Universität Kaiserslautern
Fachbereich Informatik
AG Datenbanken und Informationssysteme
Prof. Dr. Theo Härder

Möglichkeiten der Anwendungsintegration und Personalisierung Web-basierter Informationssysteme durch Portale am Beispiel eines Handelssystems für Warentermingeschäfte
- Konzepte und Realisierung -

Diplomarbeit

von

Meik Arends

Betreuer:

Dipl.-Inform. Marcus Flehmig

September 2001

Hiermit erkläre ich an Eides statt, daß ich die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur angefertigt habe.

Kaiserslautern, den 21.09.2001

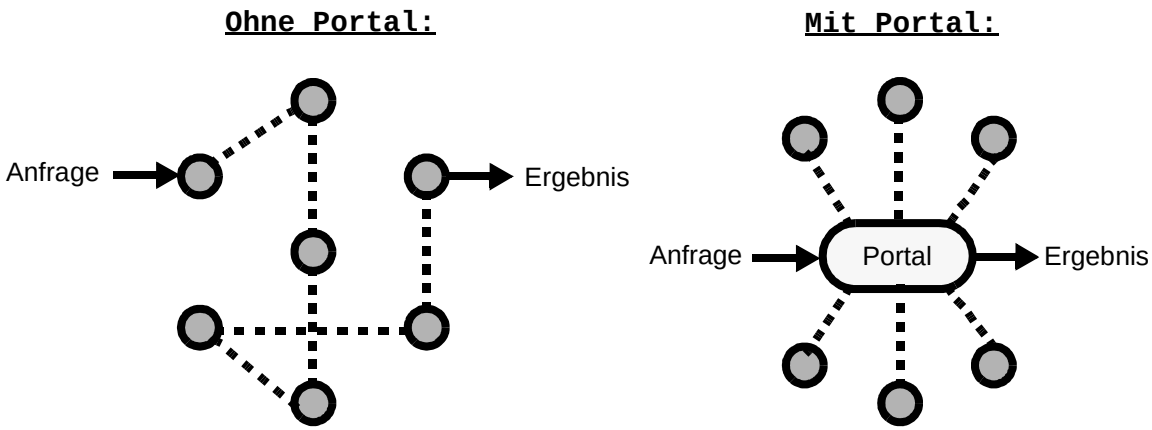
Meik Arends

Inhaltsverzeichnis

Kapitel 1 Einleitung

Die durch das Internet ermöglichte Globalisierung der Absatzmärkte eröffnet eine Vielzahl von neuen Wegen der Informationsgewinnung und Geschäftsabwicklung. Der immer schneller wachsende Markt erfordert also Möglichkeiten, die angebotenen Waren[1] in einer angemessenen und ansprechenden Form zu präsentieren. So wird das Angebot erst attraktiv, wenn der Konsument mit zusätzlichen und über den eigentlichen Umfang des Angebots hinausgehenden Informationen versorgt wird, ohne diese lange selbst suchen zu müssen. Eine Lösung für diese Problematik stellen Portale dar. Ohne Portal müssen bis zum Erhalt der gesuchten Informationen häufig mehrere Informationsquellen einbezogen werden. Das Portal ermöglicht einen zentralisierten Zugriff auf Informationen, die damit oft im ersten Anlauf gefunden werden können. Ein mögliches Szenario bei der Informationssuche mit und ohne Nutzung eines Portals zeigt Abbildung 1 [BuGo99].

