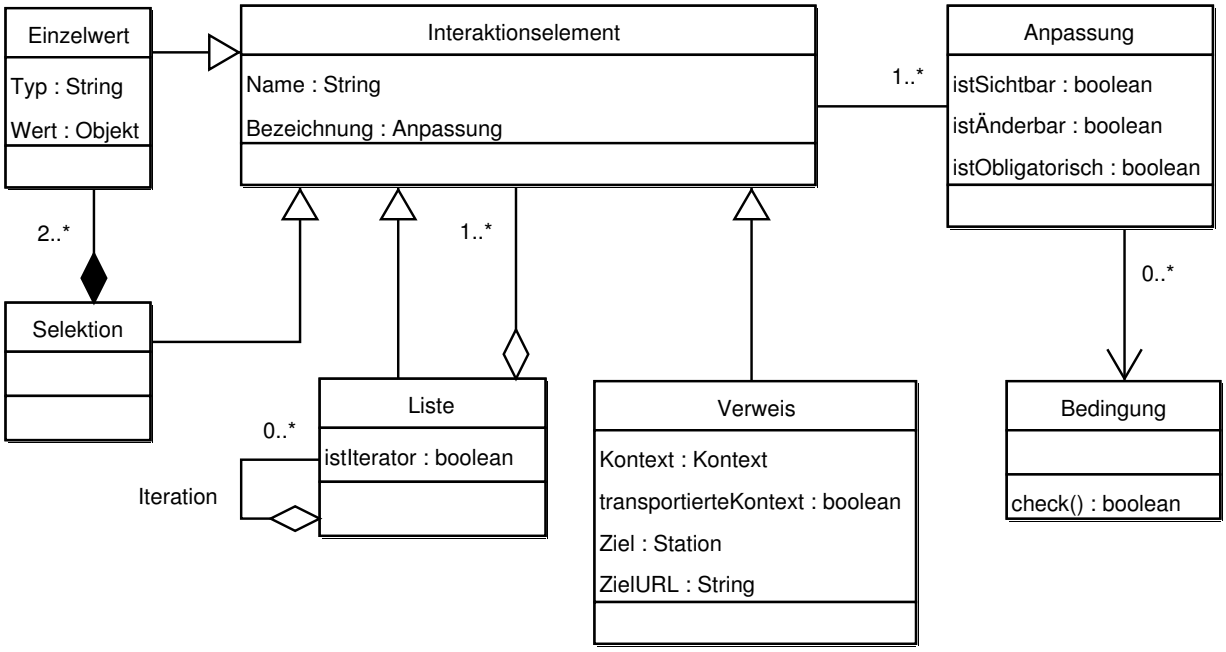

Abbildung 4.3: Konzeptuelles Klassendiagramm der Interaktionselemente.

Die eben beschriebenen Interaktionselemente sollen nun durch ein kleines, aber reales

Beispiel verdeutlicht werden. Angenommen wird ein EPM-Dienst, der Verzeichnisse eines Dokumentenverwaltungssystems anzeigt.

Ein Verzeichnis bestehe aus

- einem Namen,
- einer optionale Beschreibung und
- einer Liste von Dokumenten, die in dem Verzeichnis abgelegt sind.

Verzeichnisse sind hierarchisch geschachtelt, d.h. zu jedem Verzeichnis kann es höchstens ein Ober- aber beliebig viele Unterverzeichnisse geben.

Jedem Verzeichnis können zur Realisierung von Zugriffsrechten drei Gruppen zugeordnet werden:

- Eine Lesergruppe: Mitglieder dieser Gruppe dürfen Dokumente dieses Verzeichnisses lesen.
- Eine Schreibergruppe: Mitglieder dieses Verzeichnisses dürfen bestehende Dokumente in diesem Verzeichnis ändern oder löschen und neue Dokumente in diesem Verzeichnis anlegen.
- Eine Administratorengruppe: Mitglieder dieser Gruppe dürfen das Verzeichnis löschen, neue Unterverzeichnisse anlegen und die Leser-, Schreiber-, und Administratorengruppe festlegen.

Der Dienst verwendet als Schnittstelle eine Liste mit dem Namen „Verzeichnis“, die alle weiteren Interaktionselemente enthält:

- Einen Einzelwert mit der Bezeichnung „Namen“.
- Einen Einzelwert mit der Bezeichnung „Beschreibung“.
- Je einen Verweis für die Leser-, Schreiber- und Administratorengruppe. Die Verweise zeigen auf eine Station mit dem Namen „Gruppe anzeigen“ mit der Linie „Gruppe“.
- Eine Liste „Pfad“, die als Iterator alle übergeordneten Verzeichnisse als Verweise enthält. Der in der Liste enthaltene Verweis zeigt auf die Station „Verzeichnis anzeigen“ mit der Linie „Verzeichnis“.
- Eine Liste „Unterverzeichnisse“ als Iterator über Verweise zu allen direkten Unterverzeichnissen.
- Eine Liste „Dokumente“ als Iterator mit dem Verweis „Name“ auf die Station „Dokument anzeigen“, dem Einzelwert „Erstelldatum“ und dem Einzelwert „Autoren“.

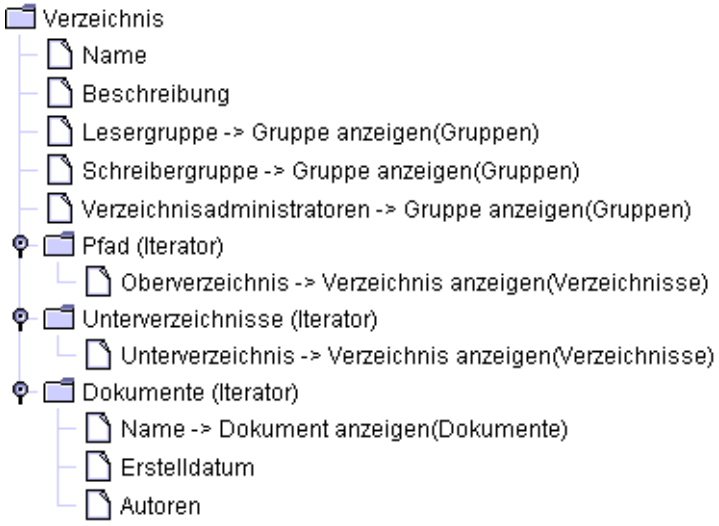

Abbildung 4.4: Interaktionselement eines Verzeichnisdienstes.

Abbildung 4.4 zeigt eine grafische Repräsentation der Struktur der Schnittstelle als Interaktionselement; Listen werden mit Ordnern symbolisiert, Einzelwerte mit Blättern und ihrem Namen, Verweise nennen zusätzlich die Zielstation und in Klammern die Linie.

Wird der Dienst zur Laufzeit mit einem entsprechendem Kontext aufgerufen, so füllt er die Einzelwerte und Verweise mit den entsprechenden Daten. Die Listen „Pfad“, „Unterverzeichnisse“ und „Dokumente“ werden zunächst als einfache Listen mit der passenden Struktur erstellt. Anschließend werden weitere einfache Listen mit den entsprechend gefüllten Einzelwerten und Verweisen als Iterationen zu den Listen „Pfad“, „Unterverzeichnisse“ und „Dokumente“ hinzugefügt.

4.4.2 Anforderung

Ein Dienst wird über die Methode *request* angefordert. Zur Anforderung werden zwei Parameter benötigt:

- Der aktuelle Kontext und
- der angemeldete Benutzer.

Als Ergebnis der Methode wird im Erfolgsfalle eine Instanz des Dienstes zurückgeliefert, deren Interaktionselemente an Werte gemäß des Kontextes gebunden wurden. Der Status der Dienstinstanz ist angefordert. Der im Status enthaltene Kontext wurde gesetzt, dies kann, muss aber nicht, der Kontext sein, mit dem der Dienst aufgerufen wurde. Ist ein Dienst kontextfrei, so ignoriert er den übergebenen Kontext, kann aber selbst einen Kontext gesetzt haben.

Durch die Übergabe des anfordernden Benutzers erhält der Dienst die Möglichkeit, sich zu personalisieren. So kann z.B. ein Dienst zur Anzeige von Beiträgen in einem Diskussionsforum solche Beiträge nicht anzeigen, d.h. keine entsprechenden Interaktionselemente zur Verfügung stellen, die von Personen geschrieben wurden, die der den Dienst anfordernde Benutzer auf eine Ausschlussliste gesetzt hat.

Zusätzlich können Seiteneffekte durch die Anforderung auf der Ebene der Dienstleistungssysteme auftreten. So werden z.B. Änderungsdienste in aller Regel Sperren auf die betreffenden Objekte setzen, damit es nicht zu konkurrierenden Zugriffen durch mehrere Benutzer gleichzeitig kommen kann. Andere Dienste, insbesondere Dienste von Shop-Systemen, könnten etwa Profiling für spätere Personalisierung durchführen und vermerken, wie oft der Benutzer sich einen Artikel schon in den Warenkorb gelegt hat. Das EPM-Modell macht keine Annahmen darüber, welche Seiteneffekte auftreten können, nur, dass es sie geben kann.

4.4.3 Ausführung und Ergebnis

Wurde ein Dienst (erfolgreich) angefordert, so kann er über die Methode *submit* ausgeführt werden.

Als einzigen Parameter erhält die Methode ein Interaktionselement, das von der Struktur her dem Interaktionselement gleichen muss, welches bei der Anforderung durch den Dienst erzeugt wurde.

Das Ergebnis der Ausführung wird im Status des Dienstes abgelegt:

- War die Ausführung erfolgreich, so ist der Status *ausgeführt* und ein mögliches Ergebnis wurde als Kontext im Status abgelegt. Evtl. wurde eine (Erfolgs-) Nachricht für den Benutzer hinterlegt.
- War die Ausführung nicht erfolgreich, so bleibt der Status auf *angefordert* und es wurde eine Fehlermeldung als Nachricht hinterlegt.

4.4.4 Beendigung

Ein Dienst wird beendet, indem seine Methode *release* aufgerufen wird. Diese Methode ist parameterlos und kann zu jeder Zeit und in jedem Status aufgerufen werden. Mit dem Aufruf ist der angeforderte Dienst endgültig beendet und kann nicht mehr weiterverwendet werden.

Mit dem explizitem Ende eines angeforderten Dienstes hat dieser die Gelegenheit, evtl. vorgenommene Seiteneffekte bei der Anforderung oder Ausführung rückgängig zu machen oder endgültig festzuschreiben. Der im Status abgelegte Kontext des Dienstes kann nicht mehr verändert werden – dieses würde auch keinen Sinn mehr machen.

Es ist Aufgabe der Laufzeitumgebung sicherzustellen, dass jeder angeforderte Dienst auch irgendwann beendet wird (vgl. Kapitel 5.2.1 und 5.3.1). Der Ersteller eines Dienstes kann so darauf vertrauen, dass die Methode für jeden angeforderten Dienst auch tatsächlich aufgerufen wird.

4.4.5 Diensttypen

Wie oben beschrieben, gehört zu einem Dienst eine Anforderungsmethode, die die Interaktionselemente an Werte bindet, und eine Ausführungsmethode, die Änderungen an den Interaktionselementen an das zugrunde liegende Dienstleistungssystem überträgt. Der übergebene Kontext kann von beiden Methoden verändert werden.

In der Praxis treten Dienste auf, die entweder keine Interaktionselemente an Werte binden oder die bei der Ausführung keine Änderungen auf das ursprüngliche Objekt übertragen. Entsprechend können drei Klassen von EPM-Dienste unterschieden werden:

- R-Dienste: Binden Werte an Interaktionselemente, übernehmen keine Änderungen.
- RS-Dienste: Binden Werte an Interaktionselemente, übernehmen Änderungen.
- S-Dienste: Binden keine Werte an Interaktionselemente, führen Änderungen durch.

Die folgende Tabelle zeigt die auftretenden Kombinationsmöglichkeiten unter der Berücksichtigung, dass Dienste kontextfrei oder kontextabhängig sein können und gibt typische Dienste für die jeweilige Konstellation an.

Kontextfrei	**Diensttyp**	**Typische Dienste**
ja	R	Anzeige statischer Inhalte
ja	RS	parametrisierte Anlage von Objekten
ja	S	direkte Anlage von Objekten
nein	R	Anzeige oder direkte Manipulation von Objekten
nein	RS	Änderung von Objekten, Anlage von Unter-Objekten
nein	S	Löschen von Objekten

4.4.6 Typisierung von Diensten

Ein Dienst enthält drei Typinformationen, die den Kontext vor der Anforderung, nach der Anforderung und nach der Ausführung des Dienstes beschreiben. Dieses Konzept ist mit den Vor- und Nachbedingungen (Pre- und Post-Conditions) aus Programmiersprachen vergleichbar, nur dass im EPM-Modell statt komplexer logischer Ausdrücke über Variable nur reine Typinformationen zulässig sind. Diese Typisierung von Diensten erlaubt die Prüfung, ob gewisse Folgen von Anforderungen und/oder Ausführungen von Diensten zulässig sind oder nicht. Hierüber lassen sich die möglichen Navigationspfade in einem Unternehmensportal bestimmen, ausgedrückt durch die zulässigen Fahrplaneinträge an den Stationen. Das EPM-Modell ist in diesem Sinne statisch typisiert.

Über die Methode *acceptType* gibt der Dienst zu erkennen, ob zur Laufzeit aus einem entsprechenden Kontext die für Anforderung bzw. Ausführung des Dienstes die benötigten Informationen gewonnen werden können. Innerhalb des EPM-Modells werden keinerlei Annahmen über die Einzelheiten des Typsystems gemacht.

4.5 Stationen

Stationen sind die Orte, an denen Dienste und Navigationsmöglichkeiten angeboten und personalisiert werden.

Die angebotenen Dienste werden im Dienstplan verzeichnet, die angebotenen Transporte werden im Fahrplan aufgeführt.

Es gibt zwei Sichten auf Stationen:

- Die Designersicht: Für Erstellung und Personalisierung. An jeder Station wird genau ein Dienst angefordert oder ausgeführt.
- Die Benutzersicht: Vereinfachte Sicht für Anwender. An jeder Station können mehrere Dienste angeboten werden, unabhängig davon, ob ein Dienst angefordert oder ausgeführt wird.

Aus der Designersicht lässt sich die Benutzersicht unmittelbar ableiten. Bei der Ableitung gehen jedoch Informationen verloren, so dass eine „Rückwandlung“ nicht möglich ist.

Im Folgenden wird mit dem Ausdruck Station stets eine Station aus der Designersicht gemeint, ist eine Station aus Sicht eines Benutzers gemeint, so wird sie als Benutzerstation bezeichnet.

Eine Station wird als sichtbare oder Anforderungsstation bezeichnet, wenn der angebotene Dienst angefordert wird. Eine Station ist unsichtbar oder eine Ausführungsstation, wenn der angebotene Dienst ausgeführt wird.

4.5.1 Dienstplan

Der Dienstplan ist ein Verzeichnis, das den angebotenen Dienst beschreibt. Der Dienstplan einer Station hat vier Einträge:

- Den Namen des Dienstes, der angefordert oder ausgeführt werden soll.
- Die Bedingung, die erfüllt sein muss, damit der Dienst angefordert oder ausgeführt werden kann bzw. darf.
- Den Hinweis, ob der Dienst ausgeführt oder angefordert werden soll.
- Ein Interaktionselement, das strukturell dem Interaktionselement des Dienstes entspricht, aber noch über spezielle Anpassungen und zusätzliche Verweise verfügen kann.

Der Dienstplan einer Benutzerstation ist einfacher aufgebaut, er enthält nur ein oder mehrere Dienste mit der jeweiligen Bedingung.

Das Interaktionselement im Dienstplan ist die Vorlage zur Visualisierung der Dienstschnittstelle gegenüber den Benutzern im Portal. Zur Laufzeit wird das vom Dienst bei der Anforderung gelieferte Interaktionselement, welches die Bindungen an konkrete Werte

enthält, mit Hilfe des im Dienstplan definierten Interaktionselementes dargestellt. Bei Verweisen auf Stationen werden automatisch nur diejenigen Verweise dargestellt, die auf solche Stationen verweisen, deren im Dienstplan festgelegte Bedingung erfüllt ist. Die Bedingung im Dienstplan legt also fest, unter welchen Umständen zu einer Station navigiert werden kann.

4.5.2 Fahrplan

Der Fahrplan ist ein Verzeichnis der möglichen Folgestationen, d.h. der möglichen Transporte, die von dieser Station aus durchgeführt werden können. Ein Eintrag im Fahrplan besteht aus folgenden Elementen:

- Dem Namen der Zielstation.
- Dem Namen der Linie, über die die Zielstation erreicht wird.
- Dem Typ des Ereignisses, bei dem dieser Transport ausgeführt wird.
- Der Information, ob der Transport direkt durchgeführt wird, oder indirekt über ein Medium (Transportmittel) erfolgt. Bei indirekten Transporten besteht die Information aus einer geeigneten Identifikation des Transportmittels und -weges.

In der folgenden Tabelle werden die verschiedenen Ereignistypen und ihre Bedeutung aufgeführt:

Ereignistyp	**Bedeutung**
Erfolg	Transport findet statt, wenn die Anforderung/Ausführung des Dienstes erfolgreich war.
Fehler	Transport findet statt, wenn die Anforderung/Ausführung des Dienstes nicht erfolgreich war.
Warnung	Transport findet statt, wenn es bei der Ausführung des Dienstes Warnungen gegeben hat.
Benutzerwahl	Transport findet statt, wenn der Benutzer den entsprechenden Verweis ausgewählt hat.

Die Ereignistypen Erfolg, Fehler und Warnung können im Fahrplan jeder Station vorkommen. Ereignisse des Typs Benutzerwahl können nur bei Anforderungsstationen vorkommen, da nur diese Stationen für den Benutzer sichtbar sind und somit die Auswahl eines Verweises ermöglichen.

Zu jedem Eintrag mit den Ereignistyp Benutzerwahl gibt es einen im Interaktionselement des Dienstplans aufgeführten Verweis.

Es kann mehrere Einträge im Fahrplan geben, die den gleichen Ereignistyp verwenden. Bei Ereignissen der Typen Erfolg, Fehler und Warnung werden alle Transporte durchgeführt, bei Ereignissen des Typs Benutzerwahl wird lediglich derjenige Transport durchgeführt, der durch den ausgewählten Verweis festgelegt wurde.

Ein Transport kann direkt oder indirekt sein. Direkte Transporte leiten den Benutzer unmittelbar an die angegebene Station weiter. Bei indirekten Transporten wird ein Transportmittel verwendet, das dem Portal zur Verfügung steht. Ein derartiges Transportmittel könnte z.B. ein Workflow- oder Groupware-System sein. Ein typischer Anwendungsfall ist die Benachrichtigung eines Administrators, falls die Ausführung eines Dienstes fehlerhaft war. Bei einem Workflow-System erhält im Fehlerfall der Administrator dann ein Item in seiner Inbox, dass z.B. eine Beschreibung des Fehlers beinhaltet und ihn zur Anforderungsstation mit dem entsprechenden Kontext führt. Für eine nähere Diskussion von Workflow-Systemen und ihrer Arbeitsweise siehe z.B. [Jab97].

4.5.3 Ableitung der Benutzersicht aus der Designersicht

Wie bereits weiter oben erwähnt, gibt es auf Stationen die Benutzer- und die Designersicht. Dabei ist die Benutzersicht eine funktionale Vereinfachung der Designersicht. Intuitiv ausgedrückt, wird aus einer sichtbaren Station eine Benutzerstation, indem ihr die Dienste und Transporte der mit ihr verbundenen unsichtbaren Stationen hinzugefügt werden. Für unsichtbare Stationen gibt es keine Benutzerstationen.

Folgende Handlungsanweisung beschreibt, wie aus einer Station S die entsprechende Benutzerstation BS abgeleitet werden kann:

- Wenn S eine Anforderungsstation ist:
  - Erzeuge im Dienstplan von BS einen Eintrag mit dem Dienst und der Bedingung von S.
  - Für jeden Transport t des Fahrplans:
    - Wenn t auf eine Anforderungsstation verweist, übernimm t in den Fahrplan von BS.
    - Wenn t auf eine Ausführungsstation AS verweist, berechne die Menge W aller möglichen Wege (Folgen von Stationen), die von AS aus direkt oder nur über andere Ausführungsstationen zu einer Anforderungsstation führen. Für jede Station TS, die auf einem Weg der Menge W liegt, erzeuge einen neuen Eintrag im Fahrplan von BS, der die gleichen Informationen wie t enthält, nur dass TS statt AS als Zielstation angegeben ist. Ist der an TS ausgeführte Dienst noch nicht im Dienstplan von BS mit der angegebenen Bedingung aufgeführt, so ergänze diesen.
- Ist S eine Ausführungsstation, so gibt es zu S keine Benutzerstation.

4.6 Linien

Dienste in Unternehmensportalen lassen sich sehr häufig zu logischen Gruppen zusammenfassen. Wie in Kapitel 2.2.5 beschrieben, geschieht dies z.B. bei SAP Portals durch

die Worksets, mit denen unter bestimmten Aspekten zusammengehörige iViews gruppiert werden. Eine andere Gruppierung könnte sich beispielsweise an den zu manipulierenden Objekten orientieren, so wie dies in Kapitel 5.2.4 geschieht.

Das Konzept der Linien in EPM nimmt die Idee der Gruppierung von Diensten auf. Jeder Gruppe von Diensten wird eine Linie mit einer eigenen Farbe zugeordnet. Wird nun im Fahrplan einer Station ein Transport eingetragen, so geschieht dieser über diejenige Linie, zu der der an der Zielstation angebotene Dienst gehört. Linien stellen also eine logische Gruppierung von Diensten dar. Die einzelnen Strecken zwischen den Stationen sind die Navigationsmöglichkeiten im Portal.

4.7 Netzpläne

Netzpläne sind das zentrale, graphische Verzeichnis für die verschiedenen EPM-Elemente. Netzpläne enthalten die verfügbaren Bedingungen, Dienste, Stationen und Linien eines Portals. Abbildung 4.5 stellt den Netzplan und alle weiteren Klassen des EPM-Modells und ihre Beziehungen in UML als Klassendiagramm dar.

.

Ein Netzplan stellt folgende Konzepte grafisch dar:

- Stationen
- Bedingungen an Stationen
- Einträge in Fahrplänen (Navigationsmöglichkeiten im Portal)

Die Visualisierung orientiert sich dabei an den in Kapitel 4.1.2 aufgeführten Darstellungsprinzipien. Die Prinzipien für Netzpläne sind:

- Ausschließliche Nutzung von Horizontalen, Vertikalen und 45°-Diagonalen.
- Verwendung von „Ticks“ zur Kennzeichnung von Stationen, deren Fahrplan höchstens zwei Einträge hat und diese Einträge dieselbe EPM-Linie verwenden.
- Stationen, die nicht als Ticks dargestellt werden, werden durch kleine Kreise symbolisiert.
- Unterscheidung der EPM-Linien durch Farbkodierung.
- Darstellung eines Eintrags im Fahrplan einer Station durch das Ziehen einer Verbindungslinie zur Zielstation. Diese Verbindungslinie wird Strecke genannt.

  Eine Strecke kann in Streckenabschnitte aufgeteilt sein, jeder Streckenabschnitt ist eine Gerade. Eine Strecke als ganzes muss deshalb nicht gerade verlaufen.

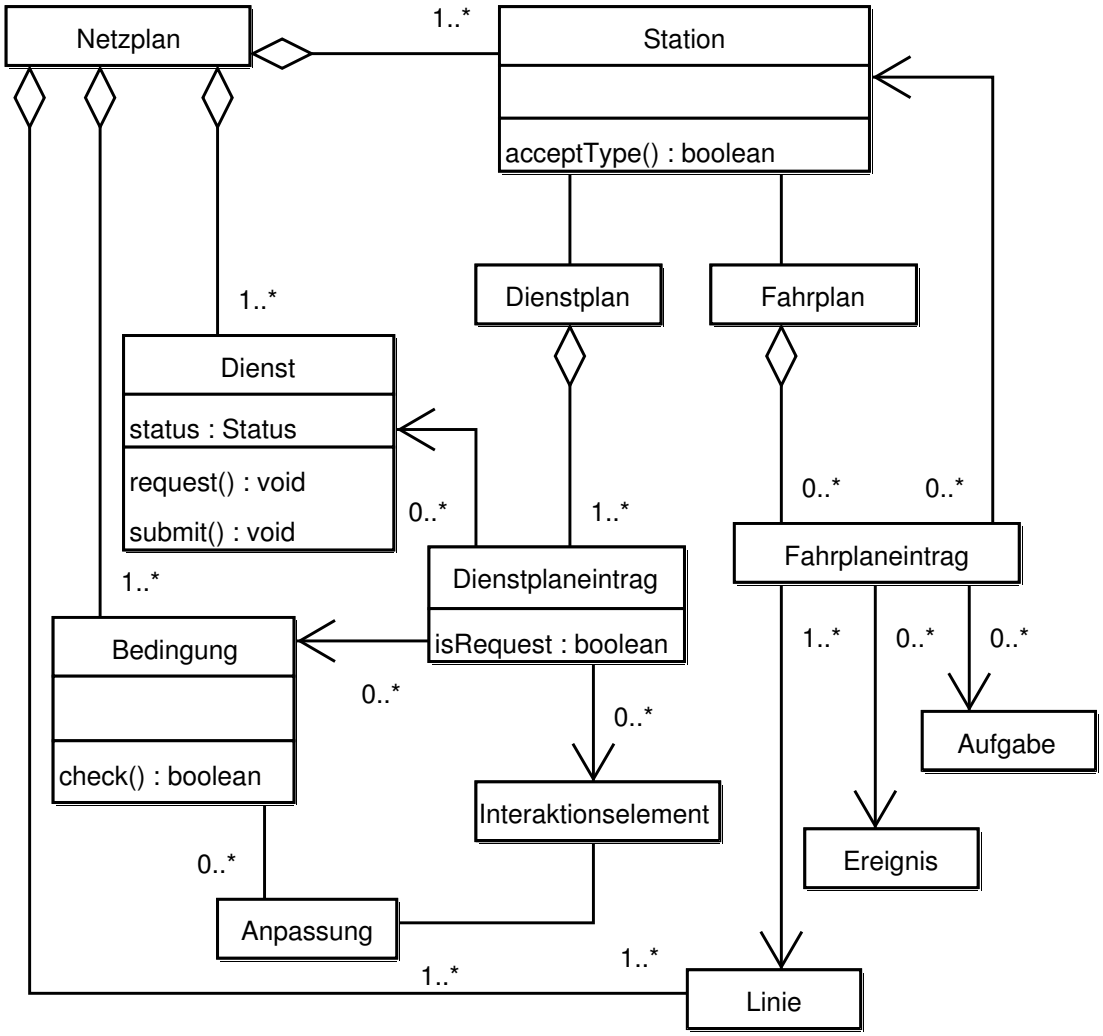

Abbildung 4.5: Der Netzplan als zentrale Klasse des EPM-Modells.