Abbildung 1 Informationssuche im Internet

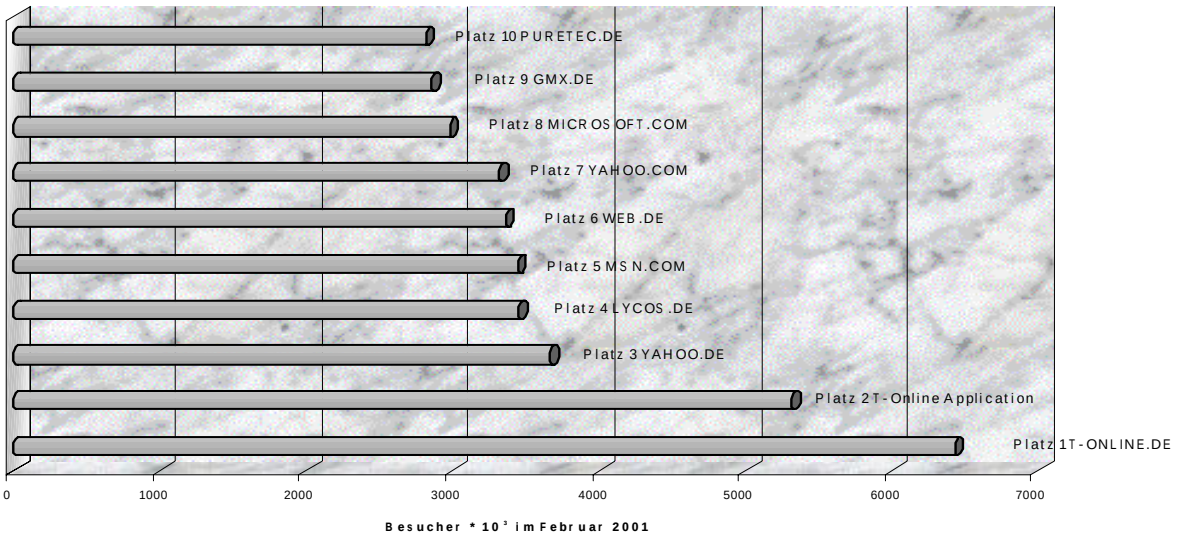


Eine weitere Verbesserung der aktuellen Situation durch Portale soll die Möglichkeit zur Personalisierung von Informationen an die Wünsche und Ansprüche des Konsumenten darstellen. Nicht zu vergessen bleiben die Interessen des Anbieters: ein gut informierter und zufriedener Kunde tätigt eher ein Geschäft. Der Anbieter kann zusätzlich auf naheliegende Waren aufmerksam machen. Portale lassen sich also, wie der Name schon erahnen lässt, als Einstiegstür in den

1. Unter Waren verstehen wir im folgenden Informationen, Wissen, Dienstleistungen oder auch Produkte.

virtuellen Raum des Internets mit all seinen Möglichkeiten und Fähigkeiten charakterisieren. Nach dem Eintritt stellt es eine Art Rezeption dar, welche dem Benutzer kompetent und individuell weiterhelfen soll. Statistiken beweisen den Erfolg des Portalkonzepts. So sind in den TOP 10 Domainaufrufen überwiegend Portale zu finden. Einen Überblick liefert Abbildung 2.

Abbildung 2 Jupiter MMXI - Top 10 Domains in Deutschland (Februar 2001)



Im Februar 2001 war demnach die in Deutschland am häufigsten angesteuerte Adresse die des Portalbetreibers T-Online. Wie in Abbildung 3 auf Seite 5 zu sehen, bietet sich dem Besucher eine Auswahl von allgemeinen aktuellen Nachrichten, über thematisierte Bereiche wie Sport, Produktinformationen der von T-Online vermarkteten Dienste bis hin zur integrierten Anwendungen. Beispielsweise steht die Organisation und Verwaltung des eigenen T-Online Internetzugangs zur Verfügung. Aufgrund dieser Entwicklungen sollen Portale in dieser Arbeit näher untersucht werden.

Abbildung 3 Screenshot Portal www.t-online.de

Kapitel 2 Motivation

Fast jeder hat es schon einmal benutzt. Das *World Wide Web* (WWW). Immer mehr Dienste stehen dem Anwender zur Verfügung. Es wächst sozusagen mit den Anforderungen welche die Benutzer an die Funktionalität des WWW stellen. Derzeit wird ein besonderes Augenmerk auf Portale und darin integrierte Applikationen gelegt. Sie sollen, wie bereits in der Einleitung erwähnt, die Nutzung des WWW komfortabler gestalten.

Zuerst soll nun in Kapitel 3 der Weg in der Entwicklung von der Homepage zum Portal betrachtet werden. Dazu wird ein Überblick über die Architektur des WWW gegeben. Diese musste sich parallel zu den Anforderungen an das WWW weiterentwickeln. Zusätzlich wird in Kapitel 3 eine Definition der Begriffe Portal, E-Commerce und Handelssystem im Web versucht und Beispiele für Portale und Handelssysteme gegeben. Mögliche Anwendungsfelder für Portale werden eingeführt und kategorisiert.

Im Kapitel 4 werden die funktionalen und nicht funktionalen Anforderungen an Portale und Handelssysteme herausgearbeitet. Diese zu identifizieren stellt eine wichtige Aufgabe bei der Planung und Konzeption von Portalen dar. Hier sollen allgemeine und spezielle Anforderungen aus Produkten unterschiedlicher Portalanbieter betrachtet werden und in einen Zusammenhang mit den in Kapitel 3 eingeführten Anwendungsfeldern gebracht werden.

Die zur Realisierung der funktionalen und nicht funktionalen Anforderungen benötigte Portalarchitektur soll in Kapitel 5 vorgestellt werden. Aus mehreren Beispielarchitekturen soll eine allgemeine Portalarchitektur erarbeitet werden. Diese soll einen Überblick über die in den meisten Portalsystemen enthaltenen Komponenten geben. Desweiteren sollen Wege und Strategien zur Realisierung von Portalsystemen aufgezeigt werden.

Kapitel 6 spezialisiert die in Kapitel 5 gewonnenen Erkenntnisse anhand einer detaillierten Betrachtung des Open-Source-Portals Jetspeed. Aus der Begutachtung der eingesetzten Software-Komponenten und der Analyse des Informationsflusses durch das Jetspeed-Portal wird dessen Architektur übersichtlich aufgearbeitet und Gemeinsamkeiten und Unterschiede zur der in Kapitel 5 vorgestellten allgemeinen Portalarchitektur hervorgehoben. Weiterhin werden zwei Design-Pattern vorgestellt und verglichen. Diese Pattern können bei der Implementierung eines Portals eingesetzt werden. Die umfassende Vorstellung des Jetspeed-Portals soll zusätzlich das Verständnis der prototypischen Implementierung eines Web-basierten Handelssystems für Warentermingeschäfte, welches im weiteren vorgestellt wird, erleichtern.

In Kapitel 7 wird zunächst das Beispielsystem eines Web-basierten Handelssystems für Warentermingeschäfte vorgestellt. Die Anforderungen an ein solches System werden aufgezeigt und analysiert sowie in einem Architekturentwurf übersichtlich dargestellt.

Kapitel 8 befasst sich mit der für Portale ebenfalls sehr wichtigen Anwendungsintegration mit deren Hilfe man Portale besonders attraktiv gestalten kann. Es werden Wege und Mechanismen zur Anwendungsintegration vorgestellt. Der Umstand, dass Jetspeed keine Funktionen zur Anwendungsintegration mitliefert, erfordert im Hinblick auf das zu realisierende Beispielsystem alternative Möglichkeiten zur Integration von Anwendungen.

Daher soll in Kapitel 9 mit SOAP gezeigt werden, wie Anwendungen mittels eines dafür entwikkelten, auf XML-basierenden Kommunikationsprotokolls integriert werden können. Nach einer Übersicht und der Vorstellung der Grundlagen des SOAP-Protokolls wird dessen Einsatz an einem einfachen Beispiel demonstriert.

Kapitel 10 und Kapitel 11 beschreiben dann die für die prototypische Realisierung des Beispielsystems notwendigen Schritte und Vorgehensweisen. Es findet eine Festlegung statt, welche der Anforderungen im Rahmen dieser Arbeit mit Hilfe des Jetspeed-Portals realisiert werden sollen, da die Zeit für eine komplette Implementierung den Rahmen dieser Arbeit sprengen würde. Kapitel 10 beschränkt sich dabei auf die Realisierung von Anforderungen durch Jetspeed-Standardkomponenten währen in Kapitel 11 gezeigt wird, wie mit Hilfe von SOAP eine Anwendungsintegration in Jetspeed stattfinden kann.