- Darstellung der „Fahrtrichtung“ zwischen zwei Stationen durch einen Pfeil auf der Strecke, wenn es keinen entsprechenden „Gegeneintrag“ im Fahrplan der Zielstation gibt, d.h. wenn nicht zwischen zwei Stationen auf derselben EPM-Linie „hin und her“-navigiert werden kann.
- Wird eine Strecke, oder ein Streckenabschnitt von zwei unterschiedlichen Linien verwendet, so wird die Strecke klein gestrichelt dargestellt, damit die unterschiedlichen Farben beide Linien abwechselnd zu sehen sind.
- Indirekte Transporte werden nicht durch eine durchgezogene, sondern eine groß gestrichelte Linie veranschaulicht.
- Die im Dienstplan einer Station angegebene Bedingung wird durch einen farbigen

Kreis dargestellt, der als Hintergrund für die Station dient.

- Der Name solcher Stationen, die einen kontextfreien Dienst anbieten, d.h. der Name solcher Stationen, zu denen mit beliebigen Kontext navigiert werden kann, wird fett und kursiv wiedergegeben.

Für jedes mit Hilfe der Enterprise Portal Maps modellierte Unternehmensportal gibt es zwei Ansichten des zugehörigen Netzplans:

- Die Designersicht mit allen Stationen.
- Die Benutzersicht mit allen Benutzerstationen.

Entscheidend ist, dass es sich dabei tatsächlich nur um Sichten des einen Modells handelt. Aus der Designersicht des Netzplans kann die Benutzersicht durch Erzeugung der Benutzerstationen aus den Stationen (vgl. Kapitel 4.5.3 gewonnen werden.

Abbildung 4.6 zeigt einen Netzplan in Designersicht, der 42 Stationen, 107 Einträge in Fahrpläne, 5 Linien und 12 Bedingungen wiedergibt.

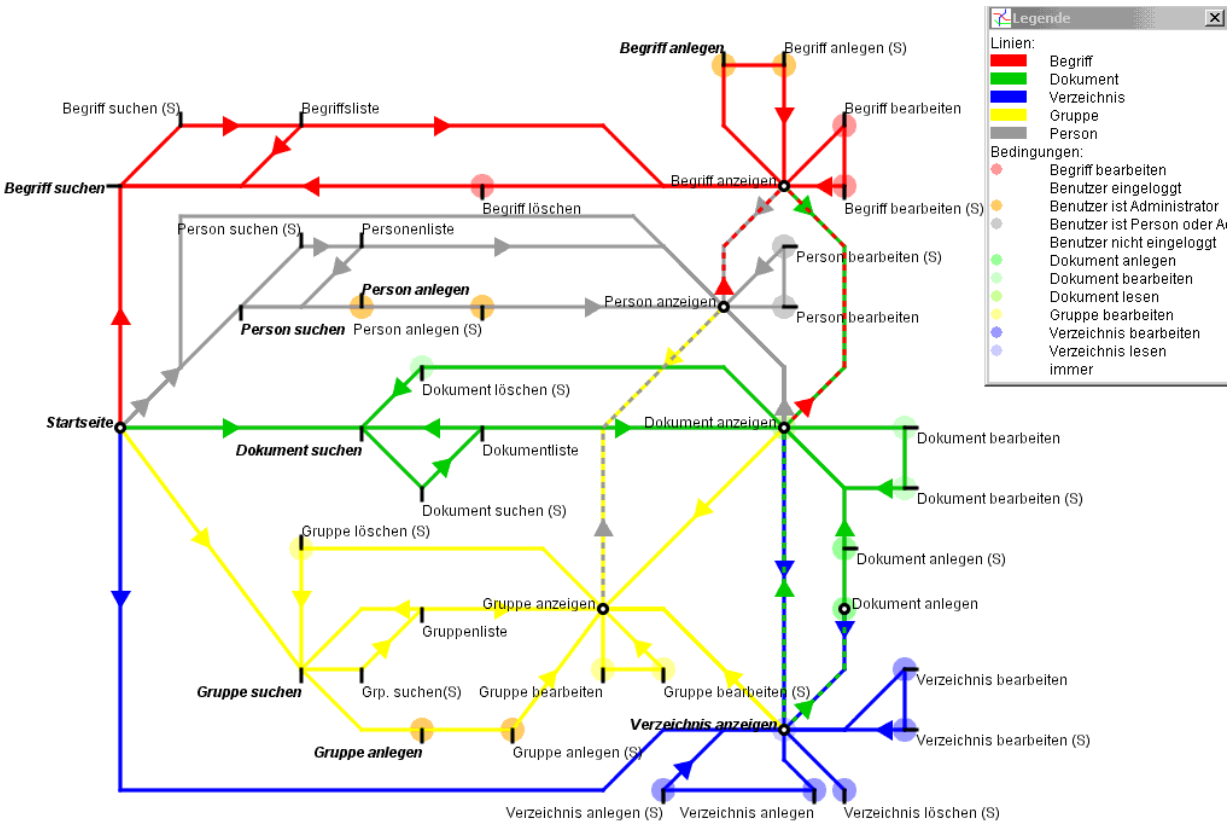

Abbildung 4.6: Netzplan mit 42 Stationen und 107 Einträgen in den Fahrplänen.

4.8 EPM als Sprache

Das Modell der Enterprise Portal Maps lässt sich auch als Sprache betrachten, deren Wörter, also die konkreten Enterprise Portal Maps, von einem Interpreter, der EPM-Laufzeitumgebung, verstanden werden.

Die Syntax der Sprache EPM lässt sich mit folgender Grammatik[2] beschreiben:

$$
\begin{array}{lll}
EPM = (V, \Sigma, P, Netzplan) & \text{mit} & \\
\Sigma = \{\ \text{'('} , \text{','} , \text{')'} , \text{'a'...'z'} , \text{'0'...'9'}\ \} & & \\
P = \{ & Netzplan & \longrightarrow (\{Station\}^+, \{Linie\}^+, \{Dienst\}^+, \{Bedingung\}^+) \\
 & Station & \longrightarrow (Name, \{Dienstplaneintrag\}^+, \{Fahrplaneintrag\}^+) \\
 & Linie & \longrightarrow (Name, Farbwert) \\
 & Dienst & \longrightarrow (Name, Interaktionselement, Status, Kontexttyp, \\
 & & \quad Kontexttyp, Kontexttyp) \\
 & Bedingung & \longrightarrow (Name) \\
 & Name & \longrightarrow \{\text{a}\|...\|\text{z}\|0\|...\|9\}^+ \\
 & Dienstplaneintrag & \longrightarrow (Bedingung, Dienst, Interaktionselement) \\
 & Fahrplaneintrag & \longrightarrow (Ereignis, Station, Linie, Transportmittel) \\
 & Interaktionselement & \longrightarrow (Name, Bezeichnung, \{Anpassung\}^+, Spez) \\
 & Ereignis & \longrightarrow \text{Erfolg} \parallel \text{Warnung} \parallel \text{Fehler} \parallel \text{Benutzerwahl} \\
 & Transportmittel & \longrightarrow \{\text{a}\|...\|\text{z}\|0\|...\|9\}^+ \\
 & Bezeichnung & \longrightarrow \{\text{a}\|...\|\text{z}\|0\|...\|9\}^+ \\
 & Anpassung & \longrightarrow (Bedingung, Bool, Bool, Bool) \\
 & Spez & \longrightarrow Einzelwert \| Selektion \| Verweis \| Liste \\
 & Bool & \longrightarrow \text{wahr} \parallel \text{falsch} \\
 & Einzelwert & \longrightarrow Typbezeichnung \\
 & Selektion & \longrightarrow (Interaktionselement\{Interaktionselement\}) \\
 & Verweis & \longrightarrow Bool, Station, URL \\
 & Liste & \longrightarrow Bool, \{Interaktionselement\}^+, \{Liste\} \\
 & Typbezeichnung & \longrightarrow \{\text{a}\|...\|\text{z}\|0\|...\|9\}^+ \\
 & URL & \longrightarrow \{\text{a}\|...\|\text{z}\|0\|...\|9\}^+ \\
 & \} & \\
V = \text{ergibt sich aus } P & &
\end{array}
$$

Ein Wort der Sprache EPM beschreibt ein konkretes Portal. Diese Beschreibung kann dann von verschiedenen Umgebungen interpretiert und angewendet werden. Wie in Kapitel 5 beschrieben werden wird, werden Wörter der Sprache, bzw. leichte Modifikationen hiervon, zwischen dem Definitionswerkzeug, dem infoAsset Broker, dem SAP Web Application Server und dem EPM-Applet ausgetauscht. Jede dieser Umgebungen benötigt neben dem eigentlichen Wort natürlich noch das Wissen, wie es mit der Information umgehen soll.

[2]Definition einer Grammatik gemäß [Schoe92, Seiten 13 und 26].

4.9 Auswirkungen auf Komponenten der Referenzarchitektur

In Bezug auf die in Kapitel 3.3 beschriebene Referenzarchitektur werden folgende Komponenten durch das EPM-Modell beeinflusst:

- Die uniforme Schnittstelle der integrierten Dienste im Portalsystem wird durch das EPM-Modell mit seinen Diensten und Interaktionselementen festgelegt.
- Die Dienstintegration muss aus einem Kontext die für den betreffenden Dienst notwendigen Daten erzeugen können.
- Der Inhalt der allgemeinen Layouts wird durch die Stationen mit den zugehörigen Personalisierungen und Navigationsmöglichkeiten bestimmt.
- Die Zugriffsrechte erhalten über das Konzept der Bedingungen eine uniforme Schnittstelle für die Personalisierung der allgemeinen Layouts.
- Die Benutzerverwaltung muss eine einheitliche Schnittstelle auf die Daten eines Benutzers, zumindest aber die verschiedenen Kennungen eines Benutzers in den verschiedenen Systemen, liefern.

Da die Definition eines konkreten Portals mit Hilfe des EPM-Modells beschrieben wird, ändert sich insbesondere die Struktur des Controllers, der in zwei Teile aufgeteilt wird: Die Enterprise Portal Map, die das Portal beschreibt und die EPM-Laufzeitumgebung, die eine gegebene Enterprise Portal Map interpretiert und entsprechend aufbereitete Seiten an die Kommunikationssysteme weiter gibt, vgl. Abbildung 4.7 im Gegensatz zu Abbildung 3.4 (Seite 68).

4.10 Zusammenfassung

In diesem Kapitel wurde das Modell der Enterprise Portal Maps, kurz EPM-Modell genannt, vorgestellt. Der EPM-Ansatz lehnt sich an die Begrifflichkeit des öffentlichen Personennahverkehrs zur Modellierung von Portalen und Diensten an, indem ein Portal mit den enthaltenen Diensten als eine Menge von Stationen angesehen wird, die über Linien miteinander verknüpft sind. Mit Hilfe dieser Sichtweise gewinnt man eine kanonische graphische Darstellungsart für Portale und Dienste, aus der der Inhalt und die Struktur eines Portals hervorgehen.

Zunächst wurden allgemeine Karten als graphischer Kommunikationsprozess erörtert, wobei die für Kommunikationsprozesse typischen Probleme auftauchen. Für die sehr guten Kommunikationseigenschaften der im ÖPNV üblichen Netzpläne sprechen hingegen zwei wesentliche Faktoren: Erstens konzentrieren sich Netzpläne auf die Kommunikation der

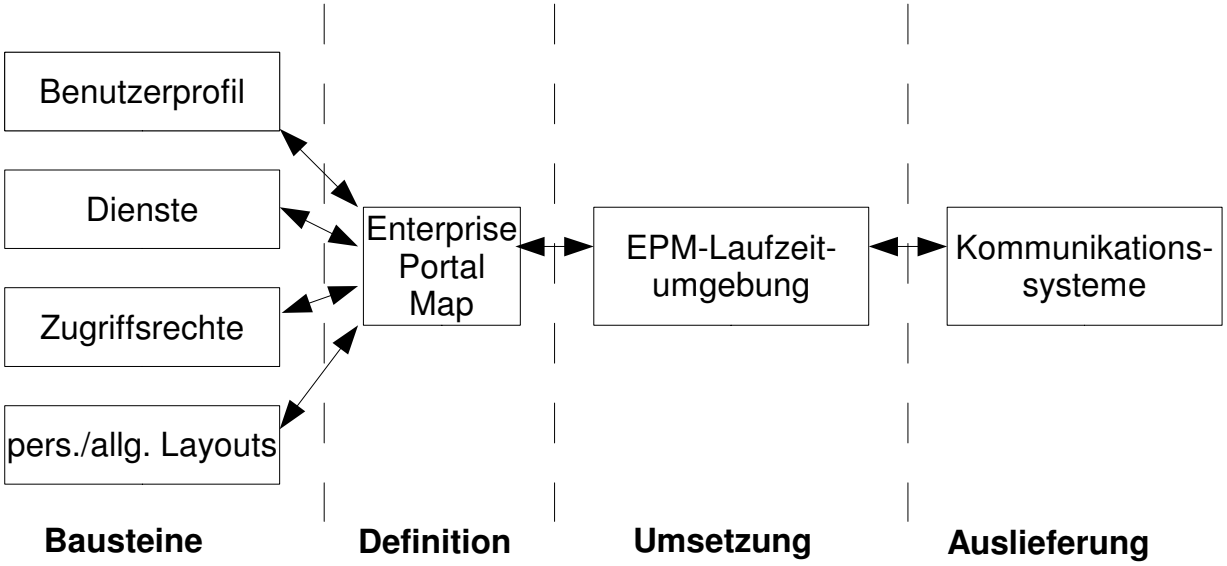

Abbildung 4.7: Die Enterprise Portal Map als zentrale Stelle für die Definition eines Portals.

Stationen und der möglichen Stationsfolgen und zweitens ist die von H.C. Beck erfundene Notation für Netzpläne ein de-facto Standard, so dass viele Menschen das Lesen der Netzpläne und das Verstehen der enthaltenen Informationen gewohnt sind.

Der Erfolg der Notation nach Beck beruht auf mehreren Design-Prinzipien, vor allem aber auf der ausschließlichen Verwendung von horizontalen, vertikalen und 45°-diagonalen Linien und der Verwendung von sog. Ticks zur Kennzeichnung von Stationen, die nur von einer Linie angefahren werden.

Im Rahmen der Enterprise Portal Maps werden Portale und Dienste wie folgt modelliert: Ein Portal besteht aus Stationen, die entweder einen Dienst anfordern oder ausführen. Der Besuch einer Station kann an eine Bedingung geknüpft sein, die erfüllt sein muss, damit ein Besuch, ggf. im aktuellen Kontext, zulässig ist. Die Schnittstelle der Dienste wird durch Interaktionselemente beschrieben und kann mit Hilfe von Bedingungen personalisiert werden. Eine Sonderstellung bei den Interaktionselementen nehmen die Verweise ein, die einen Kontext transportieren können und über die ein Benutzer von einer Station zu einer anderen Station gelangt. Dienste erhalten bei ihrer Anforderung den aktuellen Kontext und den aktuellen Benutzer als Parameter übergeben, damit sie sich initialisieren und ggf. personalisieren können. Dienste können den aktuellen Kontext durch ihre Anforderung oder Ausführung verändern.

Eine Station ist sichtbar, wenn sie einen Dienst anfordert, sie ist unsichtbar, wenn sie einen Dienst ausführt. Sichtbare Stationen heißen deshalb auch Anforderungsstationen, unsichtbare Stationen werden auch Aufführungsstationen genannt. Im Dienstplan einer Station sind alle notwendigen Details für den Aufruf des Dienstes enthalten. Der Fahrplan bestimmt, zu welcher Folgestation bei welchen Ereignissen nach der Anforderung oder Ausführung des an der Station angebotenen Dienstes navigiert werden soll. Jeder Eintrag

eines Fahrplans definiert eine Strecke zwischen zwei Stationen. Jede Strecke gehört zu genau einer Linie, nämlich zu derjenigen Linie, zu der an der Zielstation angebotene Dienst gehört.

Die Netzpläne stellen die eigentlichen Enterprise Portal Maps dar. Sie verzeichnen die angebotenen Stationen, ihre Bedingungen und Strecken und lehnen sich an die Design-Prinzipien Becks an. Jede Enterprise Portal Map verfügt über zwei Sichten: Die Designersicht, in der alle Stationen verzeichnet sind und die Benutzersicht, bei der es zu jeder Station eine für den Benutzer des Portals sichtbare Seite gibt.

Das für das Modell der Enterprise Portal Maps kann eine formale Sprache angegeben werden, die die Syntax der „wohlgeformten“ konkreten Enterprise Portal Maps beschreibt und mit deren mit Hilfe Wörter der Sprache EPM zwischen verschiedenen Programmen ausgetauscht werden können.

Das EPM-Modell wirkt sich auf verschiedenen Ebenen der Referenzarchitektur für Portalsysteme aus, u.a. die Standardisierung einiger Schnittstellen und die Forderung eine durchgängigen Identifizierbarkeit von Benutzern und Kontexten. Der zentrale, monolithische Controller eines Portalsystems wird durch einen dynamischen Interpreter der EPM-Sprache ersetzt.

Im folgenden Kapitel wird die Implementierung des Modells der Enterprise Portal Maps für zwei sehr unterschiedliche Portalsysteme und ihrer Dienste beschrieben. Bei dem ersten System handelt es sich um den in Kapitel 2.2.6 beschriebenen infoAsset Broker. Aufbauend hierauf wird eine Enterprise Portal Map für das Wissensmanagement in Unternehmen vorgestellt. Das zweite System ist der SAP Web-Application Server, der in Kapitel 2.2.5 beschrieben wurde. Als Anwendung wird hier eine Enterprise Portal Map entwickelt, die die BAPIs zum SAP-üblichen Schulungsbeispiel zur Buchung von Flügen als Stationen, Dienste und Bedingungen umsetzt.

Kapitel 5

Implementierungen des Modells der Enterprise Portal Maps

In diesem Kapitel wird die praktische Anwendung des im vorigen Kapitel beschriebenen Modells der Enterprise Portal Maps erläutert. Für die Erprobung des Modells wurde zunächst ein grafischer Editor implementiert, der alle Elemente des EPM-Modells unterstützt und mit dem beliebige EPM-Karten erstellt werden können. Für die zwei Portalsysteme infoAsset Broker und SAP Web Application Server wurden EPM-Laufzeitumgebungen erstellt, die die direkte Ausführung der durch den Editor definierten Enterprise Portal Maps erlaubt. Eine zusätzliche Anpassung der Benutzungsoberfläche bringt die dem konkreten Portal zugrunde liegende Enterprise Portal Map unmittelbar zu den Benutzern des Portals. Durch das nahtlose Zusammenspiel von Editor, Laufzeitumgebungen und Benutzungsoberfläche wird das Potential der Enterprise Portal Maps aufgezeigt.

5.1 Ein Definitionswerkzeug für Enterprise Portal Maps

Das Definitionswerkzeug für Enterprise Portal Maps, im Folgenden kurz EPM-Editor genannt, setzt sich aus drei Komponenten zusammen:

- Das *EPM-Modell*: Hierunter fallen die Klassen für Netzpläne, Stationen, Linien, Dienste, Interaktionselemente und Bedingungen.

- Der *graphische Editor*: Der graphische Editor stellt eine mit Java-Swing erstellte Oberfläche zur Manipulation des EPM-Modells bereit.

- Die *Schnittstelle zur EPM-Laufzeitumgebung*: Die Schnittstelle erzeugt unmittelbar aus dem EPM-Modell alle für die gewünschte EPM-Laufzeitumgebung notwendigen Informationen und Artefakte. Die Schnittstelle wurde zweimal implementiert: Eine

Schnittstelle für den infoAsset Broker und eine Schnittstelle für den SAP Web Application Server (Version 6.10).

Im Folgenden werden die einzelnen Komponente näher betrachtet.

5.1.1 Das EPM-Modell

Das implementierte EPM-Modell lehnt sich stark an das EPM-Modell des vorigen Kapitels an. Entsprechend gibt es die Klassen `Map`, `Station`, `Line`, `Service`, `InteractionElement`, `Task` und `Condition`, sowie einige Hilfsklassen und -schnittstellen. Das UML-Klassendiagramm entspricht dem in der Abbildung 4.5 (Seite 87) wiedergegebenen, mit Ausnahme der Bezeichnungen.

Die Klasse `Map` implementiert einen Netzplan. Sie kennt alle verzeichneten Stationen sowie die (x,y)-Koordinaten der Stationen auf dem Netzplan. Als „Behälter“ kennt sie auch alle Linien mit Farben, Dienste, Aufgaben und Bedingungen mit Farben. Die Klasse `Map` stellt Zugriffsfunktionen auf die jeweiligen Objekte bereit, so dass sie hinzugefügt, geändert oder entfernt werden können. Neben diesen mehr verwaltenden Aspekten bietet die Klasse als zentrale Methode zur Darstellung die Methode `draw` an, die eine graphische Darstellung des Netzplans erlaubt. Das Zeichnen eines Netzplans erfolgt gemäß nachstehendem Algorithmus:

```
Algorithmus draw
In: eine Fläche
Out: ein auf die Fläche gemalter Netzplan
Begin
  (* stelle Bedingungen dar *)
  if (Anzeige der Bedinungsfarben gewünscht)
    for all stations s do
      zeichne einen Kreis an der Stelle der Station s mit der Farbe
      der dem Dienst zugeordneten Bedingung.
    rof
  fi
  (* zeichne die Strecken zwischen den Stationen *)
  berechne die zweidimensionale Matrix fromTo, in der an der Stelle (i,j)
    die Menge aller Fahrplaneinträge steht, die von der Station si zu der
    Station sj führen.
  for all stations si do
    for all stations sj do
      for all transports t in fromTo[i,j] do
        boolean oneway := es gibt keinen Fahrplaneintrag von sj nach si
                          mit der gleichen Linie wie t.
        if (oneway and i>j)
          abort and continue with next t.
```

```
        fi
        sei lineSegments die Menge aller Streckenabschnitte für t.
        for all line segments l in lineSegments do
          if (l ist das erste Segment and oneway)
            zeichne einen Pfeil mit der Farbe der Linie aus t
              entsprechend dem Segment l.
          else
            zeichne eine Gerade mit der Farbe der Linie aus t
              entsprechend dem Segment l.
          fi
        rof
      rof
    rof
  rof
  (* zeichne die Stationen *)
  for all stations s do
    if (Fahrplan der Station s hat weniger als 2 Einträge oder genau 2
        Einträge, die die gleiche Linie verwenden)
      zeichne einen Tick an der Position der Station s.
    else
      zeichne einen einfachen Kreis an der Position der Station s.
    fi
    if (Dienst an der Station s kann ohne Kontext aufgerufen werden)
      zeichne Namen der Station s fett und kursiv.
    else
      zeichne Namen der Station s.
    fi
  rof
End
```

Die theoretische Laufzeit des Algorithmus wird von der Anzahl der Streckenabschnitte dominiert, da jede Strecke theoretisch in beliebig viele Streckenabschnitte unterteilt sein kann. Die Anzahl sowohl der Linien, als auch der Bedingungen kann gegen die Anzahl der Stationen abgeschätzt werden. In der praktischen Anwendung wird die Laufzeit des Algorithmus jedoch von der Anzahl der Stationen bestimmt, da Strecken selten in mehr als 5 Streckenabschnitte unterteilt sind und – wie gesagt – die Anzahl der Linien und der Bedingungen jeweils kleiner oder gleich der Anzahl der Stationen ist.

Die Darstellung lässt sich mit Hilfe von Attributen der Klasse Map in gewissen Grenzen variieren. So kann beispielsweise ein Hintergrundgitter erzeugt werden, dass bei der Positionierung der Stationen und Strecken bzw. Streckenabschnitte hilfreich ist. Auch die farbige Darstellung der Bedingungen kann unterbunden werden, dies ist insbesondere bei der Benutzersicht eines Netzplans sinnvoll. Weiterhin können graphische Eigenschaften, wie etwa die Dicke der Linien oder das Antialiasing beeinflusst werden.

Eine weitere wichtige Methode der Klasse Map stellt die Methode `getUsersMap` dar, die gemäß des in Kapitel 4.5.3 geschilderten Algorithmus aus der Designersicht die Benutzersicht ableitet. Die Benutzersicht wird als neue Instanz der Klasse Map erzeugt, wodurch sich eine Vereinfachung einiger Algorithmen des graphischen Editors ergibt.

Die Interaktionselemente werden durch eine eng an die in Abbildung 4.3 (Seite 78) dargestellte Objekthierarchie abgebildet. Dabei wurden folgende Klassen implementiert:

- `InteractionElement`: Als abstrakte Klasse werden hier die Grundeigenschaften aller Interaktionselemente implementiert. Zentral sind der Name, eine Beschriftung (Label) und die möglichen Anpassungen, die über entsprechende Methoden definiert, verändert und abgefragt werden können. Zusätzlich kann zu jedem InteractionElement eine Nachricht für den Benutzer existieren, die z.B. auf mögliche Fehler bei der Eingabe hinweisen kann. Die abstrakte Methode `getIEByName(String lookfor)` liefert ein Interaktionselement zurück, das den gesuchten Namen trägt, wenn ein solches Element über das aktuelle Interaktionselement erreichbar ist. Über diese Methode kann gezielt ein Interaktionselement, das in einer Liste enthalten ist, angesprochen werden.