Abschliessend sollen die Ergebnisse dieser Arbeit noch einmal zusammengefasst werden und ein Ausblick auf weitere Entwicklungsmöglichkeiten und Tendenzen in diesem Bereich gegeben werden. Da die Dokumentation des eingesetzten Jetspeed-Portals sehr lückenhaft ist, finden sich im Anhang A und Anhang B weitere Informationen zu diesem Portal, unter anderem eine Installationsanleitung für das Betriebssystem Linux.

Kapitel 3 Die Client / Server Architektur des World Wide Web

3.1 Überblick

Die Kommunikation im *World Wide Web* (WWW) beruht auf einer Client / Server Architektur. Vom WWW-Client ausgehende Anfragen werden von einem WWW-Server entgegengenommen, ausgewertet und beantwortet. Der Datentransport findet hierbei größtenteils über das *Hyper-Text Transfer Protocol* (HTTP) [HTTP99] statt. Mittels dieses Protokolls werden Anforderungen vom WWW-Client an den WWW-Server übermittelt. Auf dem gleichen Wege werden die Ergebnisse der Anfrage oder eventuelle Fehlermeldungen vom Server an den Client zurückgeschickt (siehe Abbildung 4 auf Seite 10). Die Ergebnisse bestehen zum einen aus nach vorgeschriebenen Regeln zusammengesetzten Dokumenten in *Hyper-Text Markup Language* (HTML) [HTML99]. Über darin enthaltene Hyperlinks können Referenzen auf andere Seiten auf dem gleichen Server, andere Server oder Dateien jeglicher Art definiert werden. Zum anderen kann als Ergebnis einer Anfrage auch direkt ein Bild oder eine beliebige Datei geliefert werden. Um die Art der gelieferten Daten feststellen zu können, wurde ab der Version 1.1 der HTTP Protokollspezifikation eine Erweiterung vorgesehen, die *Multipurpose Internet Mail Extensions* (MIME) [HTTP99]. Der WWW-Server benachrichtigt auf diesem Wege den WWW-Client bereits zu Beginn des Downloads, welcher Dateityp geliefert wird.

Da das HTTP Protokoll keine Mechanismen zum Schutz der übertragenen Daten vor Manipulation oder Abhören bietet, musste im Zuge der höheren Sicherheitsanforderungen eine Möglichkeit zur Verschlüsselung der Daten gefunden werden. Mit Hilfe von *Secure Sockets Layer* (SSL) [SSL96] steht das HTTPS Protokoll (HTTP over SSL) zur Verfügung. Dieses ermöglicht durch eine bis zu 128-Bit Veschlüsslung einen ausreichenden Schutz für die meisten Anwendungsfälle. Eine allgemeinere Beschreibung einer Client / Server Architektur findet sich in Definition 1 auf Seite 10.

Abbildung 4 Client / Server Architektur

Definition 1 Client / Server Architektur

Unter der Client-Server-Architektur (engl.: client / server architecture) versteht man eine kooperative Informationsverarbeitung, bei der die Aufgaben zwischen Programmen auf verbundenen Rechnern aufgeteilt werden. In einem solchen Verbundsystem können Rechner aller Art zusammenarbeiten. Server (WWW-Server) bieten über das Netz Dienstleistungen an, Clients (WWW-Clients) fordern diese bei Bedarf an. Die Kommunikation zwischen einem Client-Programm und dem Server-Programm basiert auf Transaktionen, die vom Client generiert und dem Server zur Verarbeitung überstellt werden. Eine Transaktion ist eine Folge logisch zusammengehöriger Aktionen, beispielsweise zur Verarbeitung eines Geschäftsvorfalls. Client und Server können über ein lokales Netz verbunden sein oder sie können über große Entfernungen hinweg miteinander kommunizieren. Dabei kann es sich um Systeme jeglicher Größenordnung handeln; das Leistungsvermögen des Clients kann das des Servers also durchaus übersteigen. Grundidee der Client-Server-Architektur ist eine optimale Ausnutzung der Ressourcen der beteiligten Systeme [Han98].

3.2 Datenbank - Integration (Stufen- und Schichtenmodell)

In der Anfangszeit des WWW handelte es sich bei den von den WWW-Servern zurückgelieferten Daten hauptsächlich um statische HTML-Dokumente. Die Dokumente wurden einfach vom Server aus einer auf der Festplatte befindlichen Datei gelesen und zurückgeliefert. Diese Art von Informationsaustausch war für die ersten Anforderungen ausreichend. Jedoch stellte sich bald ein Bedarf nach dynamischen Inhalten ein. Die so entstandene Web-basierte Software-Architektur erforderte weiterhin Möglichkeiten zum Web-Site Management. Damit hielten die Datenbanken Einzug in das WWW. Die HTML- Dokumente konnten nun dynamisch mit Inhalten aus beliebigen Datenbanken gefüllt werden. Dies vereinfachte zusätzlich die Verwaltung der Informationen, da diese an zentraler Stelle gepflegt werden konnten. Je nach Art einer Anwendung kommen für den Einsatz von Datenbanken unterschiedliche Arten der Web-Anbindung in Frage. Um dieses näher erläutern zu können, soll Anhand von Beispiel 1 der unterschiedliche Charakter möglicher Anwendungen und den daraus folgenden Anforderungen verdeutlicht werden.

Beispiel 1 Verschiedene Anwendungen mit ihren Anforderungen

Gästebuch: Besucher einer Website können z. B. zur Informationsanforderung ihre Adresse hinterlassen oder ihre Meinung über die Seite abgeben. Die Daten sollen nach Eingabe in ein Formular in einer Datenbank abgelegt werden.

Anforderungen: Vom WWW aus nur schreibende Zugriffe auf die Daten; keine Pufferung notwendig; gleichzeitige Zugriffe möglich aber eher unwahrscheinlich; reine Textdaten; keine besonderen Schutzmaßnahmen bei der Datenübertragung; kurze Verweildauer daher möglichst einfache Technologie verwenden (HTML-Form anstelle Java-Applet).

Nachschlagewerk (Katalog): Über eine inhaltliche oder alphabetische Gliederung oder eine gezielte Suche Zugriff auf die gewünschten Informationen.

Anforderungen: Vom WWW aus nur lesende Zugriffe; Pufferung von häufig angeforderten Daten; viele gleichzeitige Zugriffe möglich; Zugriff auf eine Vielzahl verschiedener Daten, z.B. Multimedia; je nach Anwendung Schutz der Daten erforderlich; mittlere bis lange Verweildauer.

Online-Banking/Broking: Dem Bankkunden soll die Möglichkeit gegeben werden, seine Geschäfte über das Internet abzuwickeln.

Anforderungen: Vom WWW aus Lese/Schreib-Zugriff; je nach Funktionalität Pufferung von Daten; gleichzeitige Zugriffe in hohem Maße, da viele Kunden auf einen Zentralrechner zugreifen; hauptsächlich Textdaten; Hauptaugenmerk auf die Authentifizierung und gesicherte Übertragung der sensiblen Daten; stark variierende Verweildauer.

Beispiel 1 zeigt unterschiedliche Anwendungsfälle einer Client / Server Architektur, woraus sich verschiedene Anordnungsvarianten ableiten lassen. Wir unterscheiden im weiteren Verlauf das Stufen- und das Schichtenmodell.