- `IESingleValue`: Diese Klasse repräsentiert einen einfachen Wert, der eine Typbezeichnung hat. Über diese Typbezeichnung kann der mögliche Wertebereich bestimmt werden. Die Methode `getIEByName` prüft auf Namensgleichheit und gibt ggf. eine Referenz auf das eigene Objekt zurück.

- `IESelection`: Bei einer Selektion kann aus einer festen Wertemenge ein einzelner Wert oder eine Teilmenge ausgewählt werden. Die Werte einer Menge müssen vom gleichen Typ sein. Die Methode `getIEByName` prüft auf Namensgleichheit und gibt ggf. eine Referenz auf die Selektion zurück.

- `IELink`: Diese Klasse repräsentiert einen Verweis. Der Verweis kann extern sein, d.h. dass der Wert nur einen Kontext identifiziert und ggf. die Ziel-URL angegeben ist, er kann zusätzlich aber auch Informationen über die Zielstation und die Linie enthalten. Über ein Statusattribut wird festgelegt, ob die als Wert gespeicherte Kontextinformation bei der Navigation mit übergeben werden soll oder nicht. Verweise können nicht änderbar sein. Die Methode `getIEByName` prüft auf Namensgleichheit und gibt ggf. eine Referenz auf den Verweis zurück.

- `IEList`: Eine `IEList` enthält eine Menge von beliebigen `InteractionElements`. Die Methode `getIEByName` prüft auf Namensgleichheit und gibt ggf. eine Referenz auf die Liste zurück. Ist die Liste selbst nicht das gesuchte Element, so werden nacheinander alle enthaltenen Interaktionselemente befragt und das Ergebnis zurückgeliefert.

Die Klasse `Service` ist die abstrakte Basis aller Dienste. Sie bietet im Wesentlichen die in Kapitel 4.4 beschriebenen Methoden an, wobei die Anforderungsmethode `request` statisch ist und als Parameter den Kontext und die Benutzerkennung erwartet. Diese statische

Methode dient als *Factory* für den konkreten Dienst und gibt entsprechend eine Instanz der Klasse Service zurück.

Die übrigen implementierten Klassen des EPM-Modells leiten sich unmittelbar aus den in Kapitel 4 beschriebenen EPM-Konzepten ab.

5.1.2 Der graphische Editor

Mit Hilfe des graphischen Editors können durch direkte Manipulation (vgl. 3.1) Netzpläne für den infoAsset Broker oder für den SAP Web Application Server erstellt oder geändert werden. Abbildung 5.1 zeigt einen Bildschirmausschnitt mit der Menüleiste, der Symbolleiste und der Kartenfläche in der Designer-Ansicht. Über die Menüs sind alle Funktionen des Editors erreichbar. Der Editor verfügt über verschiedene „Betriebs-Modi“: Zum einen unterscheiden sich die verfügbaren Befehle zur Manipulation des EPM-Modells je nach dem, ob die Designer-Sicht oder die Benutzer-Sicht aktiv sind. Zum anderen existieren die zwei Menüs „Broker“ und „SAP“, die die spezifischen Befehle für die Anbindung des infoAsset Brokers bzw. des SAP Web Application Servers zusammenfassen.

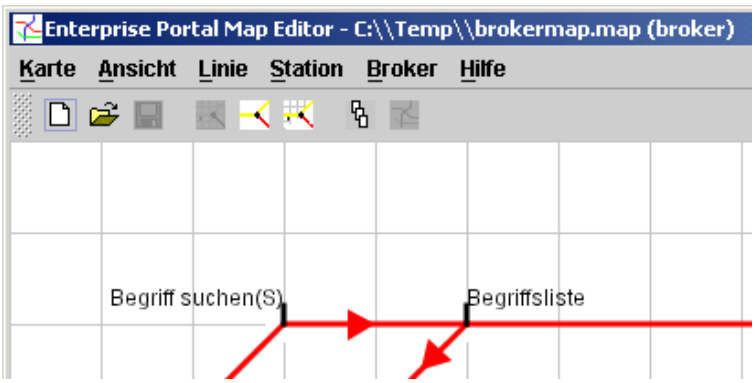

Abbildung 5.1: Bildschirmausschnitt mit Menüleiste, Symbolleiste und Kartenfläche.

Die Symbolleiste ermöglicht den schnellen Zugriff auf häufig benötigte Funktionen, wie z.B. die Umschaltung der Ansicht zwischen Designer und Benutzer oder die Generierung von HTML-Templates oder der Site Map für den infoAsset Broker bzw. den SAP Web Application Server.

Die Anordnung der Linien (Streckenabschnitte) und Stationen auf der Fläche kann durch einfaches „Drag-and-Drop“ verändert werden, wobei noch während des Ziehens die Änderungen sichtbar werden. Über die Kartenfläche sind auch mehrere Kontextmenüs der dargestellten Elemente erreichbar. Über Kontextmenüs lassen sich Strecken in Streckenabschnitte aufteilen, geteilte Streckenabschnitte wieder zusammenführen oder auch Eigenschaften von Stationen ändern.

Beispielhaft sei das Kontextmenü eines Streckenabschnitts angeführt, das in Abbildung 5.2 wiedergegeben ist. Ein Streckenabschnitt kann geteilt oder verbunden (zusammengeführt) werden, wodurch die Streckenführung zwischen zwei Stationen verändert wer-

den kann. Ein Streckenabschnitt kann gestrichelt oder durchgezogen dargestellt werden, je nach dem, ob auf der gleichen Position des Streckenabschnitts noch ein zweiter Streckenabschnitt einer anderen Linie vorhanden ist oder nicht. Die Überlagerung zweier (oder mehrerer) Streckenabschnitte entspricht der Nutzung ein und desselben Gleises im Bereich des öffentlichen Personennahverkehrs. Durch die Überlagerung von Streckenabschnitten kann das Aussehen eines Netzplans oft erheblich eleganter gestaltet werden, insbesondere weil Benutzer daran gewöhnt sind, dass Züge nicht „um die Ecke“ fahren, d.h. dass keine Verbindung zwischen zwei Stationen einen spitzen Winkel ($< 90°$) enthält. Zuletzt kann über das Kontextmenü noch ein Dialog zur Änderung der Eigenschaften der Linie, zu der der Streckenabschnitt gehört, aufgerufen werden.

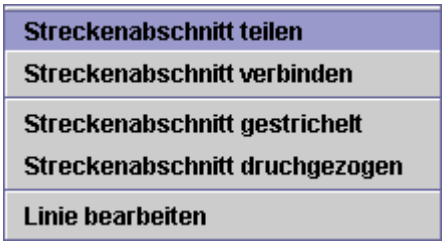

Abbildung 5.2: Kontextmenü eines Streckenabschnitts.

Die für die Definition einer Enterprise Portal Map am häufigsten verwendeten Dialoge sind der Dialog zur Änderung der Stationseigenschaften und der Dialog zur Änderung eines Interaktionselementes. In Abbildung 5.3 sind beide Dialoge vor dem Hintergrund der Kartenfläche abgebildet.

Über den Stationsdialog lässt sich der Name der Station, der auf der Karte angezeigt wird, und der technische Name, der nur für die Ablage der generierten infoAsset Broker Templates notwendig ist, ändern. Der Dialog setzt sich darüber hinaus aus drei untergeordneten Dialogen zusammen:

- Im ersten Dialog kann das Layout der Station beeinflusst werden, insbesondere die Ausrichtung des Ticks und die automatische Platzierung des Stationsnamens.
- Im dargestellten zweiten Dialog wird der an der Station angebotene Dienst spezifiziert. Hierzu müssen Bedingung, Dienst und die Art des Aufrufs des Dienstes ausgewählt werden.
- Der dritte Dialog behandelt den Fahrplan der Station. Wird der spezifizierte Dienst angefordert, so enthält der Fahrplan ausschließlich Einträge mit dem Ereignis „Benutzerwahl“ und für jeden Eintrag gibt es ein korrespondierendes Interaktionselement vom Typ „Verweis“, das auf die Zielstation verweist. Der Fahrplan kann in diesem Falle nur indirekt über die Interaktionselemente verändert werden. Wird der spezifizierte Dienst ausgeführt, dann kann der Fahrplan über den Dialog verändert werden, um zu definieren, wie sich das System in Abhängigkeit des Ergebnisses der Ausführung des Dienstes verhalten soll.

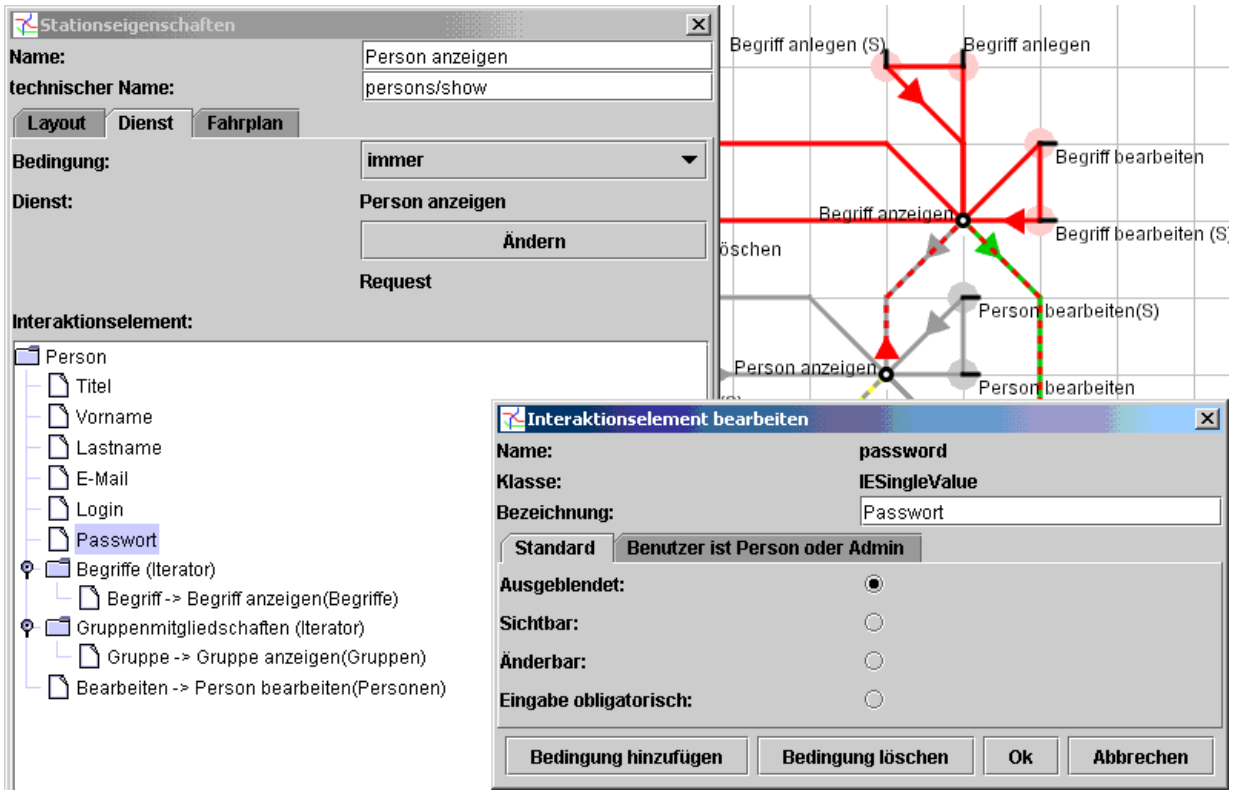

Abbildung 5.3: Dialoge zur Änderung der Stationseigenschaften und eines Interaktionselementes.

Über den ebenfalls in Abbildung 5.3 abgebildeten Dialog „Interaktionselement bearbeiten“ werden die Anpassungen festgelegt. Für jede Anpassung existiert ein eigener Reiter mit dem Namen der Bedingung, bei deren Erfüllung das Interaktionselement ausgeblendet, sichtbar, änderbar oder obligatorisch sein soll. Im dargestellten Beispiel wird das Passwort eines Benutzers an der Station „Person anzeigen“ nur dann angezeigt, wenn die Bedingung „Benutzer ist Person oder Admin“ zum Anforderungszeitpunkt erfüllt ist.

5.1.3 Die Schnittstelle zur EPM-Laufzeitumgebung

Die Schnittstelle zwischen dem EPM-Modell und der EPM-Laufzeitumgebung für ein Portalsystem beinhaltet zwei Aspekte: Der erste Aspekt ist der *Import* der vorhanden benutzerorientierten Dienste oder Funktionen, der Berechtigungen und Aufgaben in den graphischen Editor als EPM-Dienste, Bedingungen und Transportmittel. Der zweite *Aspekt* ist der Export der erzeugten Enterprise Portal Maps für die EPM-Laufzeitkomponente, die die Enterprise Portal Map auf dem jeweiligen Portalsystem ausführen soll.

Für den Import existieren drei Dateien, die die zu importierenden EPM-Dienste, die Bedingungen und die Aufgaben benennen und eine entsprechende Java-Klasse zuordnen. Die

Dateien sind einfache Java-Property-Files, die von den jeweiligen Schnittstellenobjekten gelesen und interpretiert werden müssen. Bei dem Import von Diensten, Bedingungen und Aufgaben kommt es nicht darauf an, dass die dann erzeugten Java-Objekte auch tatsächlich tun, was ihr Name impliziert. Sie dienen lediglich als Platzhalter zur Definition bzw. Manipulation der Enterprise Portal Map. Der Editor ruft keine Bedingungen oder Aufgaben auf, die Dienst-Objekte werden lediglich nach ihrem speziellen Interaktionselement und den Typinformationen befragt.

Der Export setzt sich aus der Generierung von Informationen für die EPM-Laufzeitumgebung und ggf. die Benutzungsoberfläche zusammen. Die Informationen für die Laufzeitumgebung werden unmittelbar aus der Enterprise Portal Map in der Designer-Sicht gewonnen, da nur hier alle benötigten Informationen vorliegen. Welche Informationen genau exportiert werden, hängt stark von der jeweiligen Portalsystem ab. Für die Benutzungsoberfläche wird die Benutzersicht als Graphik für die Site-Map erzeugt und es werden Informationen über die Stationen in einer speziellen Art für das EPM-Applet, das später beschrieben wird, abgelegt.

5.2 EPM für den infoAsset Broker

Für die Erprobung des Konzepts der Enterprise Portal Maps wurde zunächst der infoAsset Broker mit seiner Auslieferungskomponente und seinen Diensten ausgewählt. Folgende Aspekte waren dabei ausschlaggebend:

- Der infoAsset Broker stellt für das Wissensmanagement reale, konsistente, nicht triviale und gut dokumentierte Prozesse und Dienste zur Verfügung, die ihre Eignung in der industriellen Praxis täglich beweisen. Dabei ist auch das Berechtigungskonzept an den tatsächlichen Bedürfnissen der Anwender und der anwenden Unternehmen ausgerichtet.

- Durch die konsequente Schichtenarchitektur besteht eine echte Trennung zwischen der Auslieferungskomponente und den Diensten. Dies ermöglicht die Abstraktion der Ergebnisse vom infoAsset Broker als Plattform und die Übertragung auch auf andere Architekturen für Portale und Dienste, wie z.B. die mySAP Technologie.

- Bei der Architektur des infoAsset Brokers war von Beginn an auch an Erweiterungsmöglichkeiten gedacht worden, so dass die Integration der EPM-Laufzeitkomponente ohne Modifikation von vorhandenem Code vorgenommen werden konnte.

Die vorgenommene Implementierung der Enterprise Portal Maps für den infoAsset Broker setzt sich aus der EPM-Laufzeitkomponente für den Server einerseits und andererseits aus den Anpassungen der Benutzungsoberfläche für den Anwender des Portals zusammen. Beide Teile werden im Folgenden näher beschrieben. Zum Schluss dieses Unterkapitels wird eine Enterprise Portal Maps für das Wissensmanagement in Unternehmen vorgestellt.

5.2.1 Eine EPM-Laufzeitkomponente für den infoAsset Broker

Die EPM-Laufzeitkomponente für den Server des infoAsset Brokers setzt sich aus folgenden Elementen zusammen:

- Die EPM-Dienste, die als Schnittstelle zu den Services des infoAsset Brokers dienen.
- Die EPM-Bedingungen, die die bisherigen Berechtigungsprüfungen in den Handlern nachahmen.
- Die neue Klasse `EPMServices`, die die EPM-Dienste initialisiert, die gewünschte Enterprise Portal Map lädt und konzeptionell das Session-Objekt des infoAsset Brokers um die zwei Attribute letzte Station und letzter Dienst erweitert, in dem sie zwei Hashtables mit der Session-ID als Schlüssel und den Attributen als Wert verwaltet.
- Der Handler `StationHandler`, mit den Hilfsklassen `ReqeustServiceHandler` und `SubmitServiceHandler` für die Behandlung von Anfragen an EPM-Stationen.
- Der Handler `SiteMapHandler`, der die EPM-Karte als Site-Map anzeigt und für die Darstellung der aktuellen Station und u.U. gleichzeitig für die Animation der gesuchten Zielstation sorgt.
- Der Handler `RedirectMyPortalWelcomeHandler`, der die Standardstartseite des infoAsset Brokers auf die festgelegte Startstation der Enterprise Portal Map umlenkt.
- Der Klasse `BrokerBridge`, die Schnittstelle zwischen dem EPM-Editor und dem infoAsset Broker implementiert.

Die Elemente der Laufzeitumgebung werden in Abbildung 5.4 als UML-Klassendiagramm mit der Paketstruktur dargestellt. Die Wrapper-Klassen für Assets und Berechtigungen befinden sich in den Paketen `epm.broker.epm.services` bzw. `epm.broker.epm.authority`. Ein *Kontext* besteht bei dieser Implementierung aus einer einfachen Zeichenkette, die einer Asset-ID entspricht. Anhand der Asset-ID und der Typbezeichnung können Bedingungen und Dienste das jeweilige Asset anfordern bzw. manipulieren.

In Abbildung 5.5 wird der prinzipielle Ablauf der Anfragebearbeitung im infoAsset Broker dargestellt. Eine Anfrage wird immer vom Web-Server entgegengenommen, dieser bestimmt den aufzurufenden, speziellen Handler, an den er anschließend die Kontrolle übergibt. Wie in Kapitel 2.2.6 bereits erläutert wurde, lädt im Falle eines sichtbaren Handlers der Handler eine Repräsentation des Templates und ermittelt mit Hilfe der Services die notwendigen Informationen, um die Platzhalter im Template zu ersetzen. Sind alle notwendigen Werte für die Ersetzungen bekannt, so wird das Templates instantiiert, d.h. dass die Ersetzungen nun tatsächlich stattfinden und aus dem Template gültiges HTML erzeugt wird. Die fertige HTML-Seite wird vom Web-Server an den Anfragenden zurückgesendet. Im Falle eines unsichtbaren Handlers, ruft der Handler Methoden der Services auf und gibt anschließend die Kontrolle an einen anderen, sichtbaren oder unsichtbaren Handler weiter.

Die EPM-Laufzeitumgebung ändert den Ablauf wie folgt:

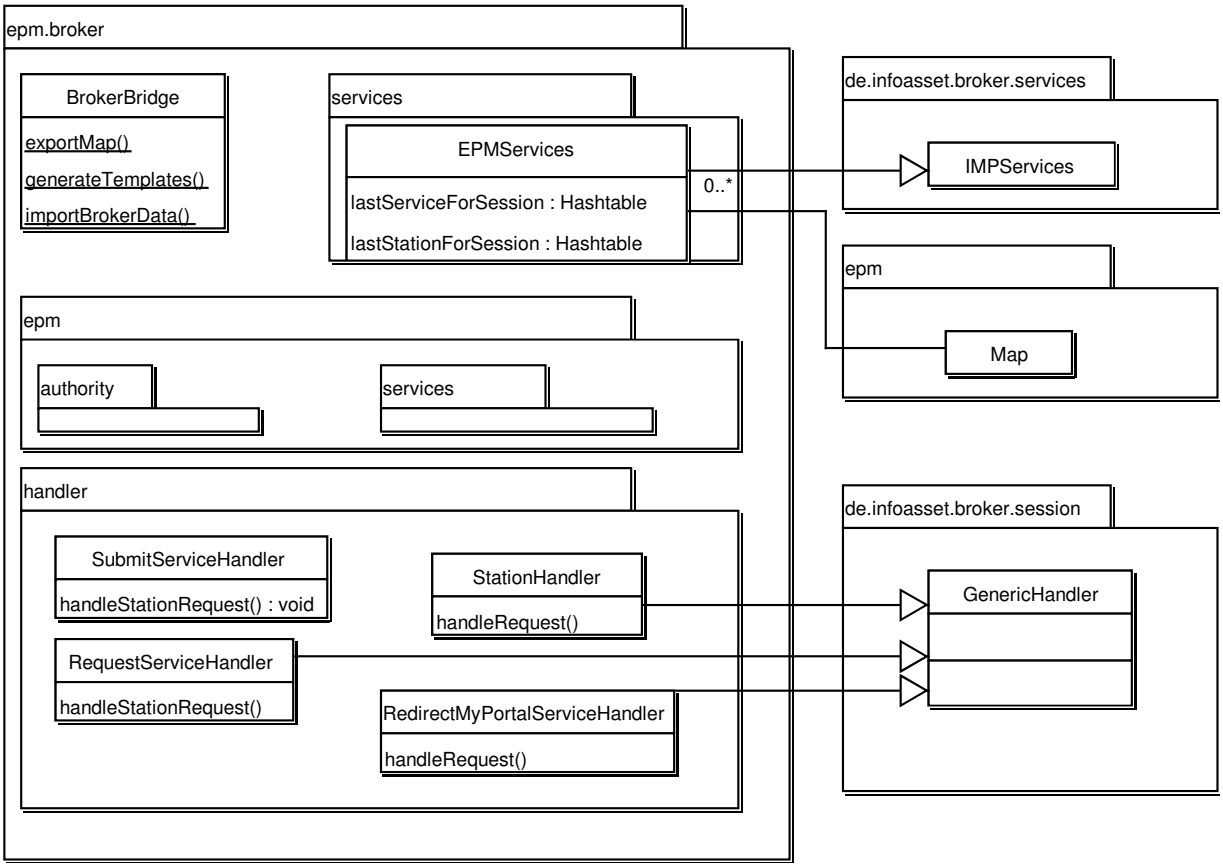

Abbildung 5.4: Die Komponenten der EPM-Laufzeitumgebung für den infoAsset Broker.

- Für alle Anfragen, die an eine EPM-Station gerichtet sind, ist in der Datei `handler.txt` der Handler `StationHandler` als verantwortlich eingetragen, über den folglich jede EPM-Anforderung abgewickelt wird.

- Die `handleRequest`-Methode des `StationHandlers` extrahiert aus der Anfrage den technischen Namen der gewünschten Station und prüft, ob die Bedingung des Dienstes an der Station erfüllt ist. Der Methode `check` wird dabei die aktuelle Session des Anfragenden als Objekt und der Sessionparameter `id` als aktueller Kontext übergeben. Ist die Bedingung nicht erfüllt, so wird eine Fehlermeldung erzeugt und der Benutzer auf die eingestellte Fehlerseite umgeleitet. Ist die Bedingung erfüllt, so wird geprüft, ob eine Anforderungsstation aufgerufen wurde. Ist dies der Fall, so wird die `release`-Methode des letzten in dieser Session verwendeten Dienstes aufgerufen. In jedem Falle wird die aktuelle Station zu dieser Session für den `SiteMapHandler` vermerkt.

- Bei einer Anforderungsstation wird nun der `RequestServiceHandler` aufgerufen.