Das Stufenmodell

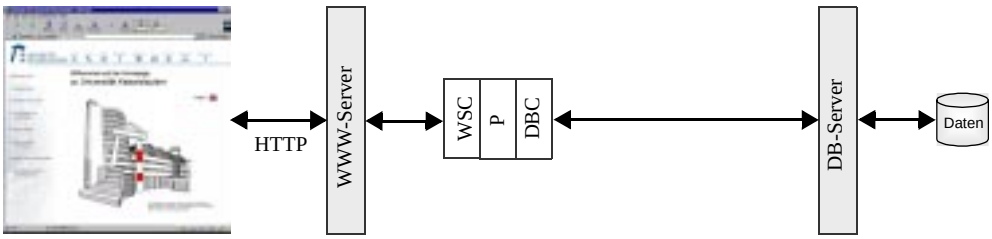
Die Datenbankanbindung beim Stufenmodell erfolgt mittels einzelner Komponenten, wobei jede Komponente für sich eine bestimmte Aufgabe erfüllt. Eine Übersicht über die Komponenten liefert Tabelle 1 auf Seite 12.

Abbildung 5 auf Seite 12 zeigt eine einstufige Anbindung. Sie kommt ihrer Einfachheit halber dann zum Einsatz, wenn keine großen Anforderungen an die Geschwindigkeit des Systems gestellt werden. Bei einer Anfrage treten erhöhte Antwortzeiten auf, da die Programme zur Datenbankanbindung erst gestartet und initialisiert werden müssen. So werden bis zum eigentlichen Verarbeitungsbeginn relativ viele Ressourcen verbraucht. Vorteilhaft für Phasen der Inaktivität ist jedoch, das keine oder nur sehr wenig Ressourcen benötigt werden. Diese Art von Anbindung kommt z.B. für die Realisierung eines elektronischen Gästebuchs wie in Beispiel 1 oder anderer Einschritt-Vorgänge in Betracht, wo die Anzahl der gleichzeitigen Zugriffe nicht sehr hoch sind.

Tabelle 1

Komponente	Beschreibung
Prozessor (engl.: processor, P)	Extraktion der relevanten Informationen aus Makro oder HTML Dateien sowie nachfolgende Interaktion mit der DBC
DB-Kommunikationskomponente (engl.: DataBase Communication, DBC)	Baut Verbindung zum DB-Server auf und veranlaßt die Abarbeitung der spezifizierten DB-Befehle
Web-Server-Kommunikationskomponente (engl.: Web Server Communication, WSC)	Leitet die vom Benutzer spezifizierten und vom Webserver empfangenen Parameter die Prozessorkomponente weiter und sorgt für den Rücktransport der dadurch erhaltenen Ergebnisse
Interkomponenten-Kommunikationsmodul (engl.: Inter Component Communication, ICC) (nur bei mehrstufigen Verfahren)	Zum Datenaustausch zwischen den unterschiedlichen Prozessen bei mehrstufigen Verfahren

Abbildung 5 Einstufige Anbindung

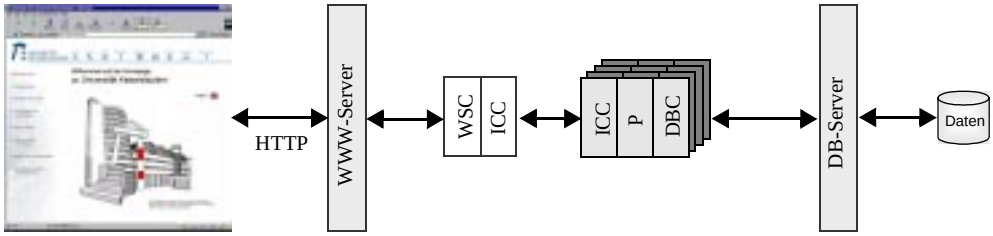


Bei der einstufigen Anbindung werden die vom Benutzer spezifizierten und vom WWW-Server empfangenen Parameter von der *Web-Server-Kommunikationskomponente* (engl.: Web Server Communication, WSC) an einen *Prozessor* (engl.: processor, P) weitergeleitet, welcher relevante Informationen extrahiert und an die Datenbank-Kommunikationskomponente weitergibt. Diese baut dann eine Verbindung zum Datenbankserver auf und veranlasst die Abarbeitung der spezifizierten DB-Operationen. Die Ergebnisse der Anfrage werden auf umgekehrtem Weg wieder an den Benutzer geleitet.

Abbildung 6 zeigt eine zweistufige Anbindung. Sie kommt bei steigenden Anforderungen an die Leistung des Systems zum Einsatz, z. B. bei der in Beispiel 1 vorgestellten Kataloganwendung.

Hierbei sind viele parallele Zugriffe zu erwarten und eine Pufferung der Daten wird wichtig, um kurze Antwortzeiten gewährleisten zu können. Die Pufferung kann z.B. in der DBC stattfinden.

Abbildung 6 Zweistufige Anbindung (mit Prozeß-Pool)



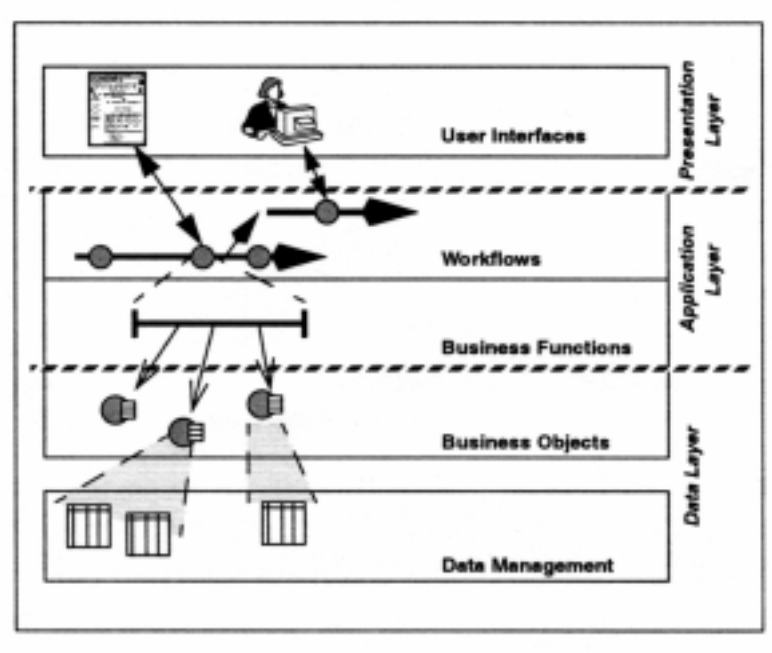
Der zweistufige Ansatz unterscheidet sich vom einstufigen Ansatz durch die Benutzung eines *Interkomponenten-Kommunikationsmoduls* (engl.: Inter-Component Communication, ICC). Dieses Modul übernimmt die Verteilung der durch das WSC empfangenen Parameter an eine Vielzahl von bereits laufenden Verabeitungsprozessen. Diese bestehen jeweils aus einem ICC, einem P und einem DBC. Da die DBC eine ständige Verbindung zum DB-Server aufrecht erhalten, sind die Antwortzeiten des Systems durch den Wegfall des Verbindungsaufbaus und der Authentifizierung nocheinmal deutlich kleiner.

Es gibt noch weitere Architekturvarianten im Stufenmodell, welche hier aber nicht weiter relevant sind und in [Loe98] umfassend behandelt werden.

Das n - Schichtenmodell

Da die Architekturen im Zuge der wachsenden Anforderungen an die Geschwindigkeit und Logik immer komplexer werden, können diese durch das Stufenmodell nicht mehr hinreichend beschrieben werden. Durch neue Methoden wie Kommunikationsmiddleware (z. B. CORBA [Omg01], [Cis01]) wird ein logisches Modell erforderlich. Hier soll das Schichtenmodell anhand eines *Busines Information Systems* (BIS) vorgestellt werden [Stei98]. Die Grundidee hierbei ist der Einsatz objektorientierter Entwurfsmethoden. Die auf diese Art entworfenen Systeme bestehen aus vorgefertigen Objekten, welche die benötigte Funktionalität zur Verfügung stellen. Dazu kommt eine passende Kommunikations-Infrastruktur wie z. B. CORBA. Die gebräuchlichste Form ist das 3-Schichten Modell. Es lässt sich grob in die Bereiche *Data-Layer*, *Application-Layer* und *Presentation-Layer* unterteilen.