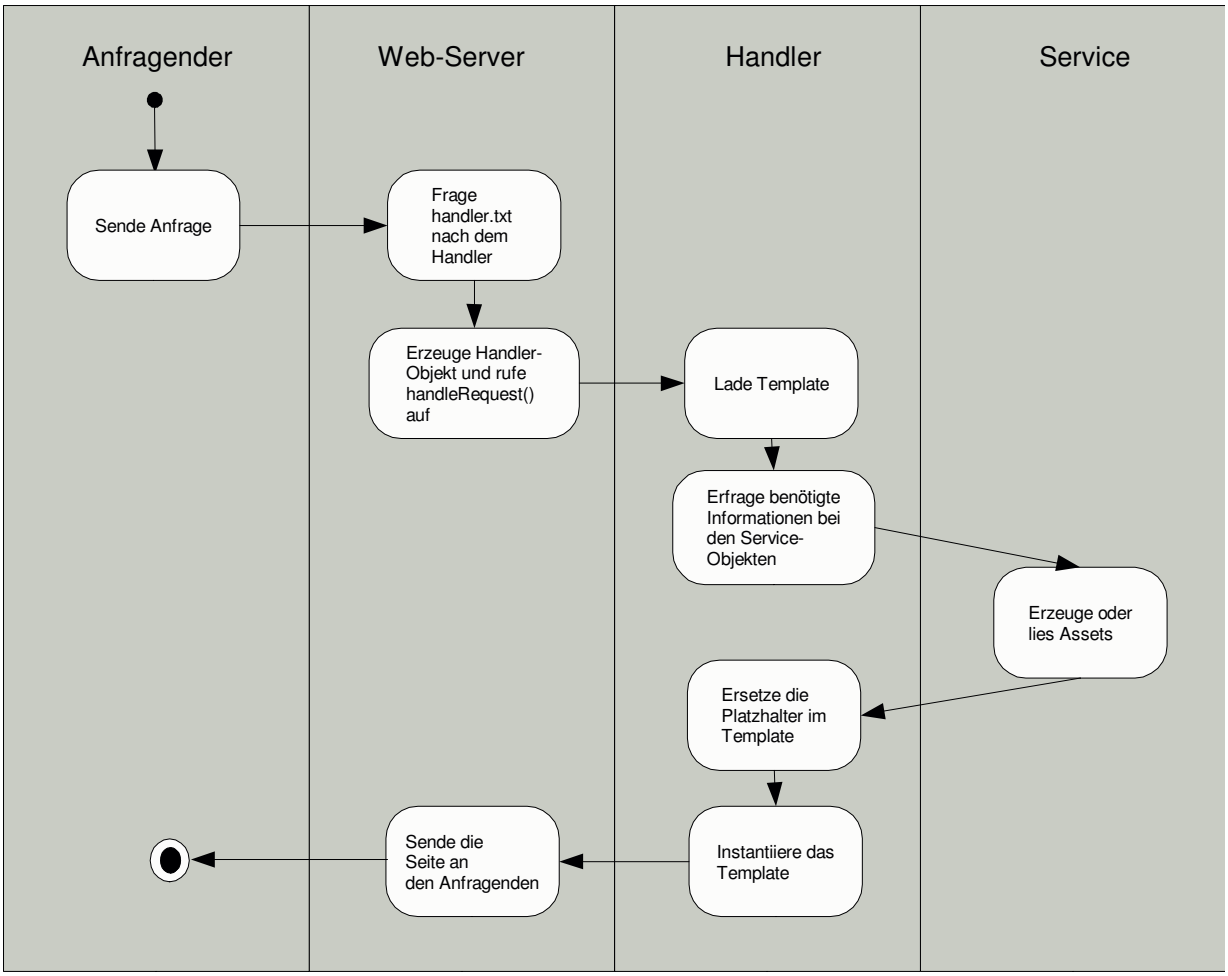

Abbildung 5.5: Ablauf einer Anfragebearbeitung im infoAsset Broker

- Handelt es sich um eine Ausführungsstation, so wird die Kontrolle entsprechend an den `SubmitServiceHandler` weitergegeben.

- Der `RequestServiceHandler` fordert als erstes den im Dienstplan verzeichneten Dienst an, indem die statische Methode `request` mit dem aktuellen Kontext und der ID des aktuellen Benutzers als Parameter aufgerufen wird. Hat der angeforderte Dienst den Kontext verändert, so wird der neue Kontext als Parameter `id` zur Session abgelegt. Anschließend wird das aktuelle Interaktionselement des Dienstes angefordert, das strukturell dem Interaktionselement im Dienstplan entspricht, aber mit den realen Werten gefüllt ist. Die Platzhalter im Template werden durch die entsprechenden Werte ersetzt, die sich aus den Werten des gefüllten Interaktionselementes und den Zielstationen und Anpassungen des Interaktionselementes aus dem Dienstplan ergeben. Dabei wird bei jedem Verweis geprüft, ob die Zielstation auch für den aktuellen Benutzer und im aktuellen Kontext zulässig ist. Eine Zielstation ist

nur dann zulässig, wenn die Bedingung im Dienstplan der Zielstation erfüllt ist. Ist die Zielstation nicht zulässig, so wird der Verweis nicht angezeigt, vgl. Kapitel 2.2.6. Bei der Ersetzung der Platzhalter wird außerdem protokolliert, welche Bedingungen als wahr bzw. falsch erkannt wurden. Die Liste der wahren Bedingungen, und damit indirekt auch die Liste der falschen, wird nach der Auslieferung der Seite über *JavaScript* dem EPM-Applet im Browser des Benutzers mitgeteilt.

- Der `SubmitServiceHandler` prüft zunächst, ob es sich um einen S-Dienst (vgl. Kapitel 4.4.5) handelt. Handelt es sich um einen S-Dienst, dann wird der Dienst proforma angefordert, um ein entsprechendes Dienst-Objekt zu erhalten. Handelt es sich um einen RS-Dienst, so wird versucht, aus den URL-Parametern des Aufrufs ein dem Dienst entsprechendes Interaktionselement zu erzeugen. Bei der Erzeugung werden Einzelwerte mit entsprechenden Typen als Fremdschlüssel betrachtet und über Services-Methoden die passenden Asset-IDs entsprechend des Typs des Einzelwertes gesucht. Konnte ein passendes Interaktionselement erzeugt werden, so wird die `submit`-Methode des Dienstes mit dem erzeugten Interaktionselement als Parameter aufgerufen. Der im Ergebnis zurückgelieferte Kontext wird als Parameter `id` zur aktuellen Session hinzugefügt. Abschließend werden alle Einträge im Fahrplan durchlaufen und diejenigen Transporte durchgeführt, deren Ereignis im Fahrplan mit dem aufgetretenen Ereignis übereinstimmt.

5.2.2 Die Schnittstelle zum EPM-Editor

Für die API-Funktionen (Asset-Methoden), Berechtigungen und Aufgaben des infoAsset Brokers wurden Wrapper-Klassen implementiert, die als EPM-Dienste, Bedingungen und EPM-Aufgaben die Broker-Konstrukte kapseln. Die Integration in den graphischen Editor geschieht mit Hilfe von Konfigurationsdateien, in denen Namen und Wrapper-Klassen benannt werden. Beispielsweise wird in der Konfigurationsdatei für Bedingungen mit der Zeile

```
Dokument schreiben = epm.authority.test.DocumentWriter
```

dem Editor die Bedingung „Dokument schreiben“ bekannt gemacht, die durch die Klasse `epm.broker.epm.authority.DocumentWriter` implementiert wird und dem Interface `epm.Authority` genügt. Da die EPM-Laufzeitumgebung wie der Editor in der Sprache Java implementiert wurde, dient die Implementierung der Klasse `DocumentWriter` sowohl der Definition und Manipulation der Enterprise Portal Map durch den Editor, als auch der Ausführung im Rahmen des infoAsset Brokers. Die Implementierung lautet wir folgt:

```
package epm.broker.epm.authority;

import epm.Authority;
import de.infoasset.broker.session.Session;
import de.infoasset.broker.interfaces.Document;
```

```
public class DocumentWriter extends BrokerAuthority
{
  public boolean check(Session s, String assetId) {
    Document doc =s.getServices().getDocuments().getDocument(assetId);
    String userId = s.getUser().getId();
    if ((doc!=null)&&(doc.mayWrite(userId))) {
      AssetLock lock = session.getServices().
                          getAssetLocks().getAssetLockFor(assetId);
      return (lock==null);
    }
    return false;
  }
}
```

Dabei wird die Klasse `DocumentWriter` von der Klasse `BrokerAuthority` abgeleitet, die die abstrakte Methode `check` definiert, die als Parameter eine Session und eine Asset-Id besitzt. Von der EPM-Laufzeitumgebung für den infoAsset Broker wird dann diese Methode mit den entsprechenden Parametern aufgerufen. Die Prüfung der Bedingung wird in diesem Falle wieder an den infoAsset Broker delegiert, indem das betreffende Dokument selbst befragt wird, ob der Benutzer, der in der Session abgelegt ist, berechtigt ist, Änderungen am Dokument vorzunehmen. Darüber hinaus darf das Dokument nicht gesperrt sein, d.h. dass kein Benutzer das Dokument derzeit bearbeitet. Innerhalb der Methode `check` könnte aber auch beliebiger anderer Code ausgeführt werden, um die Bedingung zu testen.

Das Vorgehen für die Integration von Diensten und Aufträgen ist analog zu der eben beschriebenen Vorgehensweise für Bedingungen. Die doppelte Verwendung der Dienste bzw. Aufgaben im Editor und im Rahmen der EPM-Laufzeitumgebung spiegelt sich im Quelltext wieder, wie anhand des folgenden Dienst für die Änderung einer Gruppe ersichtlich wird:

```
01 package epm.broker.epm.services.groups;
02
03 import epm.*;
04 import epm.broker.epm.services.*;
05 import epm.interactionElements.*;
06 import de.infoasset.broker.interfaces.*;
07 import de.infoasset.broker.services.*;
08 import java.util.*;
09
10 public class Edit extends BrokerService
11 {
12   protected InteractionElement groupIE;
13   protected Group g;
```

```
14
15  protected Edit(Group g, String userId) throws ServiceException {
16    super("Gruppe ändern", "group", "group", userId, g);
17    this.g = g;
18    groupIE = GroupIEs.createGroupIE(g, brokerServices,
                                       true, false, true);
20    state.setResult(null, "group");
21  } // constructor
22
23  public boolean requestIsIndexed() {
24    return true;
25  } // requestIsIndexed
26
27  public static Service request(String index, String userId)
28  throws ServiceException {
29    if (brokerServices != null)
30      return new Edit(brokerServices.getGroups().getGroup(index),
31                      userId);
32    else
33      return new Edit(null, null);
34  } // method reqeust
35
36  public InteractionElement getInteractionElements() {
37    return groupIE;
38  } // method getInteractionElements
39
40  public void submit(InteractionElement ie) {
41    IESingleValue sv;
42    IEList list;
43    IELink link;
44    IESelection sel;
45    Iterator it;
46    sel = (IESelection)ie.getIEByName("groupcategory");
47    if (sel!=null) {
48     it = sel.getSelectedValues();
49     if (it.hasNext()) g.setGroupCategoryId((String)it.next());
50    }
51    sv = (IESingleValue)ie.getIEByName("groupname");
52    if (sv!=null) g.setName(sv.getValue());
53    sv = (IESingleValue)ie.getIEByName("objectives");
54  if (sv!=null) g.setObjectives(sv.getValue());
55    list = (IEList)ie.getIEByName("members");
56    if (list!=null) {
```

```
57      it = brokerServices.getMemberships().
             getMembershipsOfGroup(g.getId());
58      while (it.hasNext())
59        brokerServices.getMemberships().
            remove(((Membership)it.next()).getId());
60      list.startIteration();
61      while (list.hasNextIteration()) {
62        it = list.nextIteration().getIEList();
63        while (it.hasNext()) {
64          Membership m = brokerServices.getMemberships().
                             createMembership();
65          m.setAuthorId(userId);
66          m.setGroupId(g.getId());
67          m.setPersonId(((IESingleValue)it.next()).getValue());
68          m.setRoleId("mebr");
69          m.setStatus("in");
70        }
71      }
72    }
73    if (g.write()) {
74      state.setExecuted(true);
75      state.setMessage("Group saved");
76      state.setResult(g.getId(), "group");
77    } else {
78      state.setErrorNo(1);
79      state.setMessage("broker failed to write group");
80    }
81  } // method request
82 } // class Edit
```

In den Zeilen 03-08 werden alle notwendigen Packages eingebunden. Der Konstruktor in den Zeilen 15-21 setzt über den Konstruktor der Oberklasse die Typinformationen und merkt sich den übergebenen Benutzer und die Gruppe. In der Zeile 18 wird über die statische Methode `createGroupIE` der Hilfsklasse `GroupIEs` das benötigte Interaktionselement erzeugt. Wurde eine gültige Gruppe als Objekt übergeben, so werden die entsprechenden Werte im Interaktionselement gebunden. Die Methode `requestIsIndexed` gibt in den Zeilen 23-25 zu erkennen, dass es sich um einen kontextsensitiven Dienst handelt. Die Methode `request` (Zeilen 27-34) ruft im Wesentlichen den Konstruktor auf, wobei über die Bindung der statischen Variable `brokerServices` erkannt wird, ob der Aufruf aus dem Editor (Variable ist nicht gebunden) oder aus der EPM-Laufzeitumgebung (Variable ist an eine Instanz der Klasse `EPMServices` gebunden) heraus erfolgt ist. Die Methode `submit` in den Zeilen 40-81 liest die neuen Werte der Gruppe aus dem übergebenen Interaktionselement aus und gibt sie an das Gruppenobjekt weiter. In den Zeilen 46-54 werden einfache

Werte gesetzt, in den Zeilen 55-72 werden die Werte eines Iterators weitergereicht. Dazu werden zuerst in den Zeilen 57-59 die alten Mitgliedschaften der Gruppe gelöscht. In den Zeilen 60-71 werden alle die Iterationen der Liste mit dem Namen „members“ durchlaufen und entsprechende Mitgliedschaften zur Gruppe hinzugefügt. Eine Iteration besteht dabei nur aus einem einzigen Interaktionselement, einem Einzelwert der Domäne `person`, der die Asset-ID einer Person repräsentiert. Wie weiter oben beschrieben, stellt die EPM-Laufzeitumgebung vor dem Aufruf der Methode `submit` sicher, dass nur gültige IDs durch den Benutzer eingegeben bzw. vom System akzeptiert werden.

Der Export einer erzeugten Enterprise Portal Map erfolgt auf Dateiebene. Es werden folgende Dateien erzeugt und exportiert:

- Eine Serialisierung der Klasse `Map` für den Import in die Server-Komponente der EPM-Laufzeitumgebung.
- Ein HTML-Template für jede Anforderungsstation.
- Eine vereinfachte, textuelle Beschreibung der Benutzersicht, die dem EPM-Applet im Browser des Benutzers als Grundlage für die Anzeige, Animation und Suche dient.
- Der Netzplan aus Benutzersicht als Grafik für die Site-Map.
- Das Template für die Site-Map inklusive der notwendigen JavaScript-Befehle zur Anzeige der aktuellen Station und ggf. der Animation der gesuchten Zielstation.

Die erzeugten HTML-Templates genügen den Anforderungen für Templates des infoAsset Brokers und berücksichtigen alle vorgenommenen Anpassungen der enthaltenen Interaktionselemente. Die erzeugten Templates werden später durch den `ShowStationHandler` der EPM-Laufzeitumgebung gefüllt. Zum Beispiel könnte das erzeugte Template für die Station „Person anzeigen“ im Wesentlichen (ohne Zeilennummern und Auslassungen) wie folgt aussehen:

```
01 <HTML>
02 <HEAD>
03 ...
04 </HEAD>
05 <BODY ...>
06  <p><h1>Person anzeigen</h1></p>
07  <p class="error">$epm_msg$</p>
08  ...
09  <TABLE>
10    <tr> <td>Titel:</td><td>$title_selectedValue$</td></tr>
11    <tr> <td>Vorname:</td><td>$firstname_value$</td></tr>
12    <tr> <td>Lastname:</td><td>$lastname_value$</td></tr>
13    ...
14    $[Benutzer_ist_Person_oder_Admin$
```

```
15    <tr> <td>Passwort:</td><td>$password_value$</td></tr>
16    $]Benutzer_ist_Person_oder_Admin[$
17    $Benutzer_ist_Person_oder_Admin]$
18    <tr> <td>Begriffe:</td>
19         <td><table>
20          $[concepts concepts$
21            $[concepts.concept_show$
22               <a href="$concepts.concept_destinationString$">
23               $concepts.concept_label$</a>
24            $concepts.concept_show]$
25            ...
26          $]concepts[$
27          $concepts]$
28         </table></td>
29    </tr>
30    ...
31    $[Link355143542_show$
32    <tr><td>
33      <input type="button" name="Link355143542" value="Bearbeiten"
34             onClick="$Link355143542_destinationString$">
35    </td></tr>
36    $Link355143542_show]$
37  </TABLE>
38 </BODY>
39 </HTML>
```

In Zeile 06 wird der Stationsname als Überschrift auf dem Template ausgegeben. Zeile 07 bestimmt, dass direkt unterhalb mögliche Nachrichten angezeigt werden sollen. In den Zeilen 10-13 werden der Titel, der Name, der Vorname etc. der Person angezeigt. Wie erwähnt, soll das Passwort nur dann angezeigt werden, wenn die Bedingung „Benutzer ist Person oder Admin“ erfüllt ist. Hierzu wird in Zeile 14 ein entsprechend benannter bedingter Platzhalter eingefügt, der die in Kapitel 5.1.2 beschriebene Anpassung umsetzt.

In den Zeilen 20-27 wird ein Listenplatzhalter verwendet, der alle der Person zugeordneten Begriffe aufzählt. Jeder Verweis auf eine andere EPM-Station wird automatisch mit einem bedingten Platzhalter versehen, so dass zur Laufzeit in Abhängigkeit der Erfüllung der Bedingung an der Zielstation der Verweis angezeigt wird oder nicht. In den Zeilen 21 und 31 finden sich Beispiele hierfür. Die Ziel-URL wird durch einen einfachen Platzhalter wiedergegeben, da u.U. zur Laufzeit zusätzliche Parameter, wie z.B. die Kontext-ID, zur URL hinzugefügt werden müssen.

5.2.3 Anpassung der Benutzungsoberfläche

Die erzeugte Benutzungsoberfläche orientiert sich an der Standard-Benutzungsoberfläche des infoAsset Brokers. Darüber hinaus wurden im Laufe dieser Arbeit verschiedene Modifikationen der Benutzungsoberfläche getestet, die jeweils andere Aspekte der Enterprise Portal Maps betonten. Als Ergebnis ergaben sich folgende wesentliche Unterschiede zu der in ([Weg02, Seiten 195ff.]) beschriebenen Standard-Benutzungsoberfläche des infoAsset Brokers:

- Statt des graphischen Begriffsnavigators wird ein prototypisches EPM-Applet verwendet.
- Es existiert eine graphische, interaktive, generierte Site-Map, die alle sichtbaren Stationen des Portals verzeichnet.
- Der Aufbau der HTML-Templates ist generiert und damit über das ganze Portal hinweg einheitlich.

Die neue Oberfläche wird in Abbildung 5.6 als Bildschirmausschnitt wiedergegeben.

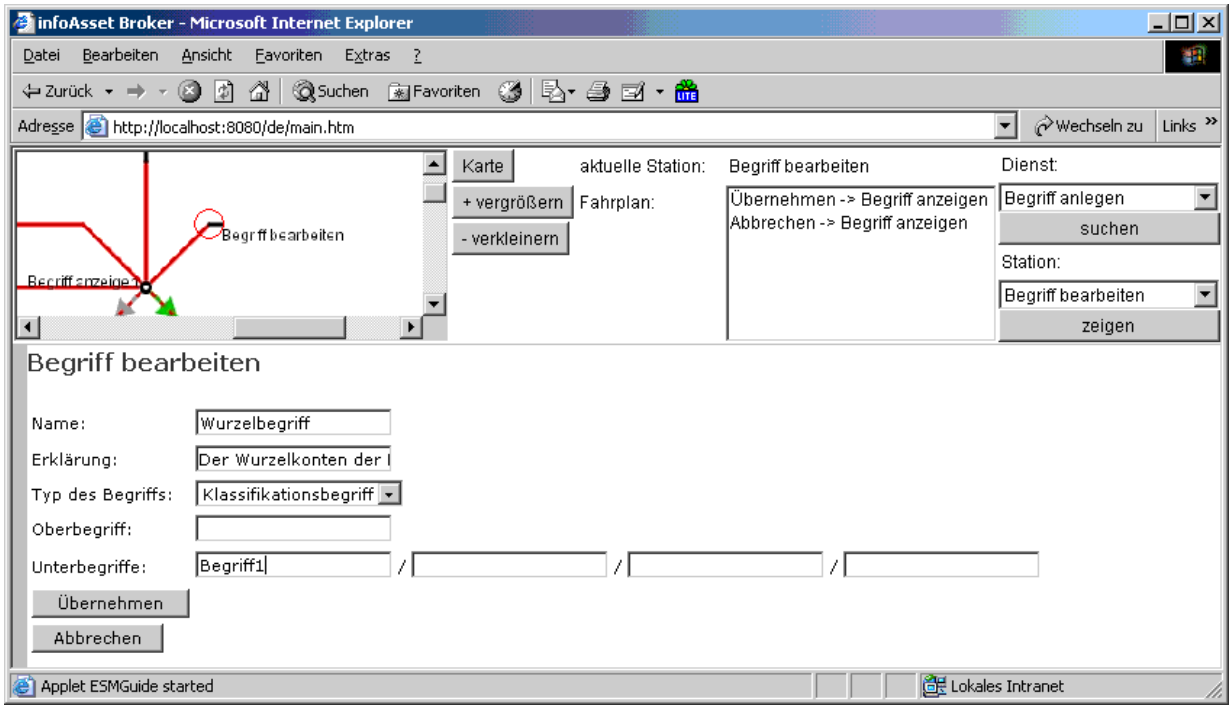

Abbildung 5.6: Die Benutzungsoberfläche mit EPM-Applet.

Das EPM-Applet, das sich im oberen Drittel befindet, bietet folgende Funktionen:

- Es zeigt die Benutzersicht der Enterprise Portal Map an und hebt durch einen roten Kreis die aktuelle Station hervor, die zusätzlich im Zentrum des Ausschnitts liegt. Die Ansicht kann vergrößert oder verkleinert werden, der Kartenausschnitt kann verschoben werden. Navigiert der Benutzer zu einer anderen Station, so wird der Kartenausschnitt entlang der Strecke verschoben, so dass der Eindruck einer „Fahrt“ von einer Station zur nächsten entsteht.

- Über die Schaltfläche „Karte“ kann auf die Site-Map navigiert werden, die als HTML-Seite selbst nicht auf der Enterprise Portal Map verzeichnet ist.

- Im mittleren Teil des Applets wird der Name der aktuellen Station und der aktuelle Fahrplan angezeigt. Dabei werden nur solche Einträge angezeigt, die der Benutzer aufgrund des aktuellen Kontextes auch tatsächlich auswählen kann.

- Der rechte Teile des Applets bietet zwei Suchfunktionen an. Bei der ersten Suchfunktion kann zunächst aus der Liste aller Dienste ein Dienst ausgesucht werden, über die Schaltfläche „suchen“ wird dann in der Liste der Stationen unterhalb der Schaltfläche diejenige Station automatisch selektiert, in deren Dienstplan der gesuchte Dienst angeboten wird. Über die Schaltfläche „zeigen“ unterhalb der Liste der Stationen wird zur Site-Map navigiert und die gesuchte Station hervorgehoben.

Die in Abbildung 5.7 dargestellte Site-Map bietet einem Benutzer folgende Hilfestellungen und Interaktionsmöglichkeiten an:

- Die aktuelle Station, in der Abbildung die Station „Dokument suchen“, wird durch einen roten, durchgezogenen Kreis markiert.

- Wurde eine Station über das Applet gesucht, so wird diese durch einen roten, rotierenden Kreisabschnitt animiert hervorgehoben. In der Abbildung wurde nach der Station „Dokument anlegen“ gesucht, es ist ein roter Kreisabschnitt zu sehen.Fährt der Benutzer mit dem Mauszeiger über eine Station, so werden die dort angebotenen Dienste als Tipp angezeigt. In der Abbildung schwebt der Mauszeiger über der Station „Dokument anzeigen“

- Klickt der Benutzer auf eine Station, die einen kontextfreien Dienst anbietet, so wird direkt dorthin verzweigt. Damit sind direkte Sprünge zu bestimmten Seiten möglich.

- Kann nicht direkt zu einer Station navigiert werden, so erscheint eine entsprechende Fehlermeldung.

5.2.4 EPM für ein Wissensmanagementportal

Nachdem im bisherigen Teil dieses Kapitels die Möglichkeiten zur Definition einer Enterprise Portal Map und ihre Integration in den infoAsset Broker als Portalsystem erläutert wurden, wird im Folgenden ein reales Beispiel beleuchtet, das die praktische Anwendung der vorgestellten Konzepte verdeutlichen soll.

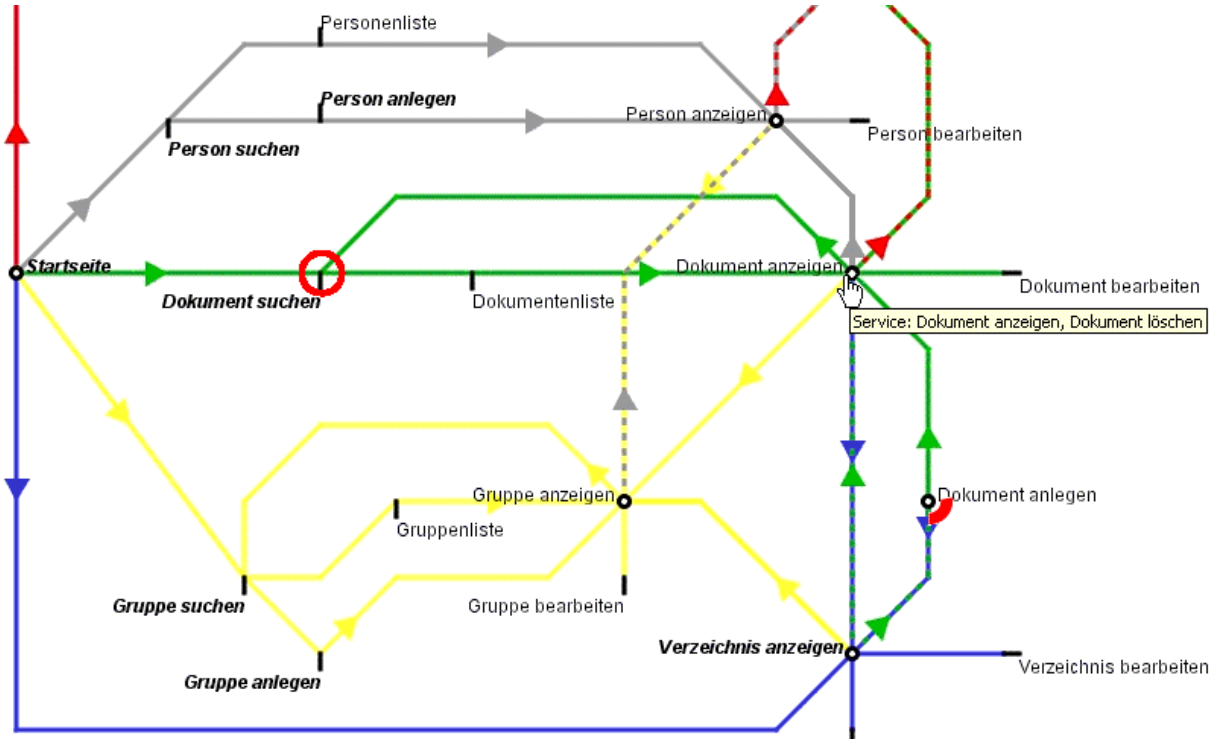

Abbildung 5.7: Auszug aus einer Site Map.

Ein praktischer Ansatz zum Wissensmanagement

Der infoAsset Broker stellt als Produkt einen an der Praxis orientierten Ansatz zum Wissensmanagement in Unternehmen dar. Die wesentlichen Elemente des Ansatzes sind ([inf01]):

- *Dokumente*: Dokumente repräsentieren das explizite Wissen. Ein Dokument verfügt über diverse Eigenschaften, wie z.B. einen Titel, einen oder mehrere Autoren, einen Status oder auch eine zugehörige Datei. Der infoAsset Broker unterstützt verschiedene Strategien zur Suche nach Dokumenten, u.a. die Suche über Eigenschaften und eine Volltextsuche.

- *Verzeichnisse*: Verzeichnisse sind die logischen Ablageorte für Dokumente. Ein Dokument kann dabei nur in genau einem Verzeichnis abgelegt sein. Das Verzeichnis bestimmt im Zusammenspiel mit dem Status des Dokuments die Regeln zur Sicht- und Änderbarkeit des Dokuments für verschiedene Benutzergruppen. Dem Verzeichnis ist weiterhin eine Gruppe zugeordnet, deren Mitglieder das Verzeichnis administrieren dürfen.

- *Personen*: Personen sind die Träger des impliziten Wissens in Unternehmen. Personen interagieren mit den im System vorhandenen Informationsobjekten, wie z.B.

Dokumenten, Verzeichnissen oder anderen Personen. Der infoAsset Broker verwaltet diverse Informationen über Personen, u.a. den Namen, die Adresse, die erstellten, verfassten oder geänderten Dokumente, die Wissensgebiete und die Gruppenmitgliedschaften.

- *Gruppen*: Personen können Mitglieder in beliebig vielen Gruppen sein. Gruppen im infoAsset Broker sind flach, d.h., dass Gruppen keine Untergruppen besitzen können. Das Berechtigungskonzept der Standardauslieferung des infoAsset Brokers beruht ausschließlich auf der Mitgliedschaft zu Gruppen.

- *Begriffe*: Begriffe und ihre Beziehungen untereinander stellen ein zentrales Ordnungssystem, eine sog. Wissenslandkarte, für die Informationsobjekte im infoAsset Broker dar. Über die Begriffe wird das implizite Wissen thematisch mit dem expliziten Wissen verknüpft.

In der nachfolgenden Tabelle werden die wesentlichen Funktionen und Berechtigungen für das Wissensmanagement nach obigen Ansatz systematisch aufgelistet. Informationsobjekte (Assets) in eckigen Klammern deuten darauf hin, dass das genannte Element über genügend Daten verfügt, um eine Navigation zu dem in den eckigen Klammern genannten Element zu ermöglichen.

Element	**Funktion**	**Informationsobjekte**	**Berechtigung**
Dokument	Anlage eines Dokuments in einem Verzeichnis und Eingabe der Eigenschaften.	Document, [Directory], [Person], [Concept]	Dokument anlegen (entspricht Schreibrecht für Dokumente im Verzeichnis)
Dokument	Anzeige eines Dokuments mit seinen Eigenschaften	Document, [Directory], [Person], [Concept]	Dokument lesen (entspricht Leserecht im Verzeichnis für Dokumente mit dem entsprechenden Status)
Dokument	Änderung der Eigenschaften eines Dokuments	Document, [Directory], [Person], [Concept]	Dokument ändern (entspricht Schreibrecht für Dokumente im Verzeichnis)
Dokument	Löschen eines Dokuments	Document	Dokument ändern
Dokument	Suchen eines Dokuments anhand von Eigenschaften	Document, [Directory], [Person], [Concept]	-

Element	**Funktion**	**Informationsobjekte**	**Berechtigung**
Verzeichnis	Anzeige eines Verzeichnisses mit seinen Eigenschaften und enthaltenen Dokumenten	Directory, [Document], [Group]	Verzeichnis lesen
Verzeichnis	Anlage eines Unterverzeichnisses mit Eigenschaften	Directory, [Group]	Verzeichnis ändern
Verzeichnis	Änderung der Eigenschaften eines Verzeichnisses	Directory, [Group]	Verzeichnis ändern
Verzeichnis	Löschen eines Verzeichnisses	Directory	Verzeichnis ändern
Person	Anlage einer neuen Person im infoAsset Broker	Person	Benutzer ist Administrator
Person	Anzeigen der Eigenschaften einer Person	Person, [Group], [Concept]	keine für allgemeine Eigenschaften, Benutzer ist Person oder Administrator für besondere Eigenschaften
Person	Änderung der Eigenschaften einer Person	Person	Benutzer ist Person oder Administrator
Person	Nach einer Person über den Namen suchen	Person	-
Gruppe	Anlage einer Gruppe	Group	Benutzer ist Administrator
Gruppe	Anzeige einer Gruppe mit Eigenschaften und der Mitgliederliste	Group, [Person]	-
Gruppe	Bearbeiten der Eigenschaften und der Mitgliederliste einer Gruppe	Group, [Person]	Gruppe bearbeiten
Gruppe	Löschen einer Gruppe	Group	Gruppe bearbeiten
Gruppe	Nach einer Gruppe über den Namen suchen	Group	-
Begriff	Anlage eines Unterbegriffs mit Eigenschaften	Concept	(Ober-)Begriff bearbeiten
Begriff	Anzeige eines Begriffs mit Eigenschaften und zugeordneten Personen und Dokumenten	Concept, [Person], [Document]	-

Element	Funktion	Informationsobjekte	Berechtigung
Begriff	Änderung der Eigenschaften, der zugeordneten Dokumente und Personen eines Begriffs	Concept, [Person], [Document]	Begriff bearbeiten
Begriff	Löschen eines Begriffs	Concept	Begriff bearbeiten
Begriff	Nach einem Begriff über seinen Namen suchen	Concept	-

Neben diesen Funktionen werden für den tatsächlichen Einsatz eines Portals für das Wissensmanagement noch (mindestens) zwei weitere Funktionen benötigt: Erstens eine Möglichkeit für die Benutzer, sich im Portal anzumelden und zweitens eine Möglichkeit zur technischen Administration des Servers. Beides sind unabdingbare Funktionen eines jeden (personalisierbaren) Portals.

EPM für ein Wissensmanagementportal

Um für ein vollständiges Wissensmanagementportal eine Enterprise Portal Map zu definieren, wurden für jede im vorigen Abschnitt aufgezählte Funktion entsprechende EPM-Dienste entwickelt, die zur Implementierung der Funktionalität ausschließlich auf die Services des infoAsset Brokers zurückgreifen. Nach Assets geordnet wurden folgende Dienste erstellt:

- `Document: Create, Delete, Edit, Search, SearchResultList, Show`
- `Directories: Create, Delete, Edit, Show`
- `Persons: Create, Delete, Edit, Search, SearchResultList, Show`
- `Groups: Create, Delete, Edit, Search, SearchResultList, Show`
- `Concepts: Create, Delete, Edit, Search, SearchResultList, Show`

Alle Dienste `Create`, `Edit` und `Search` sind vom Typ RS-Dienste, die `Delete`-Dienste sind S-Dienste, `SearchResultList` und `Show` sind R-Dienste.

Die Suche nach einem Asset teilt sich auf zwei Dienste auf: Erstens in einen Dienst `Search`, der als Interaktionselement die Suchparameter zur Verfügung stellt und die Ergebnismenge berechnet. Zweitens in einen Dienst `SearchResultList`, der lediglich der Darstellung der Ergebnismenge dient und eine iterative Liste als Interaktionselement hat.

Zusätzlich zu den oben genannten Diensten wurde der Dienst `Homepage` entwickelt, der als R-Dienst auf der Startseite des Portals angefordert wird und die Anzeige einer personalisierten Grußformel ermöglicht.

Die im vorigen Unterkapitel erwähnten Berechtigungen wurden als Bedingungen implementiert. Insgesamt wurden folgende Bedingungen entwickelt: `Always`, `ConceptEditor`, `DirectoryEditor`, `DirectoryReader`, `DirectoryWriter`,

`DocumentReader`, `DocumentWriter`, `GroupEditor`, `IsLoggedIn`, `IsNotLoggedIn`, `UserIsAdmin`, `UserIsPersonOrAdmin`. Die Methode `check` der Bedingung `Always` evaluiert stets zu wahr. Die Bedingungen `IsLoggedIn` bzw. `IsNotLoggedIn` prüfen, ob der aktuelle Benutzer eingeloggt ist, oder als so genannter „anonymer Benutzer“ das Portal verwendet. Die Bedeutungen der übrigen Bedingungen ergeben sich aus ihrem Namen.

In Abbildung 5.8 wird die Designersicht der entstandenen Enterprise Portal Map im EPM-Editor wiedergegeben. Es wurden 42 Stationen angelegt.

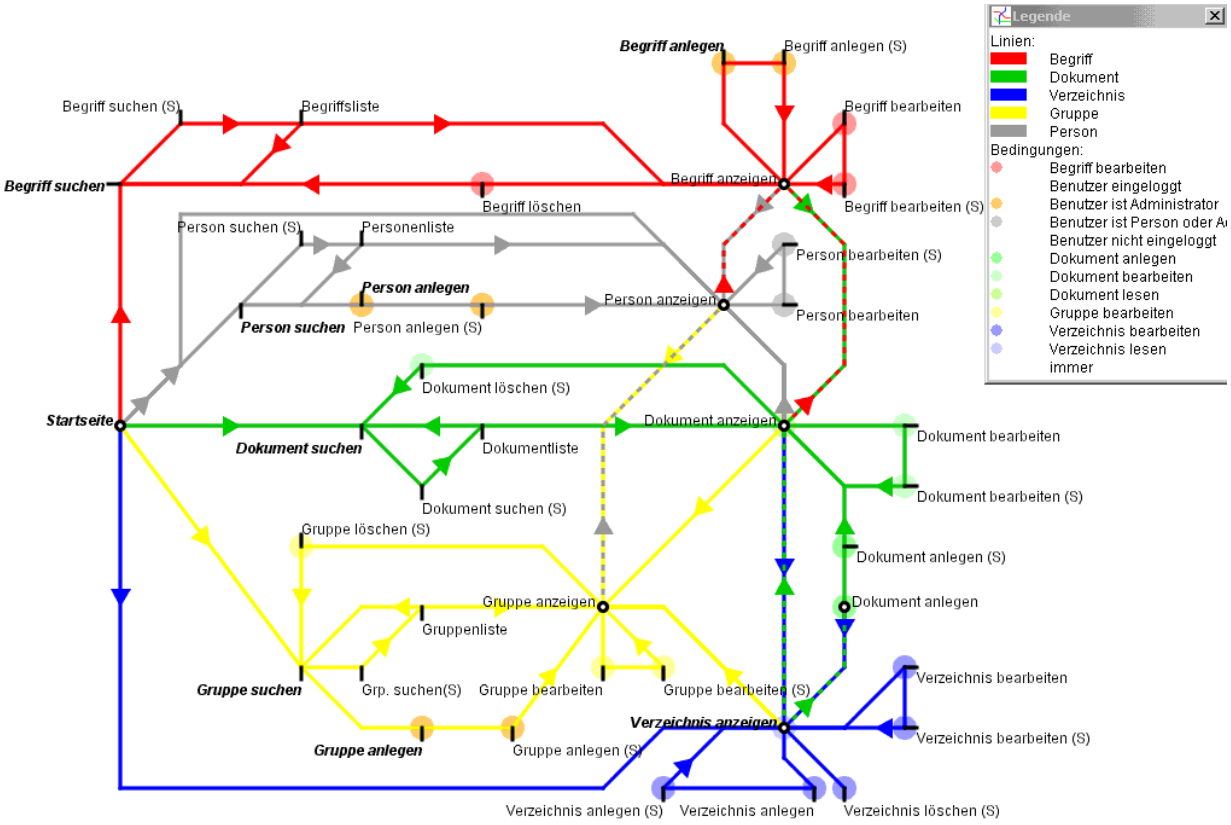

Abbildung 5.8: Designersicht einer Enterprise Portal Map für ein Wissensmanagementportal.

Der Name einer Station lässt einen Rückschluss auf den angebotenen Dienst zu. Endet der Name mit dem Anhängsel „(S)“, so wird der Dienst an dieser Station ausgeführt, d.h., dass die Station eine Ausführungsstation ist. Die fünf Linien Begriffe, Personen, Dokumente, Verzeichnisse und Gruppen entsprechen den verwendeten Assets, dabei findet ein Transport immer auf der Linie statt, die zu dem an der Zielstation verwendeten Asset passt. Es bestand keine Notwendigkeit, eine Linie Startseite einzuführen, da es keinen direkten Verweis auf die Station Startseite gibt. Ein solcher Verweis ist auch nicht notwendig, da zu der Station aus jedem Kontext heraus direkt navigiert werden kann.

Die Benutzersicht entspricht optisch der Designersicht, wenn man alle Ausführungsstationen entfernt und die „Umwege“ der Streckenführung über die Ausführungsstationen beseitigt oder begradigt. Die in Abbildung 5.7 (Seite 112) in Teilen gezeigte Site-Map wurde unmittelbar aus der generierten Benutzersicht gewonnen.

Die Station `Startseite` weist im Vergleich zu den anderen Stationen eine Besonderheit auf: Das Interaktionselement des angebotenen Dienstes wurde um drei Verweise erweitert, die auf externe Seiten, d.h. auf Seiten, die nicht mit dem EPM-Definitionswerkzeug erstellt wurden, verweisen. Es handelt sich dabei zum einen um zwei wechselseitig angezeigte Verweise zum An- bzw. Abmelden des Benutzers und um einen Verweis für den Administrator des Portals auf die Administrationsseiten des infoAsset Brokers. Abbildung 5.9 zeigt die Startseite, wie sie der angemeldete Benutzer „Max Mustermann“, der Mitglied der Gruppe Administrator ist, erhält. Unter der Anzeige seines Namens und dem Zeitpunkt der letzten Anmeldung erscheint der Verweis auf die externe Funktion zum Abmelden, anschließend werden gemäß der Enterprise Portal Map Verweise zu EPM-Stationen (hier als Schaltflächen dargestellt) angezeigt. Der Verweis auf die externe Serveradministration folgt als letzter Verweis.

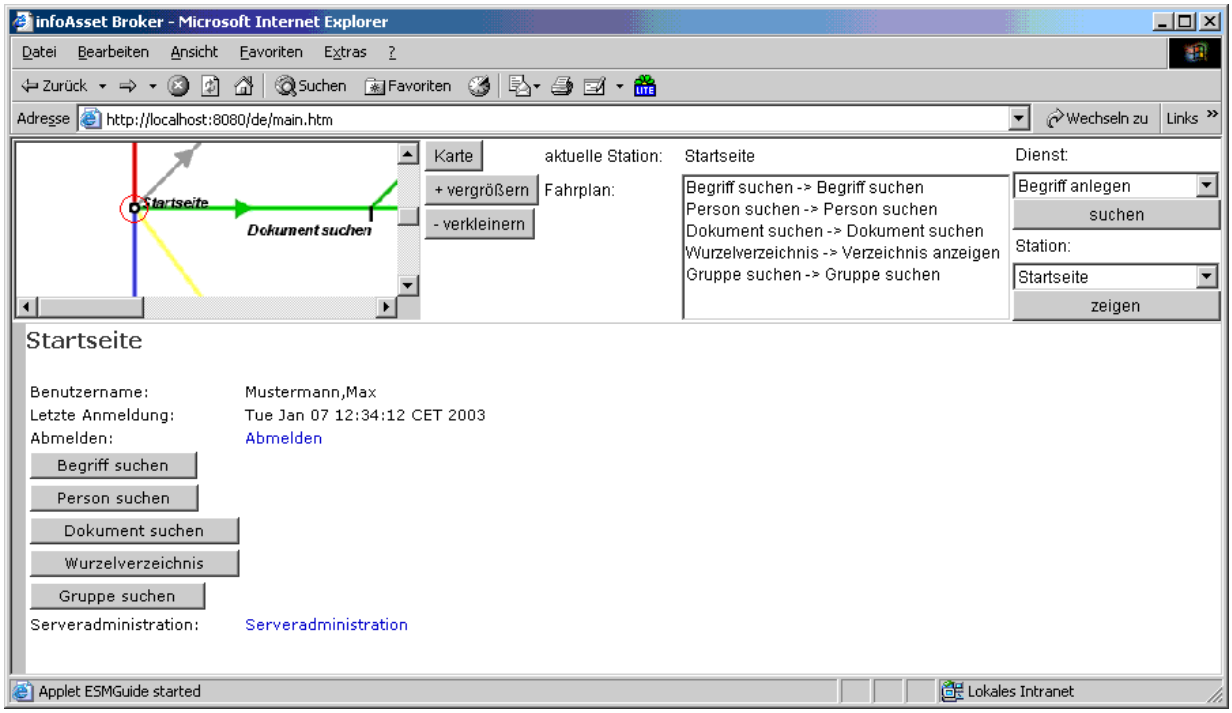

Abbildung 5.9: Startseite des Portals nach der Anmeldung als Administrator.

5.3 EPM für den SAP Web Application Server

Nachdem im vorigen Unterkapitel die Anwendung des Modells der Enterprise Portal Maps auf das Wissensmanagement in Unternehmen auf der Basis des infoAsset Brokers beschrieben wurde, soll nun die Erweiterung eines anderen Portalsystems um das Modell der Enterprise Portal Maps vorgestellt werden. Aus folgenden Gründen fiel die Wahl auf den SAP Web Application Server als Portalsystem:

- Mit dem SAP Web Application Server als Bestandteil der SAP R/3-Basis kann auf eine kommerziell sehr erfolgreiche Plattform zurückgegriffen werden, deren betriebswirtschaftlicher Nutzen für Unternehmen außer Frage steht. Die Betriebswirtschaftlichen Funktionen werden dabei zunehmend in Form von Geschäftsobjekten (engl. Business Objects) modelliert. Die Methoden der Geschäftsobjekte, die sog. *BAPIs* (Business Application Programming Interfaces), sind aus Programmen heraus aufrufbar, auch wenn es sich dabei nicht um echte Methoden im Sinne einer Programmiersprache, sondern um Funktionsbausteine handelt.

- Der SAP Web Application Server unterscheidet sich von seinem Aufbau und von seiner Herangehensweise an die Problematik der Dienste und Portale her deutlich vom infoAsset Broker. Vorteile, die sich bei der Verwendung von Enterprise Portal Maps sowohl mit den infoAsset Broker, als auch mit dem SAP Web Application Server zeigen, können als Vorteile des EPM-Modells gewertet werden, da sie sich offensichtlich nicht nur aus den Spezifika des jeweiligen Systems ergeben.

- Die Objekt-orientierten Erweiterungen der ABAP-Sprache und die volle Reflexivität der Entwicklungsumgebung lassen die Entwicklung von Programmen zu, die weit mächtiger sind, als dies die traditionellen Programmiersprachen für Datenbanken ermöglichen.

Im Folgenden wird zunächst die EPM-Laufzeitumgebung für den SAP Web Application Server und anschließend die Schnittstelle zum EPM-Editor eingehender beschrieben. Die Schnittstelle zum Editor gestaltet sich dabei etwas komplizierter, da die die Laufzeitumgebung in der Sprache ABAP geschrieben ist und somit Objekte bzw. passende Repräsentanten zwischen dem ABAP der Laufzeitumgebung und dem Java des Editors in beide Richtungen ausgetauscht werden müssen. Zum Abschluss dieses Unterkapitels wird eine Enterprise Portal Map vorgestellt, die die Methoden der Geschäftsobjekte zum klassischen ABAP-Schulungsbeispiel der Flugbuchungen als EPM-Dienste und -Bedingungen umsetzt.

5.3.1 EPM-Laufzeitumgebung für den SAP Web AS

Die EPM-Laufzeitumgebung für den SAP Web Application Server setzt sich aus folgenden Komponenten zusammen:

- Die EPM-Dienste in ABAP, die die BAPI-Aufrufe kapseln.

- Der generische EPM-Dienst `SAPService`, der im EPM-Editor als Platzhalter für die eigentlichen ABAP-EPM-Dienste dient.
- Die EPM-Bedingungen in ABAP, die die bisherigen Prüfungen in den BSP-Seiten nachahmen.
- Die generische EPM-Bedingung `SAPAuthority`, die im EPM-Editor als Platzhalter für die ABAP-EPM-Bedingung dient.
- Die zustandsbehaftete BSP-Applikation `Z_EPM_RUNTIME`, die die EPM-Laufzeitumgebung auf der Basis des SAP Web Application Servers implementiert. Neben einigen Mime-Objekten beinhaltet die Applikation folgende die Seiten:
  - `default.htm`: Die Einstiegseite zum erstmaligen Aufruf des Portals. Diese Seite definiert die notwendigen Frames für die Benutzungsoberfläche.
  - `error.htm`: Diese Seite wird bei internen Fehlern angezeigt.
  - `guide.htm`: Hierüber wird das EPM-Applet eingebunden.
  - `line.htm`: Hilfsseite, um „schöne“ Trennlinien zwischen den Frames zu ziehen.
  - `sitemap.htm`: Zeigt die von der Schnittstelle zum EPM-Editor erzeugt Grafik als Site-Map an.
  - `station.htm`: Die Standardseite, die Verteilung der Anfragen und die Anzeige von Anforderungsstationen vornimmt.
  - `submit.htm`: Hilfsseite für die Bearbeitung einer Ausführungsstation.
- Die Klasse `Z_EPM_APPLICATION`, die von der BSP-Applikation `Z_EPM_APPLICATION` als Applikationsobjekt (vgl. Kapitel 2.2.5) verwendet wird. Die Klasse dient einerseits als Behälter für verschiedene Daten und andererseits stellt sie eine Methode zur Abbildung eines Interaktionselementes auf HTML-Code und zum Import einer Enterprise Portal Map aus dem EPM-Editor zur Verfügung.
- Der Report `Z_EPM_EXPORT` verwendet Teile der in Kapitel 4.8 vorgestellten Grammatik, um die vorhandenen EPM-Dienst und -Bedingungen an den EPM-Editor zu übertragen.

Die Klassen der EPM-Laufzeitumgebung werden in Abbildung 5.10 als UML-Diagramm dargestellt. Auffällig ist, dass nur einige - nicht alle - Klassen des Modells der Enterprise Portal Maps implementiert wurden. Dies ist dadurch zu erklären, dass die fehlenden Klassen, wie z.B. *Linie*, nur für die *Definition*, nicht aber für die *Ausführung* einer Enterprise Portal Map notwendig sind.

Die EPM-Laufzeitumgebung verwendet als BSP-Applikation nicht das Model-View-Controller-Pattern, sondern stützt sich auf BSP-Seiten mit Ablauflogik. Wie in Kapitel 2.2.5 erwähnt, werden bei solchen Seiten zu verschiedenen Zeitpunkten unterschiedliche ABAP-Abschnitte (Handler) aufgerufen. Es existieren folgende Zeitpunkte mit entsprechenden Handlern:

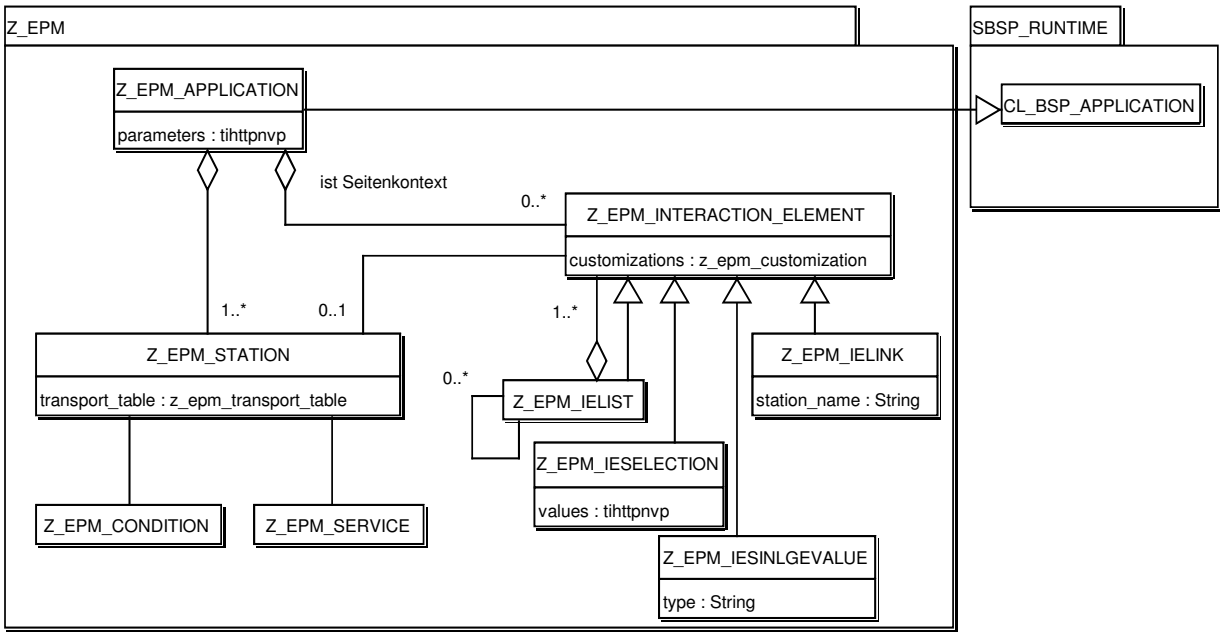

Abbildung 5.10: Die Klassen der EPM-Laufzeitumgebung für den SAP Web Application Server.

- *OnCreate*: Dieser Handler wird bei zustandsbehafteten Applikationen einmal bei der ersten Erzeugung der Seite aufgerufen. Hier werden typischerweise Instantiierungen u.ä. vorgenommen.

- *OnRequest*: Wird immer aufgerufen, wenn es eine Anforderung (einen Request) auf diese Seite gab. Bei zustandslosen Applikationen werden hier typischerweise interne Datenstrukturen (re-)initialisiert.

- *OnInitialization*: Zu diesem Zeitpunkt können für die BSP-Seite benötigte Daten beschafft werden.

- *OnInputProcessing*: Dieser Handler dient der Verarbeitung von Eingaben durch den Benutzer. Bei SUBMIT-Buttons in HTML-Formularen steht zusätzlich die Information zur Verfügung, welcher Button betätigt wurde. In Abhängigkeit der Benutzereingabe wird typischerweise zu diesem Zeitpunkt explizit zur nächsten Seite navigiert.

- *OnManipulation*: Dieser Handler kann noch letzte Änderungen am HTML-Stream vornehmen, bevor er versendet wird. Alle server-seitigen Skripte sind bereits ausgeführt.

Die Abfolge der Events ist wie folgt:

1. Zuerst findet das Ereignis OnCreate immer dann statt, wenn eine Seite zu ersten mal aufgerufen wird. Bei einer zustandslosen BSP-Applikation ist dies bei *jedem* Aufruf der Seite der Fall.

2. Anschließend tritt das Ereignis OnRequest bei zustandslosen wie zustandsbehafteten Applikationen ein.

3. Handelt es sich bei der Anforderung um HTML-Input resultierend aus einem SUBMIT in einem HTML-Formular, wird der OnInputProcessing-Handler aufgerufen.

4. Nun findet das Ereignis OnInitialization statt.

5. Nach der Erzeugung der Antwortseite tritt ggf. noch das Ereignis OnManipulation ein.

Abbildung 5.11 stellt die Abfolge der Ereignisse graphisch dar. Dabei ist zu beachten, dass zu jedem Zeitpunkt die Abfolge unterbrochen werden kann, indem explizit zu einer anderen Seite navigiert wird.

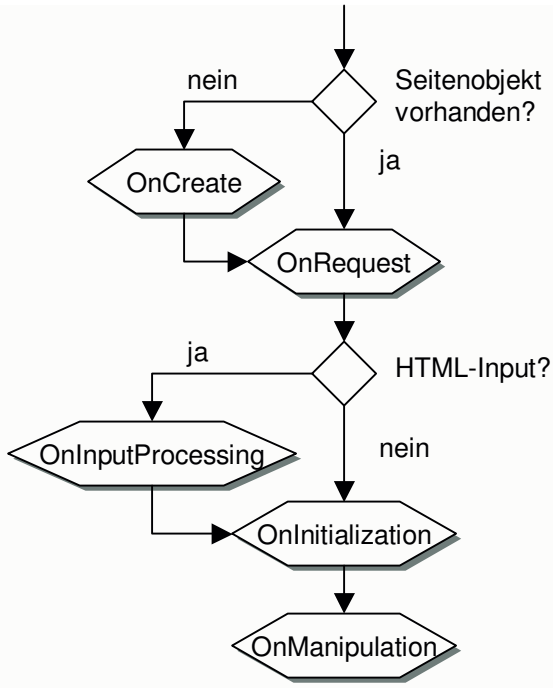

Abbildung 5.11: Abfolge der Ereignisse während der Abarbeitung einer BSP-Seite.

Die EPM-Laufzeitumgebung hat zwei zentrale Seiten, die immer wieder aufgerufen werden: Zum Ersten die Seite `station.htm` für die Darstellung Anforderungsstationen und der Umsetzung des Kontrollflusses. Zum Zweiten existiert die Seite `submit.htm`, die

einen angeforderten Dienst ausführt und gemäß des Fahrplans der aktuellen Station die nächste Station aufruft.

Die Seite `station.htm` implementiert Handler für folgende Zeitpunkte:

- *OnCreate*: Hier wird in das Applikationsobjekt die gewünschte Enterprise Portal Map eingelesen, falls dies notwendig ist. Die Enterprise Portal Map liegt dabei als Text-Datei auf dem Server vor, die im Wesentlichen der in Kapitel 4.8 angegebenen Syntax entspricht.

- *OnRequest*: Dieses ist der zentrale Handler für die Ablaufsteuerung der Laufzeitumgebung. Er wird weiter unten eingehender erörtert.

- *OnInputProcessing*: Wird nach der Anforderung eines Dienstes über einen SUBMIT-Button die Ausführung angefordert, so werden in diesem Handler die im HTTP-Request mit übergebenen Parameter ausgelesen und im Applikationsobjekt für den *OnRequest*-Handler der Seite `submit.htm` abgelegt. Anschließend wird aufgrund des betätigten Buttons die nächste aktuelle Station ermittelt und zur Seite `submit.htm` navigiert.

Der *OnRequest*-Handler der Station `station.htm` ist zentral für den Ablauf zuständig. Er ist wie folgt implementiert (Zeilen, die mit „*“ beginnen, sind Kommentarzeilen):

```
01 DATA: station TYPE REF TO z_epm_station,
02       ste TYPE zepm_service_table,
03       service TYPE REF TO z_epm_service,
04       idescr TYPE REF TO cl_instance_description,
04       app_context TYPE zepm_contexts,
05       s TYPE string.
06
07 * Get station
08 IF NOT csn IS INITIAL.
09   station = application->get_station( csn ).
10   application->currentstation = station.
11 ELSE.
12   station = application->currentstation.
13 ENDIF.
14
15 IF NOT station IS BOUND.
16   CONCATENATE 'Illegaler Stationsname "' csn '"' INTO s.
17   navigation->set_parameter( name = 'msg' value = s ).
18   navigation->goto_page( 'error.htm' ).
19 ELSE.
20   READ TABLE station->servicetable INDEX 1 INTO ste.
21   IF station->request NE 'X'.
```

```
22 *   redirect to submit station
23     navigation->goto_page( 'submit.htm' ).
24   ELSE.
25 *   release last service
26     IF application->last_service IS BOUND.
27       msg = application->last_service->status-message.
28       CALL METHOD application->last_service->('RELEASE').
29     ENDIF.
30
31 *   read context if required
32     IF NOT context IS INITIAL.
33       READ TABLE application->contexts
34             WITH KEY namespace = context_station
35                      number = context
36             INTO app_context.
37       IF sy-subrc NE 0.
38         WRITE context TO msg_debug LEFT-JUSTIFIED.
39         CONCATENATE 'Konnte Kontext' context_station
40          '(' msg_debug ') nicht lesen!'
41           INTO msg_debug SEPARATED BY space.
42         navigation->set_parameter( name = 'msg'
43                                    value = msg_debug ).
44         navigation->goto_page( 'error.htm' ).
45         RETURN.
46       ENDIF.
47       application->context = app_context-context.
48     ENDIF.
49
50 *   delete old contexts for this station
51     DELETE application->contexts
52      WHERE namespace = station->name.
54
55 *   reqeust service
56     CREATE OBJECT idescr EXPORTING the_subject = ste-service.
57     CREATE OBJECT service TYPE (idescr->class_name).
58     CALL METHOD service->('REQUEST')
59        EXPORTING
60          context = application->context
61          user = application->user
62        EXCEPTIONS
63          wrong_or_no_context_supplied = 1
64          OTHERS                       = 2.
65     IF service->status-status EQ '2'.
```

```
66       navigation->set_parameter( name = 'msg'
67                     value = service->status-message ).
68       navigation->goto_page( 'error.htm' ).
69       RETURN.
70     ENDIF.
71
72 *   get HTML representation
73     CLEAR trueconds.
74     CALL METHOD application->get_html_for_ie
75       EXPORTING
76         ie                 = ste-interaction_element
77         ie_value           = service->status-context
78         namespace_html     = ''
79         namespace_contexts = station->name
80         tabrow             = 'X'
81       IMPORTING
82         html               = html
83       CHANGING
84         trueconds          = trueconds.
85
86 *   set some variables
87     sname = station->name.
88     application->last_service = service.
89     application->context = service->status-context.
90
91   ENDIF.
92 ENDIF.
```

In den Zeilen 01-05 werden einige Hilfsvariablen deklariert. Die Variable `idescr` in Zeile 04 ist eine Referenz auf die Beschreibung einer Klasse, über die zur Laufzeit Informationen über die Klassenhierarchie erfragt werden können.

In den Zeilen 07-13 wird anhand des Seitenattributs[1] `csn` die aktuelle Station ermittelt und im Applikationsobjekt abgelegt, falls das Attribut übergeben bzw. gesetzt wurde. Ist das Attribut initial, wird die im Applikationsobjekt abgelegte Station als die aktuelle angenommen. In den Zeilen 15-18 wird zur Fehlerseite verzweigt, falls keine aktuelle Station gefunden werden konnte.

In Zeile 20 wird der Dienstplan der aktuellen Station ausgelesen. Handelt es sich um eine Ausführungsstation, dann wird zur Seite `submit.htm` navigiert (Zeilen 21-23). Ansonsten wird der Dienst letzte Dienst beendet (Zeilen 25-29).

Anschließend wird in den Zeilen 32-48 der *Kontext* hergestellt. Ein Kontext besteht aus einem Interaktionselement, meistens eine Liste mit weiteren Elementen. Ein Kontext wird

[1] vgl. Kapitel 2.2.5.

über den Namen der Station, an der er erzeugt wurde, und einer fortlaufenden Nummer eindeutig identifiziert (Zeilen 33-36). Diese Identifizierung der Kontexte ermöglicht es, die Kontexte auf dem Server zu belassen und nicht an den Client zu übertragen. Lediglich die Identifizierungsmerkmale werden übertragen. Hier besteht ein deutlicher Unterschied zur Laufzeitumgebung für den infoAsset Broker: Beim infoAsset Broker können aus den IDs der Kontexte in den (Broker-)Diensten die entsprechenden Informationen direkt gewonnen werden - es handelt sich bei den Kontexten ja um Assets, die anhand der Asset-ID eindeutig identifizierbar sind. Die (infoAsset Broker-)EPM-Laufzeitumgebung braucht sich entsprechend nicht um die Speicherung von Kontexten zu kümmern. Im Rahmen des SAP Web Application Servers gibt es kein systemweit eindeutiges Identifizierungsmerkmal für beliebig im System abgelegte Daten, entsprechend muss die (SAP-)EPM-Laufzeitumgebung dies Merkmal für die Verwaltung der Kontexte schaffen und die Daten selbst verwalten. Die Speicherung des Stationsnamens in der ID des Kontextes dient dabei dazu, diejenigen Kontexte zu identifizieren, die veraltet sind und deshalb gelöscht werden können (Zeilen 50-52). Da die Kontexte bei jedem Aufruf einer Station neu erzeugt werden, können vor der Erzeugung der neuen Kontexte die alten Kontexte mit diesem Stationsnamen (relativ) gefahrlos gelöscht werden. Über die anderen gespeicherten Kontexte kann keine Aussage getroffen werden, da ein Benutzer stets die Möglichkeit hat, über den „Zurück“-Button des Web-Browsers zu früheren Seiten zurückzukehren und dort einen anderen Verweis mit Kontexttransfer zu wählen, ohne, dass der OnRequest-Handler nochmals durchlaufen würde.

In den Zeilen 55-70 findet die eigentliche Anforderung statt. In Zeile 56 werden Klassen-Informationen über den im Dienstplan eingetragenen Dienst besorgt. Da in ABAP-Objects, zumindest in der Version 6.10 des SAP Web Application Servers, statische Methoden in abgeleiteten Klassen nicht überschrieben werden können, konnte die Anforderung eines Dienstes nicht wie in Java nur über die Methode `request` gelöst werden. Statt dessen wird ein Dienst zunächst instantiiert (Zeile 57) und dann erst angefordert (Zeilen 58-64). Treten bei der Anforderung eines Dienstes Fehler auf, so wird mit entsprechender Nachricht zur Fehlerseite verzweigt (Zeilen 65-70).

Mit dem Methodenaufruf `get_html_for_ie` des Applikationsobjekts in den Zeilen 72-84 wird aus dem im Dienstplan abgelegten Interaktionselement mit Hilfe der Werte im Kontext des Dienstes gültiger HTML-Code als Zeichenkette erzeugt. Die Variable `html` ist ein Seitenattribut, das im Layout der Seite mit Hilfe der Anweisung `<%= html%>` in den Körper der HTML-Seite eingebettet wird. Zusätzlich hat die Methode das Seitenattribut `trueconds` verändert, in dem die Namen aller erfüllten Bedingungen aufgezählt sind. Diese Aufzählung wird dann auf dem Client über einen JavaScript-Funktionsaufrufs an das EPM-Applet weitergereicht. Die Methode `get_html_for_ie` erzeugt als Nebeneffekt die durch den Kontext des Dienstes vorgegebenen Kontexte und speichert diese in der Tabelle `contexts` des Applikationsobjekts.

5.3.2 Die Schnittstelle zum EPM-Editor

Die Schnittstelle zwischen der EPM-Laufzeitumgebung und dem EPM-Editor funktioniert in beide Richtungen:

- Der Report `Z_EPM_EXPORT` erzeugt Konfigurationsdateien für den Import von Diensten, Bedingungen und Aufgaben in den EPM-Editor. Während die Konfigurationsdateien für Bedingungen und Aufgaben denen des infoAsset Brokers (Kapitel 5.2.2) ähneln, ist die Konfigurationsdatei „services.txt“ für Dienste anders aufgebaut. Ihr Aufbau ist Folgender:

  SAP-Klassenname = Interaktionselement

  Dabei wird das Interaktionselement durch die in Kapitel 4.8 angegebenen Produktionsregeln für Interaktionselemente beschrieben. Die Java-Klasse `SAPBridge` im Paket `epm.sap` erzeugt daraus entsprechende Stellvertreter-Objekte für den EPM-Editor.

- Die Methode `exportMap` der Klasse `SAPBridge` erzeugt aus der Designer-Sicht einer Enterprise Portal Map eine Textdatei, die von der EPM-Laufzeitumgebung interpretiert wird.

- Die Methode `exportUserMap` der Klasse `SAPBridge` erzeugt ein Bild der Benutzersicht im JPEG-Format für die Site-Map und eine Textdatei für das EPM-Applet. Beide Dateien werden als MIME-Objekte im MIME-Repository des SAP Web Application Servers abgelegt.

- Die Methode `create_map` der Klasse `Z_EPM_APPLICATION` liest die von der Methode `exportMap` erzeugte Textdatei ein und interpretiert diese als Enterprise Portal Map. Dabei werden Instanzen der in Abbildung 5.10 (Seite 120) dargestellten Klassen erzeugt und miteinander verknüpft.

5.3.3 Kapselung der SAP-Funktionen

Wie bei der Laufzeitumgebung für den infoAsset Broker, werden auch bei SAP Web Application Server die eigentlichen Funktionen und Zugriffe in „EPM-Kapsel“ verpackt und damit standardisiert. Diese Standardisierung ermöglicht die einheitliche Verwendung und Kombination von Informationsartefakten des SAP Web Application Servers, die sich bisher nur durch explizite Programmierung verbinden ließen. Die Kapselung von Bedingungen soll anhand der Methode `check` des Bedingung `Z_EPM_C_BOOKING_RESERVED` aufgezeigt werden. Die Bedingung prüft, ob eine Flugbuchung nur reserviert wurde:

```
01 METHOD check .
02   DATA: carrid TYPE sbook-carrid,
03         bookid TYPE sbook-bookid,
04        sbook_wa TYPE sbook,
05         ie TYPE REF TO z_epm_interaction_element,
06        iesv TYPE REF TO z_epm_iesinglevalue.
07
08 * Parameter aus Kontext lesen
09   CALL METHOD context->('GETIEBYNAME')
```

```
10     EXPORTING
11       sname  = 'CARRID'
12     IMPORTING
13       RESULT = ie.
14   iesv ?= ie.
15   carrid = iesv->value.
16   CALL METHOD context->('GETIEBYNAME')
17     EXPORTING
18       sname = 'BOOKID'
19     IMPORTING
20       RESULT = ie.
21   iesv ?= ie.
22   bookid = iesv->value.
23
24 * Buchung auslesen.
25   SELECT bookid FROM sbook
26     INTO sbook_wa-bookid UP TO 1 ROWS
27    WHERE carrid = carrid
28      AND bookid = bookid
29      AND customid = user
30      AND reserved <> space
31      AND cancelled = space.
32   ENDSELECT.
33
34 * Ergebnis festlegen
35   IF sy-subrc EQ 0.
36    result = 'X'.
37   ELSE.
38     CLEAR result.
39   ENDIF.
40 ENDMETHOD.                    "
```

In den Zeilen 08-22 werden aus dem übergeben Kontext (ein Objekt der Klasse `Z_EPM_INTERACTION_ELEMENT`) die Einzelwerte `CARRID` und `BOOKID` ausgelesen, die als Schlüssel für den Zugriff auf die Buchungsdaten in den Zeilen 25-32 dient. In den Zeilen 35-39 wird das Ergebnis in das Zeichenfeld `result` zurückgeschrieben, wobei der Wert „X“ für wahr steht und das leere Zeichen für falsch[2].

Die Implementierung der Bedingungen erfolgte noch weitgehend „zu Fuß“, da SAP in diesem Falle keine ausreichend gekapselten Funktionen zur Verfügung stellt und somit nur der unmittelbare Durchgriff auf die Tabelle im R/3 bleibt. Anders sieht hingegen die Situation bei den Diensten aus: Hier kann sehr effizient und generisch auf BAPIs

[2]Dies ist die übliche Abbildung der boolschen Werte unter SAP R/3.

zurückgegriffen werden, die nur eine kleine Kapselung benötigen. Folgender Auszug aus der `request`-Methode des EPM-Dienstes `Z_EPM_SFLIGHT_SHOW_DETAIL` zeigt dies:

```
01 METHOD request .
02   DATA: ...
03
04 * get BAPI-Parameters
05   CALL METHOD context->('GETIEBYNAME')
06      EXPORTING
07        sname  = 'AIRLINEID'
08      IMPORTING
09        RESULT = ie.
10   iesv ?= ie.
11   key-airlineid = iesv->value.
     ...
12 * call BAPI
13   CALL FUNCTION 'BAPI_FLIGHT_GETDETAIL'
14     EXPORTING
15       airlineid       = key-airlineid
16       connectionid    = key-connectid
17       flightdate      = key-flightdate
18     IMPORTING
19       flight_data     = f_data
20       additional_info = add_inf
21       availibility    = avail
22     TABLES
23       return          = return.
24
25   LOOP AT return INTO return_wa.
26     IF return_wa-type EQ 'E'.
27       status-status = '2'.
28       status-message = return_wa-message.
29       RETURN.
30     ELSEIF return_wa-type EQ 'S'.
31       status-message = return_wa-message.
32     ENDIF.
33   ENDLOOP.
34
35 * generate approriate IE
36   DATA: ies TYPE z_epm_ies.
37   list ?= interaction_element->copy( ).
38   ies = get_ie_for_struct( struct_name = 'BAPISFLDAT' value = f_data ).
39   APPEND LINES OF ies TO list->members.
```

```
40   ies = get_ie_for_struct( struct_name = 'BAPISFLADD' value = add_inf ).
41   APPEND LINES OF ies TO list->members.
42   ies = get_ie_for_struct( struct_name = 'BAPISFLAVA' value = avail ).
43   APPEND LINES OF ies TO list->members.
44   status-context = list.
45   status-status = '0'.
46 ENDMETHOD.                    "
```

Zunächst werden in der Zeile 04ff. Werte für die Aufrufparameter `airlineid`, `connectionid` und `flightdate` aus dem übergebenen Interaktionselement gelesen. Anschließend wird in den Zeilen 13-23 der BAPI-Baustein aufgerufen. In den Zeilen 25 bis 33 werden die vom BAPI gelieferten Rückmeldungen in den Status des Dienstes übernommen. Verlief der BAPI-Aufruf fehlerfrei, dann wird in den Zeilen 36-43 aus den ABAP-Strukturen mit Hilfe der Methode `get_ie_for_struct` des Objekts `Z_EPM_SERVICE` eine interne Tabelle mit entsprechenden Interaktionselementen (Einzelwerten) dynamisch aus den Laufzeit-Typ-Informationen des ABAP-Repository erzeugt. Die Einträge der internen Tabelle werden dann der Liste hinzugefügt (Zeilen 39, 41 und 43). Die Liste selbst wird als neuer Kontext im Status des Dienstes abgelegt (Zeile 44).

Eine noch generischere Abbildung der BAPIs auf EPM-Dienste ist sicherlich möglich, da das Prinzip der BAPIs recht kompatibel zum Modell der EPM-Dienste ist.

5.3.4 Die Benutzungsoberfläche

Die implementierte Benutzungsoberfläche ähnelt stark der in Kapitel 5.2.3 beschriebenen Benutzungsoberfläche für den infoAsset Broker. Die in Abbildung 5.12 gezeigt Benutzungsoberfläche teilt sich in zwei Bereiche auf:

- Das EPM-Applet im oberen Drittel. Es ist das selbe, das auch in Verbindung mit dem infoAsset Broker zum Einsatz kommt (vgl. Kapitel 5.2.3). Auch die Funktionalitäten, insbesondere die Animation und das Zusammenspiel mit der interaktiven Site-Map sind identisch.

- Das Hauptfenster.

Das Hauptfenster besteht aus drei horizontal angeordneten Teilen:

1. Dem Titel der Seite, der dem Stationsnamen der aktuellen Station entspricht. Im Beispiel „Meine Buchungen“.

2. Dem eigentlichen Hauptteil, der dem von der Methode `get_html_for_ie` erzeugten HTML-Code entspricht. Im Beispiel eine Tabelle.

3. Dem Fußteil, der Informationen über die Benutzerkennung angibt, wenn ein Benutzer eingeloggt ist. Im Beispiel ist die Kennung des aktuellen Benutzers „336“.

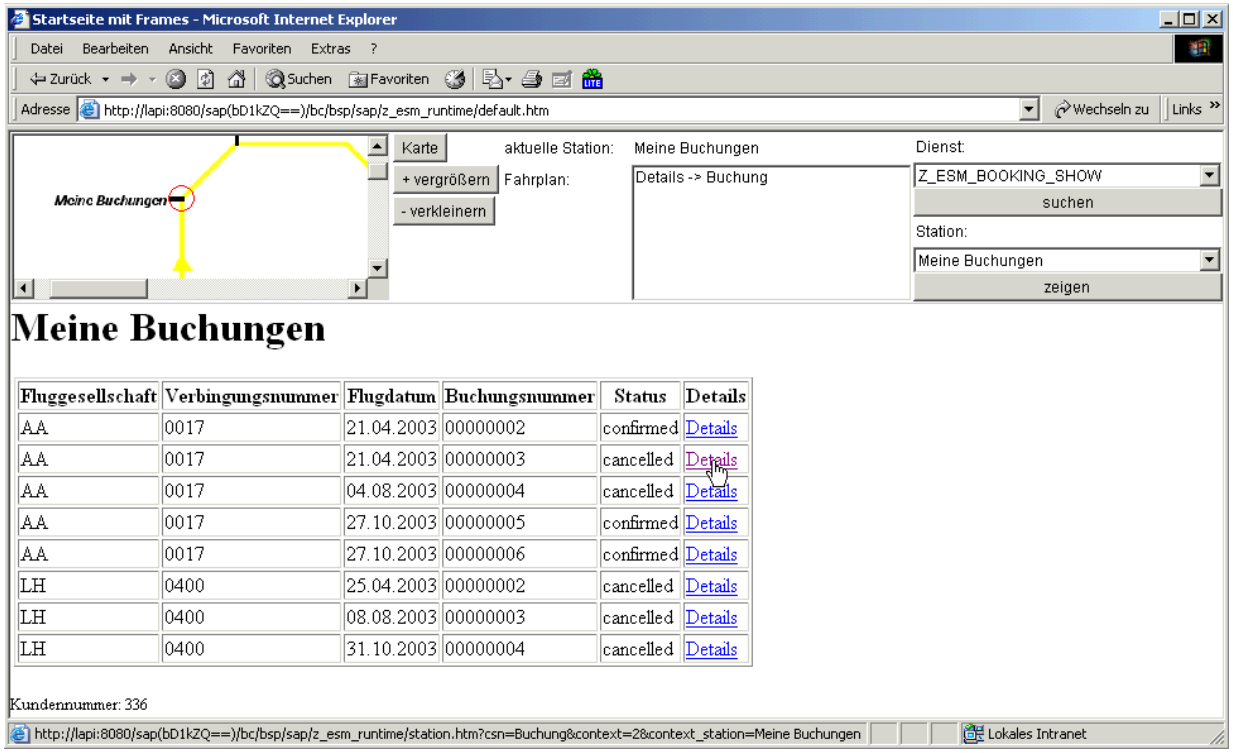

Abbildung 5.12: Die SAP-EPM-Benutzungsoberfläche mit Applet.

In der Status-Anzeige des Web-Browsers in Abbildung 5.12 (ganz unten) ist noch etwas weiteres zu erkennen: Der Mauszeiger schwebt über einem Verweis der einen Kontext transportiert, die resultierende URL hat drei Parameter: Den Parameter `csn`, der die nächste Station bestimmt, in diesem Falle die Station „Buchung“, wie auch aus dem angezeigten Fahrplan im Applet deutlich wird. Der zweite Parameter lautet `context` und gibt die Nummer des gewünschten Kontextes an. Der dritte Parameter lautet `context_station` und identifiziert gemeinsam mit den Parameter `context` den Kontext eindeutig (vgl. Kapitel 5.3.1).

5.3.5 EPM für Flugbuchungen

Nachdem die Implementierung des Modells der Enterprise Portal Maps für den SAP Web Application Server vorgestellt wurde, soll im folgenden Teil ein Anwendungsbeispiel, also ein konkretes Portal mit seinen enthaltenen Diensten, vorgestellt werden. Das Beispiel stützt sich auf das SAP-übliche Schulungsbeispiel zur Buchung von Flügen, da es wohl den meisten ABAP-Programmieren geläufig ist und viele Aspekte geschäftskritischer Anwendungen berührt.

Schulungsbeispiel Flugbuchungen

Für das Schulungsbeispiel der Flugbuchungen werden im SAP Web Applikation Server 6.10 drei Geschäftsobjekte, engl. *Business Objects*, ausgeliefert, die sich wie folgt beschreiben lassen[3]:

1. Flight
   - *Definition*: Das Business-Objekt Flug ist eine Dienstleistung einer Fluggesellschaft, mit der Kunden zu einer bestimmten Zeit (Datum und Uhrzeit) von einem Startort zu einem Zielort per Flugzeug befördert werden.
   - *Struktur*: Ein Flug wird durch die Fluggesellschaft, die Flugnummer und das Flugdatum identifiziert. Eigenschaften eines Fluges sind z.B. Abflugort, Ankunftsort, Abflugzeit, Ankunftszeit, Ticketpreis und Währung, Plätze insgesamt und freie Plätze (jeweils für die Flugklassen Economy Class, Business Class und First Class).
2. FlightCustomer
   - *Definition*: Das Business-Objekt Flugkunde ist ein Geschäftspartner, der bei einem Reisebüro oder einer Fluggesellschaft Flugbuchungen durchführen kann.
   - *Struktur*: Ein Flugkunde wird durch eine Kundennummer eindeutig identifiziert. Eigenschaften eines Flugkunden sind z.B. Name des Flugkunden, Anrede, Straße, Postfach, Postleitzahl, Ort, Länderschlüssel, Region, Telefonnummer eines Flugkunden, E-Mail-Adresse eines Flugkunden, Kundentyp, Rabattsatz und Sprachenschlüssel.
3. FlightBooking
   - *Definition*: Das Business-Objekt Flugbuchung ist eine Platzreservierung für einen Flug bei einer Fluggesellschaft.
   - *Struktur*: Eine Flugbuchung wird durch die Fluggesellschaft und die Buchungsnummer identifiziert. Eigenschaften einer Flugbuchung sind z.B. die Fluggesellschaft, die Flugnummer, die Kundennummer des buchenden Kunden, Flugklasse, Nummer der Verkaufsstelle oder Nummer des Reisebüros, Stornierungskennzeichen, Reservierungskennzeichen und Name des Passagiers.
   - *Bemerkung*: Eine Flugbuchung kann einen der drei folgenden Status besitzen:
     - „gebucht“, d.h. Buchung ist gültig und rechtsverbindlich,
     - „storniert“, d.h. Buchung wurde zurückgenommen,
     - „reserviert“, d.h. es wird ein Platz für einen Kunden für eine bestimmte Zeit freigehalten.

[3]Die Beschreibung der Objekte *Flight* und *FlightBooking* wurden dem *Business Object Repository* (BOR) entnommen.

Wird bei einer reservierten Buchung die Buchung nicht in den Status „gebucht“ überführt, so wird diese nach Ablauf einer Frist von der Fluggesellschaft automatisch storniert.

Diese Geschäftsobjekte verfügen über Methoden (BAPIs), mit Hilfe derer Geschäftsvorfälle abgewickelt werden können. Abbildung 5.13 zeigt einen Bildschirmausschnitt aus dem Business Object Repository mit den Geschäftsobjekten, ihren Schlüsselfeldern und den Methoden. Geschäftsobjekte werden dabei mit einer farbigen Scheiben symbolisiert. Scheiben mit vier kleinen Kästchen in der linken unteren Ecke markieren Schlüsselfelder. Methoden werden mit Scheiben und einem kleinen Pfeil dargestellt, wobei statische Methoden zusätzlich mit einen Kasten um die Scheibe hervorgehoben werden.

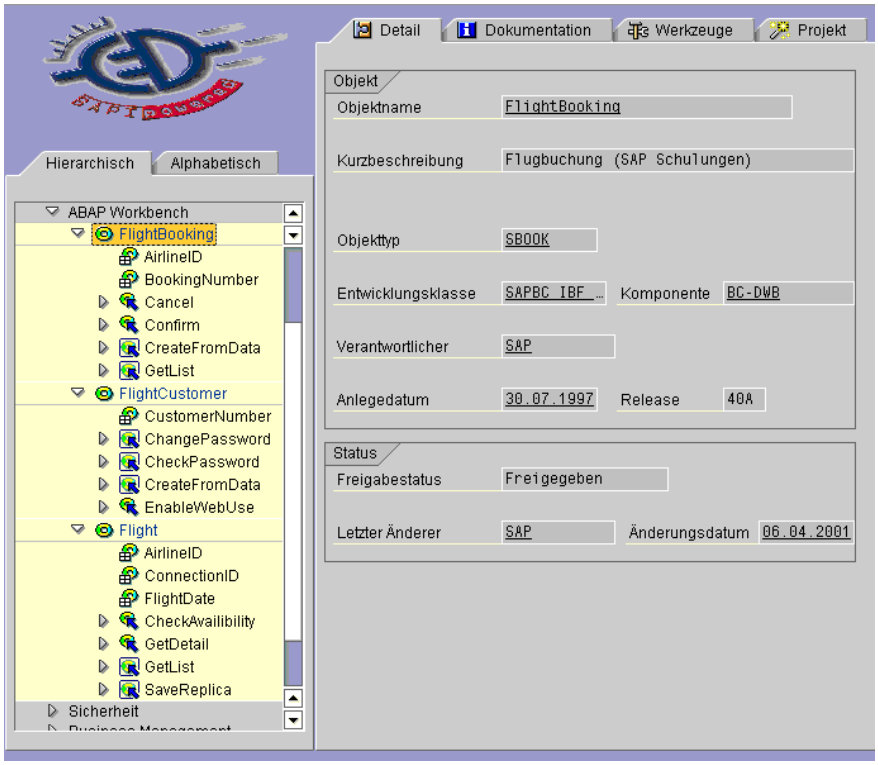

Abbildung 5.13: Flugbuchungen: Geschäftsobjekte, ihre Schlüsselfelder und Methoden.

Die Methoden der Geschäftsobjekte sind als Funktionsbausteine implementiert. Instanz-abhängige Methoden erwarten als Aufrufparameter mindestens die Schlüsselfelder des Objekts, vergleiche den BAPI-Aufruf in Kapitel 5.3.3 im Quelltext des EPM-Dienstes `Z_EPM_SFLIGHT_SHOW_DETAIL` Zeilen 13-17.

EPM für Flugbuchungen im Internet

Für die Erstellung eines Portals zur Buchung von Flügen im Internet wurden folgende Dienste, abgeleitet von der Klasse `Z_EPM_SERVICE`, im Paket `Z_EPM_TEST` implementiert:

EPM-Dienst	Beschreibung	Typ	BAPI
Z_EPM_BOOKING_SHOW	Flugbuchung anzeigen	R	-
Z_EPM_BOOKING_SHOW_ALL	Alle Buchungen eines Fluggastes anzeigen	R	BAPI_FLBOOKING_GETLIST
Z_EPM_CREATE_CUSTOMER	Neuen Kunden anlegen	RS	BAPI_FLCUST_CREATEFROMDATA
Z_EPM_LOG_IN	Log-in für Fluggäste	RS	BAPI_FLCUST_CHECKPASSWORD
Z_EPM_LOG_OUT	Log-out für Fluggäste	S	-
Z_EPM_SFLIGHT_BOOK	Buchen eines Fluges	RS	BAPI_FLBOOKING_CREATEFROMDATA
Z_EPM_SFLIGHT_CANCEL	Buchung stornieren	S	BAPI_FLBOOKING_CANCEL
Z_EPM_SFLIGHT_CONFIRM	Buchung bestätigen	S	BAPI_FLBOOKING_CONFIRM
Z_EPM_SFLIGHT_SEARCH	Flugsuche	RS	BAPI_FLIGHT_GETLIST
Z_EPM_SFLIGHT_SEARCH_RES_LIST	Flugliste anzeigen	R	-
Z_EPM_SFLIGHT_SHOW_DETAIL	Einzelheiten eines Fluges anzeigen	R	BAPI_FLIGHT_GETDETAIL

Drei dieser elf Dienste verwendenden aus unterschiedlichen Gründen keine BAPIs.

- Für den Dienst Z_EPM_LOG_OUT kann es kein BAPI geben, da sich ein Benutzer (nur) am Portal „Flugbuchung“ anmeldet, nicht aber am SAP Web Application Server. Die Anmeldung am SAP Web Application Server geschieht automatisch beim ersten Aufruf des Portals mittels eines festgelegten Standard-Benutzers.

- Die Visualisierung der Flugliste mittels des Dienstes Z_EPM_SFLIGHT_SEARCH_RES_LIST benötigt keinen BAPI-Aufruf, da die Liste ja das Ergebnis der Ausführung des Dienstes Z_EPM_SFLIGHT_SEARCH ist. Der Dienst Z_EPM_SFLIGHT_SEARCH_RES_LIST verwendet entsprechend nur den bei der Anforderung übergebenen Kontext.

- Der Dienst Z_EPM_BOOKING_SHOW verwendet keinen BAPI-Aufruf, da dem Geschäftsobjekt *FlightBooking* die entsprechende Methode zur Anforderung einer speziellen Buchung fehlt. Alternativ zum implementieren direkten Durchgriff auf die Datenbanktabelle SBOOK wäre der Aufruf des BAPIs BAPI_FLBOOKING_GETLIST und die anschließende Filterung der gesuchten Buchung anhand der Buchungsnummer möglich.

Zusätzlich zu den Diensten wurden mehrere Bedingungen implementiert, die Berechtigungen und Voraussetzungen für Dienste umsetzten. Mit 15 Stationen setzt die in Ab-

bildung 5.14 dargestellte Enterprise Portal Map ein Portal für Flugbuchungen im Internet um. Abbildung 5.12 (Seite 130) zeigt eine Seite des Portals aus Benutzersicht.

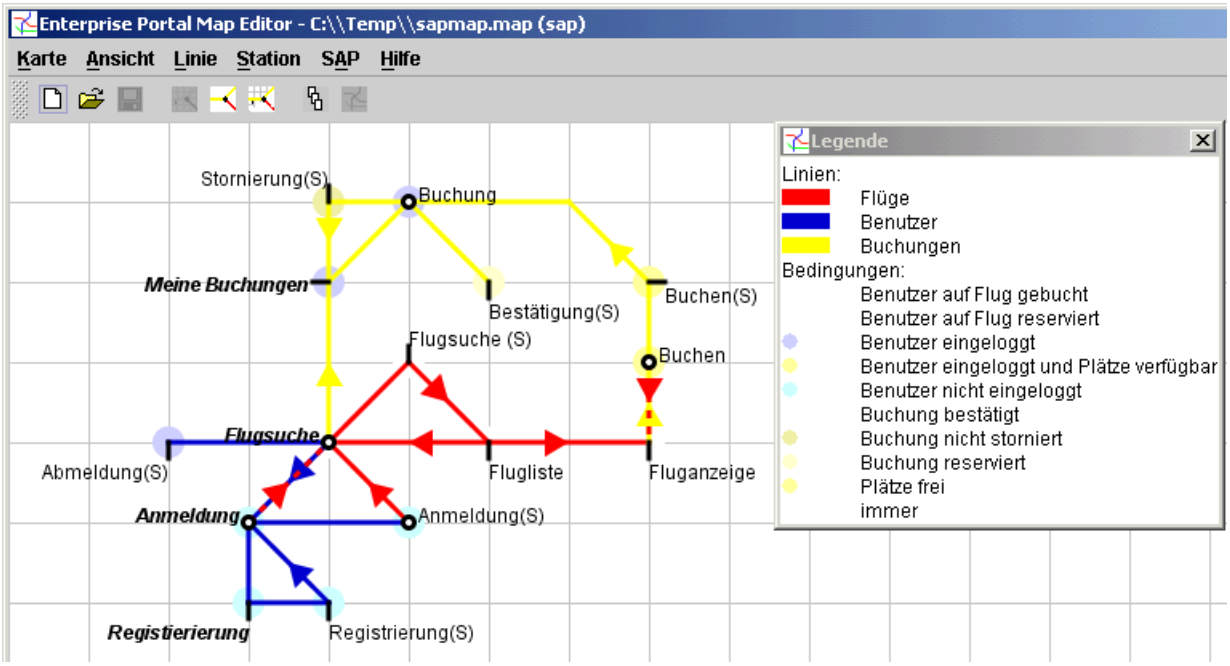

Abbildung 5.14: Designersicht auf eine Enterprise Portal Map für Flugbuchungen im Internet.

Es existieren die drei Linien *Flüge*, *Benutzer* und *Buchungen*. Von der Startseite „Flugsuche“ aus kann sich ein Benutzer über die Linie *Flüge* verfügbare Flüge ansehen und ggf. über die Linie *Buchungen* einen Flug buchen, reservieren oder eine Buchung stornieren. Über die Linie *Benutzer* kann sich ein Benutzer einloggen - die Voraussetzung, damit die Linie *Buchungen* befahren werden kann - ausloggen oder als neuer Benutzer registrieren.

Die Stationsnamen lassen einen unmittelbaren Rückschluss auf den Dienst an der Station zu, wobei ein angehängtes „(S)“ darauf hindeutet, dass der Dienst ausgeführt wird.

5.4 Zusammenfassung

In diesem Kapitel wurde die Implementierung und die Anwendung des Modells der Enterprise Portal Maps vorgestellt.

Für die Erprobung des Konzepts wurde zunächst ein graphischer Editor entwickelt, mit dessen Hilfe beliebige Enterprise Portal Maps, aufbauend auf dem Modell aus Kapitel 4, definiert werden können. Die Benutzersicht wird automatisch aus der Designersicht gewonnen, es bleiben u.U. nur einige optische Korrekturen an der Streckenführung übrig. Das Definitionswerkzeug ist so angelegt, dass es für mehrere verschiedene Systeme für Portale und Dienste einsetzbar ist. Die allgemeine Schnittstelle zwischen Editor und Portalsystem

besteht einerseits aus dem Import vorhandener Dienste, Bedingungen und Aufgaben in den Editor und andererseits aus dem Export der erzeugten Enterprise Portal Map, so dass die EPM-Laufzeitumgebung in dem Portalsystem die Enterprise Portal Map sofort ausführen kann.

Im Anschluss wurden zwei EPM-Laufzeitumgebungen und Anwendungen, basierend auf verschiedenen Portalsystemen, vorgestellt.

Das erste Portalsystem ist der infoAsset Broker, dessen Anfragebearbeitung sich ohne Veränderung des Kernsystems so anpassen ließ, dass Enterprise Portal Maps unmittelbar ausgeführt werden können. Die EPM-Laufzeitumgebung für den infoAsset Broker nutzt die Tatsache aus, dass der infoAsset Broker und der EPM-Editor beide in der Sprache Java implementiert wurden. Die Schnittstellen zwischen beiden Systemen konnten so sehr einfach gehalten werden, da beide auf denselben Wrapper-Klassen für Funktionen des infoAsset Brokers arbeiten.

Die Standard-Benutzungsoberfläche des infoAsset Brokers wurde so verändert, dass die einem Portal zu Grunde liegende Enterprise Portal Map über ein eigens entwickeltes Applet auch für die Anwender nutzbar ist. Dabei erhält der Benutzer sowohl eine Orientierungshilfe, wo er sich im Moment befindet, als auch Hinweise, welche Stationen als nächstes von ihm/ihr besucht werden können und verschiedene Suchmöglichkeiten nach speziellen Diensten oder Stationen.

Als Anwendung wurde eine konkrete Enterprise Portal Map mit 42 Stationen vorgestellt, die ein vollständiges Portal zum Wissensmanagement umsetzt. Dabei bildeten die Informationsobjekte, die Berechtigungskonzepte und Anwendungsfälle der Standard-Auslieferung des infoAsset Brokers als System zum Wissensmanagement in Unternehmen die Grundlage.

Das zweite Portalsystem ist der SAP Web Application Server. Die EPM-Laufzeitumgebung wurde als BSP-Applikation realisiert, die aus insgesamt sieben Seiten besteht. Die zentrale Seite ist dabei die Seite `station.htm`, die wesentlich den Ablauf steuert und zur Visualisierung von Anforderungsstationen dient.

Da aufgrund der verschiedenen Programmiersprachen keine gemeinsame Nutzung von Objekten zwischen Editor und Laufzeitumgebung möglich war, wurden auf beiden Seiten eigene Stellvertreterklassen erstellt, die sich jeweils auf den Definitions- bzw. Ausführungsaspekt beschränkten. Über die in Kapitel 4.8 vorgestellt Grammatik wurden Informationen über Dienste, Bedingungen, Aufgaben und Netzpläne ausgetauscht. Die Kapselung der SAP-Funktionen wurde über EPM-Dienste realisiert, die im Wesentlichen aus den Kontextinformationen die notwendigen Aufrufparameter für BAPIs gewinnen und die Ergebnisse des BAPI-Aufrufs wieder als Kontextinformationen, angereichert mit Statusinformationen, ablegen.

Die Benutzungsoberfläche für das EPM-SAP-Portal gleicht der Oberfläche, wie sie für den EPM-infoAsset Broker geschaffen wurde. Es kommt das identische Applet zum Einsatz.

Als Anwendung wurde eine konkrete Enterprise Portal Map mit 15 Stationen vorgestellt, die das SAP-typische Schulungsbeispiel für Flugbuchungen im Internet realisiert.

Kapitel 6

Bewertung und Ausblick

Zum Abschluss dieser Arbeit wird das in Kapitel 4 vorgestellte Modell der Enterprise Portal Maps und seine in Kapitel 5 beschriebenen Implementierungen daraufhin untersucht, inwieweit es die in Kapitel 2.2.2 genannten Schwächen bisheriger Ansätze für Portale und Dienste erfüllt und wo der Nutzen und die Grenzen der gewählten Metapher liegen. Anschließend werden Erfahrungen mit der Notation der Netzpläne dargelegt, die während des gesamten Entstehungszeitraums dieser Arbeit gemacht wurden. Abschließend wird ein kurzer Ausblick gegeben, der Hinweise für zukünftige Arbeiten und Forschungsrichtungen geben kann.

6.1 Bewertung des Ansatzes

Die Bewertung des Ansatzes erfolgt im Hinblick auf zwei Aspekte: Zum Einen, inwieweit die aufgestellten Anforderungen bei der Gestaltung von Portalen erfüllt werden, zum Anderen, inwieweit sich die Metapher als solche bewährt hat.

6.1.1 Erfüllung der Anforderungen

Die Bewertung des Modells der Enterprise Portal Maps findet anhand der in Kapitel 2.2.2 aufgestellten Anforderungen bei der Gestaltung von Portalen statt. Im Einzelnen bedeutet dies:

- *Für die Benutzer eines Portals muss die Struktur eines Portal und die enthaltenen Dienste auf möglichst einfache, nicht-technische Art vermittelt werden.*

  Das EPM-Modell mit seiner Metapher von Stationen und Linien und der sehr bekannten und in der Praxis sehr gut verstandenen grafischen Notation erfüllt diese Forderung. Das bedeutet nicht, dass ein Benutzer sich nicht zuerst mit dem Netzplan etwas vertraut machen muss, um die gebotenen Möglichkeiten zu verstehen. Bei Tests begannen Benutzer nach einer kurzen Eingewöhnungszeit sehr schnell, die Seiten eines Portals mit den Stationen zu verknüpfen. Aufgrund dieser Zuordnung bekamen

Benutzer mit Hilfe des Netzplans den Eindruck, dass das Portal eine räumliche Struktur habe. Das EPM-Applet mit seiner Anzeige der aktuellen Station verhinderte, dass sich Benutzer „verloren“ fühlten, zumindest konnten Benutzer, wenn sie sich auf einer ihnen unbekannten und nicht sofort einsichtigen Seite befanden, jederzeit über die Site-Map, also den Netzplan, zurück zu bekannten Seiten navigieren. Die Bedeutung der Anzeige der aktuellen Station auf dem Netzplan durch das EPM-Applet nahm mit zunehmendem Kenntnisstand des Benutzers ab, da der Netzplan bei häufiger Verwendung verinnerlicht und nur noch bei der Navigation zu bisher unbekannten Stationen benötigt wurde.

- *Die Designer eines Portals, die der fachlichen Führungsebene in Unternehmen zuzurechnen sind, benötigen ein Modell, das die angebotenen Dienste, die vergebenen Berechtigungen (die Personalisierungen auf der Ebene des Portalsystems) und die Navigationsstruktur deutlich und übersichtlich darstellt, ohne auf die Ebene der Implementierung zu wechseln.*

  Das Modell der Enterprise Portal Maps mit seiner eins-zu-eins Zuordnung von einer Station zur Anforderung oder Ausführung eines Dienstes ermöglicht es, bei entsprechender Benennung der Stationen einen sofortigen Überblick über die angebotenen Dienste zu erhalten. Auch die Bedingung, die erfüllt sein muss, damit die Navigation zu einer bestimmten Station zulässig ist, wird durch die Farbcodierung auf dem Netzplan sofort erkennbar. Die möglicherweise komplizierte Implementierung der Bedingung bleibt hinter dem Namen verborgen und ist für den Portal-Designer uninteressant, solange der Name hinreichend Aufschluss über die Semantik gibt. Die detaillierten Zuordnung, welche Teile eines Informationsobjektes oder welche Eingabefelder unter welchen Bedingungen an einer Station sichtbar oder änderbar sind, wird jeweils lokal, also unmittelbar bei dem betreffenden (Interaktions-)Element festgelegt. Durch die Vermeidung von globalen Anpassungsregeln, wie sie beispielsweise bei WebML existieren, bleiben die Personalisierungen auf der Ebene des Portalsystems vorhersehbar, d.h. dass es keine „plötzlichen“ Personalisierungen auf Grund von Nebeneffekten globaler Regeln geben kann. Durch die erreichte Transparenz des definierten Portals können später auch die Folgen von Änderungen an der Navigationsstruktur oder den Bedingungen besser abgeschätzt werden.

- *Entwickler benötigen ein Modell, das klare Schnittstellen für die Integration von Diensten, Berechtigungskonzepten und Personalisierungsmöglichkeiten auf der Ebene des Portalsystems definiert.*

  Das EPM-Modell definiert die geforderten Schnittstellen für die Dienste in Form der einfachen request/submit-Semantik und der Interaktionselemente, die die Datenschnittstelle darstellen. Das einzige, was der Entwickler eines Dienstes über die Laufzeitkomponente und damit indirekt über das Portalsystem wissen muss, ist, wie er aus dem übergebenen Kontext und der übergebenen Benutzer-Id die für den Dienst benötigten Informationen erhalten kann. Das Berechtigungskonzept und die Personalisierungsmöglichkeiten auf der Ebene des Portalsystems werden durch die Bedingun-

gen umgesetzt, die eine feste Schnittstelle haben und unabhängig von irgendwelchen Kontrollflüssen sind. Ein Entwickler kann sich also auf die tatsächliche Implementierung bzw. Integration von Diensten und Bedingungen konzentrieren, ohne dass er sich Gedanken über die konkrete Verwendung im Rahmen einer Enterprise Portal Map machen müsste.

- *Web-Designer benötigen ein Modell, das es ihnen erlaubt, sich ausschließlich auf die grundsätzlichen Fragen des Erscheinungsbildes der Seiten zu konzentrieren.*

  Die durchgehende Modellierung der zu erzeugenden Seiten über Interaktionselemente ermöglicht eine Generierung der Portal-Seiten, wie dies in den Kapiteln 5.2.3 und 5.3.4 beschrieben wurde. Für die Generierung können Vorlagen verwendet werden, wodurch ein durchgängiges Erscheinungsbild aller Seiten gewährleistet wird. Auch lassen sich die Portal-Seiten leicht neu erzeugen, falls sich die Vorlagen geändert haben. Web-Designer können sich also auf die Gestaltung der Vorlagen für die Seitengenerierung konzentrieren.

- *Alle Gruppen gemeinsam benötigen ein Modell, das ihnen hilft, die Probleme und Anliegen der anderen Gruppen besser zu verstehen.*

  Da es sich bei dem Modell der Enterprise Portal Maps um ein durchgängiges Modell handelt mit dem alle Gruppen arbeiten, dient es so auch der gemeinsamen Sprachbildung, eine wesentliche Voraussetzung für eine gute Kommunikation.

- *Das Modell muss allen Beteiligen Arbeit abnehmen und nicht Zusätzliche aufbürden.*

  Wie bei den Implementierungen in Kapitel 5 deutlich wurde, ist das Modell der Enterprise Portal Maps weit mehr als ein Modell – eine Enterprise Portal Map beschreibt ein Portal derart, dass das Portal automatisch generiert zur Verfügung steht, inklusive Dokumentation und Orientierungshilfe für die Endanwender. Dies ist eine erhebliche Arbeitserleichterung.

Aus einem etwas technischerem Blickwinkel betrachtet beantwortet das Modell der Enterprise Portal Maps die Frage „wem wann was wo im Portal zur Verfügung stellt wird“ zentral mit einem Modell, das nur aus wenigen, einfachen Elementen besteht. Dennoch bietet das Modell für jede beteiligte Gruppe des Entwicklungs- und Anwendungsprozesses die Lösung einiger konkreter Probleme, vor denen die Gruppen jeweils stehen. Es findet eine weitere *Separation of Concerns* bezogen auf die beteiligten Gruppen statt, und gleichzeitig wird eine gemeinsame Kommunikationsbasis geschaffen.

Speziell auf den infoAsset Broker bezogen, ergeben sich durch den Einsatz des Modells der Enterprise Portal Maps folgende, zusätzliche Vorteile:

- Die Programmierung neuer Handler ist überflüssig. Damit entfällt ein aufwändiger Arbeitsschritt bei der Anpassung des infoAsset Brokers für einen speziellen Anwendungsfall.

- Die aufwändige manuelle Erstellung der HTML-Templates entfällt.

- Das Berechtigungskonzept ist nicht mehr fest in den Handlern verdrahtet, sondern kann deklarativ auf die jeweiligen Erfordernisse angepasst werden.

- Es wurde „nebenbei“ ein durchgängiges Sperrkonzept für Assets geschaffen, das es vorher nicht gab. Der Programmierer eines EPM-Dienstes entscheidet nur durch die Parametrisierung eines Konstruktoraufrufs, ob für die Anforderung des Dienstes eine Sperre benötigt wird oder nicht. Alles Weitere wird automatisch durch die EPM-Laufzeitumgebung veranlasst.

In Bezug auf den SAP Web Application Server ergeben sich ähnliche Vorteile:

- Die Programmierung von BSP-Seiten oder Controllern entfällt völlig.

- Portale erhalten ein völlig neues Berechtigungs- und Konsitenzkonzept, das es bei bisher in dieser Form nicht gab, aber notwendig ist. Es stellt sich als zusätzliches Berechtigungskonzept neben dem (alten) Berechtigungskonzept des R/3 und dem des SAP Portals Server dar.

- Die Nutzung der BAPI-Funktionen über eine Web-Oberfläche kann mit nur minimalen, einmaligem Programmieraufwand erreicht werden. Der Aufwand, einen BAPI-Aufruf in einem EPM-Dienst zu kapseln ist erheblich geringer, als eine eigene BSP-Seite hierfür zu entwickeln, die zusätzlich nur bedingt wiederverwertbar wäre.

6.1.2 Nutzen und Grenzen der Metapher

> „Metaphors allow us to understand one thing in terms of another, without thinking the two are the same“[LoJo80].

Die Übertragung von Notationen und Begriffen aus dem Bereich des öffentlichen Personennahverkehrs birgt zwei wesentliche Vorteile:

1. Die meisten Menschen kennen sich im ÖPNV und mit der Notation der Netzpläne einigermaßen gut aus und können sich orientieren.

2. Die Notation der Netzpläne skaliert auch für größere Netze mit hunderten von Stationen[1]

Die Übertragung der ÖPNV-Konzepte auf Portale und Dienste gelang auch weniger geübten Benutzern sehr gut, wie weiter oben beschrieben. Die Verwendung der Begriffe „Station“ oder Linie statt „Seite“ oder „Link“ bereitete keine größeren Schwierigkeiten.

Die Darstellung der Enterprise Portal Maps skaliert mindestens so gut wie die Darstellung im ÖPNV, da die EPM-Darstellung über eine etwas geringere Anzahl wahrnehmbarer Elemente auf der Karte verfügt, wie folgende Tabelle verdeutlicht:

[1] Vgl. etwa den Londoner oder Berliner Netzplan.

visuelles Element	**auf ÖPNV-Karte**	**auf EPM-Karte**
einfache Station (Tick)	ja	ja
Umsteigestation als Kreis	ja	ja
Strecken als Teil von Linien	ja	ja
Anzeige des aktuellen Standorts	ja	ja
Tarifzonendarstellung	ja, als aneinander grenzende Flächen	ja, als Kreise um die Stationen
Geographische Orientierungspunkte (Flüsse, Seen, etc.)	ja	nein
Sonstige Symbole, wie Parkplätze oder Flughäfen	ja	nein

Ob ein wichtiger Faktor für Netzpläne im ÖPNV auch bei Enterprise Portal Maps gilt, konnte im Rahmen dieser Arbeit nicht geklärt werden: Die relative Unveränderlichkeit des Layouts über längere Zeiträume hinweg. Ein derartige Unveränderlichkeit bei EPMs ist zumindest zweifelhaft, da sich Portale in ihren Inhalten radikaler ändern können, als Stationen und Streckenführungen im ÖPNV.

Nach dieser Bewertung des Ansatzes werden im folgenden Abschnitt praktische Erfahrungen im Zusammenhang mit der Notation der Netzpläne wiedergegeben.

6.2 Erfahrungen mit der Notation der Netzpläne

Im Laufe der Entstehung dieser Dissertation hat sich immer wieder gezeigt, wie wichtig und wie richtig die Befolgung von Becks wesentlichem Prinzip ist: Je einfacher und damit übersichtlicher die Darstellung, desto besser.

Versuche mit einer alternativen Darstellung von Stationen, die von mehr als einer Linie angefahren werden, oder in deren Fahrplan der Ereignistyp Erfolg, Fehler oder Warnung mehrfach vorkommen, führten nur bei relativ einfachen Netzplänen mit wenigen Stationen zu befriedigenden Ergebnissen. Auch die Darstellung von Bedingungen nicht durch einfache Kreise an den Stationen, sondern durch eine Fläche (ein Polygon), die (das) alle Stationen umfasst, deren Dienst mit derselben Bedingung versehen ist, erreicht nicht die für größere Netzpläne unbedingt notwendige Klarheit und Eleganz. Es zeigte sich immer wieder, dass die Darstellung nach Beck sich gerade dort bewährt, wo andere Darstellungen von Zusammenhängen[2] problematisch werden: Bei „großen“ Netzen, die mehr als 10-15 Knoten enthalten, die nicht nur rein linear miteinander vernetzt sind.

Erfolglos verliefen Versuche, durch die Integration von Graph-Algorithmen eine Funktion zum automatischen Layout der Netzpläne zu erhalten. Das Problem bei allgemeinen Layout-Algorithmen für Graphen besteht darin, dass es sich bei den Netzplänen eben nicht um allgemeine Graphen, sondern um eine ganz spezielle Notation handelt. Die Ergebnisse eines automatischen Layouts verletzten insbesondere die Bedingung, dass nur horizonta-

[2]Als Beispiele seien hier die übliche Darstellungen von *Petri-Netzen* (siehe z.B. [Bau90]) oder von Prozessen in Form von EPKs, wie sie in [Sch98] zu finden sind, genannt.

le, vertikale und 45° Linien verwendet werden dürfen und berücksichtigten nicht, dass die Strecke zwischen zwei Stationen nicht unbedingt entlang einer Geraden verlaufen muss. Auf den ersten Blick scheinen Layout-Algorithmen für Leiterbahnen besser geeignet, da diese sich ebenfalls bemühen, nur horizontale, vertikale und 45° Linien zu verwenden. Das Optimierungsziel bei Algorithmen für Leiterbahnen sind aber möglichst kurze Strecken (und damit eine höhere Taktzahl bzw. niedrigere Signallaufzeiten), und nicht, wie bei den Netzplänen, eine möglichst elegante und für Menschen verständliche Darstellung.

Wichtig bei der Gestaltung von Netzplänen ist auch die Wahl der Farben. In praktischen Versuchen zeigte sich, dass für die Darstellung von Linien gesättigte Farben vorteilhaft sind, da diese auf einem weißen Hintergrund gut sichtbar sind und somit die Navigationsmöglichkeiten klar kommuniziert werden. Als „Farbe“ für Stationen hat sich schwarz bewährt, dies gilt sowohl für die Umsteigestationen, die durch einen Kreis dargestellt werden, als auch für einfache Stationen, die durch einen Tick dargestellt werden. Die einheitliche Farbgebung der Stationen hebt sie auf dem Netzplan hervor, weicht aber von Becks ursprünglichem Design ab, der die Ticks in der Farbe der anfahrenden Linie gezeichnet hat. Für Bedingungen sollten leuchtend helle, ungesättigte Farben verwendet werden, die sich, wenn möglich, an den Farben der Linien orientieren sollten, an denen die Stationen liegen, wo die Bedingungen Voraussetzung für den Besuch sind. So wird z.B. in Abbildung 5.8 (Seite 116) für die Linie Dokument die Farbe Grün verwendet, entsprechend sind die Bedingungen „Dokument lesen“, „Dokument bearbeiten“ und „Dokument anlegen“ in unterschiedlich hellen Grüntönen dargestellt. Eine geeignete Farbgebung kann erheblich zur Übersichtlichkeit und intuitiven Verständlichkeit eines Netzplans beitragen.

6.3 Ausblick

In dieser Arbeit wurde das Modell der Enterprise Portal Maps definiert und zwei prototypische Implementierungen auf der Basis des infoAsset Brokers bzw. des SAP Web Application Servers 6.10 vorgestellt. Dabei haben sich mehrere Punkte ergeben, die in zukünftigen Arbeiten behandelt werden könnten:

- Verständlichkeit des Modells auch für einfache Endanwender sollte in empirischen Tests genauer – nicht nur stichprobenartig wie im Verlaufe dieser Arbeit geschehen – überprüft werden.

- Es könnten EPMs für deutlich mehr Unternehmensportale erstellt werden, um mögliche typische Muster in Portalen erkennen zu können.

- Das vorgestellte EPM-Applet sollte eingehenden, wissenschaftlichen Usability Tests unterzogen werden, wie sie etwa in [Lin94] beschrieben werden. Dabei wäre auch die Frage zu klären, wie eine geeignete Integration des in [inf01] erwähnten graphischen Begriffsnavigators für den infoAsset Broker bei gleichzeitiger Verwendung des EPM-Applets aussehen könnte.

- Für die Erzeugung der Templates für den infoAsset Broker sollten veränderbare Vorlagen geschaffen werden, die durch Web-Designer leicht angepasst werden können.

- Die Generierung des HTML-Codes durch das Objekt `Z_EPM_APPLICATION` im SAP Web Application Server könnten statt der bisher festen Vorlage, unterschiedliche Vorlagen für jede Station entwickelt werden, die auch die erwähnte Tag-Bibliothek der SAP verwendet.

- Das Konzept der EPM-Dienste bietet sich auch für die automatische Integration von Web-Services an, die derzeit im kommerziellen Umfeld stark an Popularität gewinnen. Eine solche Integration sollte ohne manuelle Nacharbeit möglich sein, da sich die Beschreibung der Web-Services – im Gegensatz zur Business API des infoAsset Brokers – maschinell auswerten lässt.

- Für die Erschließung eines breiteren Anwenderkreises könnte das EPM-Definitionswerkzeug und die EPM-Laufzeitumgebung für weitere Portalsysteme, z.B. cocoon, angepasst werden. Während das Definitionswerkzeug vermutlich nicht oder nur im sehr geringen Umfang zu modifizieren wäre, müsste die EPM-Laufzeitumgebung komplett neu implementiert werden, wie dies in Kapitel 5 beschrieben wurde.

- Letztlich kann das Modell der Enterprise Portal Maps auch für Zwecke des *Reverse Engineering* vorhandener Portale verwendet werden, wie es in einer frühen Phase dieser Arbeit, bezogen auf den infoAsset Broker, bereits geschehen ist. Dabei wäre zumindest ein automatisches (Teil-)Layout der Netzpläne wünschenswert. Die Anpassung eines Algorithmus für Leiterbahnen scheint hierfür am vielversprechensten zu sein.

Literaturverzeichnis

[Ala01] Alatalo, T.: Lessons learned from putting a Web engineering approach to practice. In: *The 24th Information Systems Research Seminar in Scandinavia*, 11-14 August 2001, Ulvik in Hardanger, Norway,2001

[Bar01] Bartel, R.: *Endgeräteunabhängige Realisierung von interaktiven Client/Server-Anwendungen am Beispiel eines integrierten Web- und WAP-Portaldienstes.* Diplomarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, 2001

[Bau90] Baumgarten, B.:*Petri-Netze: Grundlagen und Anwendungen*, Mannheim, Wien, Zürich: BI-Wissenschaftsverlag, 1990

[BöJu+03] Böhmann, T., Junginger, M., Krcmar, H.: Modular Service Architectures: A Concept and Method for Engineering IT services. In *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003

[Bra02] Brandl, A.: *Generierung interaktiver Informationssysteme und ihrer Benutzungsoberflächen für mehrere Benutzer*, Fakultät für Informatik, Technische Universität München, Dissertation, 2002

[BrCo+02] Brambilla, M., Comai, S., Fraternali, P.: Hypertext Semantics for Web Applications. In: *SEBD Italian National Conference on DataBase Systems*, Portoferraio, June 2002,2002

[Bru96] Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. In: *User Modeling and User-Adapted Interaction.* Kluwer academic publishers,Nr. 2-3 (1996), 1996, Seite 87-129

[BuMc81] Burke, M., McLaren, I.: London's public transport diagrams - a visual comparison of some graphic conventions. In: *Information Design Journal.* Band 2, Nr. 2 (1981), 1981, Seite 103-112

[ChCu+01] Christensen, E., Cubera, F., Meredith, G., Weerawarana, S.: *Web Services Description Language (WSDL) 1.1 (Note).* Über http://www.w3.org/TR/2001/NOTE-wsdl-20010315, Stand 2001

[CrFr+00] Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. In: *WWW9 / Computer Networks*. 33 (1-6), 2000, Seite 137-157

[dtv90] DTV:*dvt-Lexikon in 20 Bänden*, München: Deutscher Taschenbuch Verlag, 1990

[Du91] Drosdowski, G.:*Duden Rechtschreibung der deutschen Sprache*, Mannheim: Dudenverlag, Band 1, 20. Auflage, 1991

[EbGu+02] Eberhardt, C., Gurzki T., Hinderer, H.: *Marktübersicht Portalsoftware*, Artikel Nr. 242, über`www.contentmanager.de`, Stand April 2003.

[Epi02] Epicentric Inc.: *Epicentric Foundation Server 4.0 - Fact sheet*. Über www.epicentric.com, Stand Okt. 2002

[FeFl+00] Fernandez, M. F. , Florescu, D., Levy, A. Y., Suciu D.: Declarative specification of web sites with Strudel. In: *VLDB Journal*. 9 (1), 2000, Seite 38-55

[FoSc98] Fowler, M., Scott, K.:*UML konzentriert - Die neue Standart-Objektmodellierungssprache anwenden*, Bonn: Addison Wesley Longman, 1998

[FrPa00] Fraternali, P., Paolini, P.: Model-driven development of web applications: the autoweb system. In: *ACM TOIS*. 18(4), 2000, Seite 323-382

[FrPa98] Fraternali,P., Paolini, P.: A Conceptual Model and Tool Environment for Developing More Scalable, Dynamic and Customizable Web Applications. In:Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (Hrsg.):*Proceedingsof the 6th International Conference on Extending Database Technology(EDBT '98)*, Valencia, Spain. Heidelberg: Springer, 1998, Seite 421-435

[GaPa+93] Garzotto, F., Paolini, P., Schwabe, D.: HDM - A Model-Based Approach to Hypertext Application Design. In: *ACM TOIS*. 11 (1), 1993, Seite 1-26

[Gar94] Garland, K.:*Mr Beck's Underground Map*, Middleesex: Capital Transport Publishing, 1994

[GoCa+01] Gómez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web applications. In: *IEEE Multimedia. Special Issue on Web Engineering*. 8 (2), 2001, Seite 26-39

[HaCh93] Hammer, M., Champy, J.: *Reengineering the Corporation. A Manifesto for Business Revolution*. New York: Harper Business, 1993

[HuHo95] Huberman, B., Hogg, T.: Communities of Practice: Performance and Evolution. In: *Computational and Mathematical Organization Theory*. 1. Jahrgang, 1995, Seite 73-92

[inf01] infoAsset AG: *The infoAsset Broker - Technical White Paper.* Über www.infoasset.de, Stand 2001

[IsSt+95] Isakowitz, T. , Stohr, E. A. , Balasubramanian, P.: RMM: A Methodology forStructured Hypermedia Design. In: *Communications of the ACM.* 38 (8), 1995, Seite 34-43

[Jab97] Jablonski, Böhm, Schulze (Hrsg.): *Workflow-Managment: Entwicklung und Anwendung von Systemen*, Heidelberg: dpunkt-Verlag, 1997

[Jac02] Jacobsen, C.: *Personalisierung von Portalsystemen*: Konzepte und Techniken, Fachbereich Informatik, Universitat Hamburg, Diplomarbeit, 2002

[KoKoe+01] Kobsa, A., Koenemann, J., Pohl, W.: Personalised hypermedia presentation techniques for improving online customer relationships. In: *The Knowledge Engineering Review.* Vol. 16:2, 2001, Seite 111-155

[Kri98] Kristensen, A.: Developing HTML based WebApplications. In: *1st International Workshop on WebEngineering*, Brisbane, Australia,1998

[KrKo02] Kraus, A., Koch, N.: Generation of Web Applications from UML Models Using an XML Publishing Framework. In: *Integrated Design and Process Technology*, IDPT-2002, Society for Design and Process Science, 2002

[LoJo80] Lakoff, G., Johnson, M.: *Metaphores We Live By*, The Univeristy of Chicago Press, 1980

[Lin94] Lindgaard, G.: *Usability Testing and System Evaluation.* A guide for designing useful computer systems, London: Chapman and Hall, 1994

[Mac95] MacEachern, A.: *How Maps Work*, New York, London: The Guilford Press, 1995

[MaFr02] Maurino, A., Fraternali, P.: Commercial Tools for the Development of Personalized Web Applications: A Survey. In:Bauknecht, K., Tjoa, A.M., Quirchmeyr, G. (Hrsg.): *EC-Web 2002.* Berlin, Heidelberg: Springer-Verlag, 2002, Seite 99-108

[MaLe02] Matthes, F., Lehel, V.: Persönliche Informations- und Wissensportale als dualer Ansatz zu Unternehmensportalen. In:Schubert, S., Reusch, B., Jesse, N. (Hrsg.): *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der Gesellschaft für Informatik e.v. (GI), 30. September - 3.Oktober 2002 in Dortmund.* LNI 19: GI, 2002, Seite 551-557

[MaLe02a] Matthes, F., Lehel, V.: Dokument- und Kontaktsynchronisation mit mobilen Datenbanken: Anforderungen und Lösungsansätze aus Sicht von Unternehmensportalen. In:Höpfner, H, Saake, G. (Hrsg.): *Tagungsband Workshop Mobile Datenbanken und Informationssysteme - Datenbanktechnologie überall und jederzeit, Arbeitskreis Mobile Datenbanken und Informationssysteme im GI-Fachausschuss*

2.5, März 2002. Magdeburg: Otto-von-Guericke-Universität Magdeburg, 2002, Seite 3-9

[MaSt+02] Maedche, A., Staab, S., Studer, R., Sure, Y., Volz, R.: SEAL- Tying Up Information Integration and Web Site Management by Ontologies. In: *IEEE Data Engineering Bulletin*. 25 (1), 2002, Seite 10-17

[MaSt02] Matthes, F., Steinfatt, K.: Vernetzung und Erschließung heterogener Wissensquellen durch den infoAsset Broker. In:Gronau, N. (Hrsg.): *Wissensmanagement - Strategien, Prozesse, Communities. 3. Oldenburger Fachtagung*. Aachen: Shaker, 2002, Seite

[Min85] Minsky, M.: *The Society of Mind*, New York: Simon and Schuster, 1985

[Mon91] Monmonier, Mark: *HOW to LIE with MAPS*, Chicago, London: The University of Chicago Press, 1991

[NeBa+99] Neil, J., Bass, B., O'Connor, C., Aldort, J., Grimditch, T.: *The New Business Portal*, Forrester Report Febrary 1999, Cambrigde: Forrester Research, 1999

[PaIn+97] Pastor, O., Insfran, E., Pelechano, V., Romero, J., Merseguer, J.: OO-METHOD:An OO Software Production Environment Combining Conventional and FormalMethods. In:Olive, A., Pastor, J.A. (Hrsg.):*Proceedings 9th International Conference on Advanced Information Systems Engineering CAiSE 1997*. Heidelberg: Springer, 1997, Seite 145-158

[PaPe+98] Pastor, O., Pelechano,V., Insfran, E., Gomez, J.: From Object Oriented Conceptual Modeling to Automated Programming in Java. In:Tok Wang Ling, Sudha Ram, Mong-Li Lee (Hrsg.):*ER '98. 17h InternationalConference on the Entity Relationship Approach*. Heidelberg: Springer, 1998, Seite 183-196

[Plum02] Plumtree Software Inc.: *The Plumtree Corporate Portal 4.5 Technical White Paper*. Über www.plumtree.com, Stand Sept. 2002

[RaMue+02] Raulf, M., Müller, R., Steffens, U., Matthes, F., Scheunert, K., Schmidt,J.: Begriffsorientierte Dokumentenverwaltung für das internetgestützte Projektmanagement - Der FHH InfoBroker für das Projekt "sap für hamburg". In:Richter, R. (Hrsg.):*Tagungsband 4. GI-Fachgruppentagung Management und Controlling von IT-Projekten* , Glashütten (Taunus). Heidelberg: dpunkt.verlag, 2001, Seite 163-186

[Re02] Rees, M. J.: Evolving the Browser Towards a Standard User Interface Architecture. In: *Conferences in Research and Practice in Information Technology*, Melbourne, Australia, 2002

[Rob52] Robinson, Arthur: *The Look of Maps*, Madison: University of Wisconsin Press, 1952

[RoSc+99] Rossi, G., Schwabe, D., Lyardet, F.: Web Application Models are More than Conceptual Models. In:Chen,P. P., Embley, D. W., Liddle, S. W. (Hrsg.):*Proceesings of the International Workshop on the World Wide Web and Conceptual Modeling*, Paris, Oct. 1999. Heidelberg: Springer, 1999, Seite 239-252

[Roy70] Royce, W.: Managing the development of large software systems: concepts and techniques. In: *Proc. IEEE WESTCON*, Los Angeles, pages 1-9, August 1970. Reprinted in Proceedings of the Ninth International Conference on Software Engineering, March 1987, pp. 328-338.

[SaGr+01] Sandvad, E., Grønbæk, K., Sloth, L., Knudsen, J.: A Metro Map Metaphor for Guided Tours on the Web: the Webvise Guided Tour System. In: *World Wide Web*. Band 8, 2001, Seite 326-333

[SAPE02] SAP AG: *mySAP Technology: Exchange-Infrastruktur: Prozessorientierte Integration und Zusammenarbeit*. Über www.sap-ag.de, Stand 2002

[SAPH02] SAP AG: *Online Hilfe*. Über help.sap.com, Stand Novmeber 2002

[SAPP02] SAP AG: *mySAP Technology: Portal Infrastruktur: Benutzerorientierte Integration und Zusammenarbeit*, White Paper, Version 1.1. Über www.sap-ag.de, Stand Okt. 2002

[SAPW02] SAP AG: *mySAP Technology: Web Application Server: Technologie und Web-Dynpro*. Über www.sap-ag.de, Stand November 2002

[Sch83] Schneiderman, B.: Direct Manipulation: A Step Beyond Programming Languages. In: *IEEE Computer*. Band 16 (8), 1983, Seite 57-69

[Sch98] Scheer, August-Wilhelm:*ARIS - Modellierungsmethoden, Metamodelle, Anwendungen*, Berlin: Springer, 3. Auflage, 1998

[Schoe92] Schöning, U.:*Theoretische Informatik kurz gefaßt*, Mannheim: BI-Wiss.-Verl., 1992

[SchuSchw99] Schumacher, M., Schwickert, A.: Web-Portale – Stand und Entwicklungstendenzen. In: Lehrstuhl für Allg. BWL und Wirtschaftsinformatik (Hsrg.): *Arbeitspapiere WI*. Nr. 4/1999, Johannes-Gutenberg-Universität, Mainz, 1999

[Su02] Suhl, U.: *Betriebliche Informationssysteme mit dem Schwerpunkt entscheidungsunterstützende Systeme*. Foliensatz zur Vorlesung SS 2002, Freie Universität Berlin, Fachbereich Wirtschaftwissenschaften, Lehrstuhl für Wirtschaftsinformatik, über www.wiwiss.fu-berlin.de/suhl, Stand April 2003

[TrDe00] De Troyer,O., Decruyenaere, T.: Conceptual Modelling of Web Sites for End-Users. In: *WWW Journal*. 3 (1), 2000, Seite 27-42

[W3C01] W3C: *SOAP version 1.2 (working draft).* Über www.w3.org/TR/2001/WD-soap12-part0-20011217, Stand 2001

[WebML02] WebML.org: *WebML User Guide - Version 3.0.* Über www.webml.org, Stand November 2002

[WebR02] WebRatio: *Modeling data entry and operations in WebML.* Über http://webml.elet.polimi.it/webml/readings/WebML2.html, Stand November 2002

[Weg02] Wegner, H.: *Analyse und objektorientierter Entwurf eines integrierten Portalsystems für das Wissensmanagement*, Arbeitsbereich STS, Technische Universität Hamburg-Harburg, Dissertation, 2002

[Zah02] Zahir, Rahila: *mySAP Portals: Concept - Architecture - Technology.* Arbeitsbereich Softwaresysteme (STS), Technische Universität Hamburg-Harburg, project thesis, 2002

[ZhKe+02] Zhao, W., Kearney, D., Gioiosa, G.: Architectures for Web Based Applications. In: *Fourth Australasian Workshop on Software and Systems Architectures*,2002

[Zie94] Ziemer, S.: *Intelligent Tutoring Systems in general and Curricular in particular - Directed Reading 1994, University of Warwick, UK.* Über www.sts.tu-harburg.de/ st.ziemer/its.pdf, Stand November 2002

[Zie97] Ziemer, S.: *SAP R/3 - An Overview of its Concepts and Languages.* Fachbereich Informatik, Universität Hamburg, Studienarbeit, 1997

[Zie98] Ziemer, S.: *Eine transportorientierte Workflow-Sprache: Definition, Anwendung und Bewertung.* Fachbereich Informatik, Universität Hamburg, Diplomarbeit, 1998

[ZiLa02] Ziegeler, C., Langham, M.: *Building XML Portals with Cocoon.* Über www.xml.com/pub/a/2002/07/24/xmlportal.html, Stand Sept. 2002

[ZiWe+01] Zirpins, C., Weinreich, H., Bartelt, A., Lamersdorf, W.: Advanced Concepts for Next Generation Portals. In: Tjoa, A.M. and Wagner R.R.: *Proceedings of the 12th International Workshop on Database and Expert Systems Applications.* Los Alamos: IEEE Computer Society, 2001, Seite 501-506

[Zol01] Zoller, P.: *HMT: Modeling Interactive and Adaptive Database-driven Hypermedia Applications*, Fakultät für Informatik, Technische Universität München, Dissertation, 2001