Abbildung 7 Schichtenmodell



Die unterste Schicht, der Data-Layer besteht aus *Business Objects* (BO) und dem *Data Management*. Die Hauptaufgabe des Data Managements ist das Speichern von Daten und der Schutz vor eventuell auftretenden Fehlern und Inkonsistenzen. Dazu werden hier auch die transaktionellen Gesichtspunkte berücksichtigt. Man erhält damit ein System mit ACID-Eigenschaften [Hr83]. So wird eine solide Basis für die darüberliegenden Schichten realisiert. Natürlich lässt sich die Implementierung der ACID-Eigenschaften bei Bedarf auch in höhere Schichten verlagern. In der Schicht der Business Objects findet eine Abstraktion der Funtionalitäten der untersten Schicht für die höheren Schichten statt. Sie sorgen so für einen transparenten Zugriff auf die Daten. Weiterhin können sie zur Integration bestehender heterogener Systeme genutzt werden.

Die mittlere Schicht, der Application-Layer, besteht aus den *Business Functions* (BF) und den Workflows. In ihr wird die Logik des Systems implementiert. Die Business Functions sind, im Gegensatz zu den Business Objects, nicht mehr applikationsunabhängig. Sie integrieren die Logik in das System und kombinieren semantisch zusammengehörende Aktionen der Business Objects zu einer applikationsabhängigen Aktion. Um die Systemkomponenten möglichst flexibel nutzen zu können, ist eine Trennung des Kontrollflusses der Business Functions von deren Funktionalität sinnvoll. Die so erhaltenen *Workflow Objects* (WfO) bestimmen die Ausführung von Transaktionen, welche mehrere Business Functions ansprechen können sowie den Kontrollfluss zwischen den einzelnen Komponenten.

In der obersten Schicht, dem Presentation-Layer, soll dem User die Benutzung verschiedener Clients ermöglicht werden. Egal ob Web-Browser, mobiler Client oder bei der Arbeit am eigenen PC soll dem Benutzer die gleiche Funktionalität bereit gestellt werden. Dies ist nur bei strikter Trennung des Presentation-Layer vom Application-Layer möglich. Mehr Informationen zum 2-, 3- oder n-Schicht Modell findet man in [Cis01].

Von den statischen Dokumenten über die dynamisch erzeugten Dokumente kam nun noch der Bedarf neue oder bereits bestehende Anwendungen im Web integrieren zu können. So führte der Weg zu Portalen, welche besonders im Bereich Eelectronic-Commerce Verwendung finden.

3.3 Handelssysteme, Electronic-Commerce und Portale im Web

Im WWW findet man viele Bezeichnungen für Portale. So z.B. *Unternehmens-Portal* oder *Unternehmens-Informations-Portal* (engl.: *Enterprise Information Portal*, EIP), *Intranet-Portal* oder *Informations-Broker.* An dieser Stelle soll eine vereinheitlichte Definition und Klassifizierung von Handelssystemen und Portalen im Web versucht werden.

3.3.1 Electronic-Commerce und Handelssysteme

Die Definition von Handelssystemen im Web stellt kein geringes Problem dar. Da Handel vielfältig ausgelegt werden kann, soll die Definition hier auf Electronic-Commerce bezogen sein. Für diesen Begriff existieren wiederum eine Vielzahl von Definitionen jedoch keine allgemeine und allgemein gültige. Im Folgenden soll aus mehreren bestehenden Definitionsversuchen das Wichtigeste zusammengetragen werden.

- *„E-Commerce bezeichnet die Ausnutzung der technischen Mittel elektronischer Datennetze, um Wirtschafts- und Absatzprozesse einer Unternehmung zu fördern und neue Absatzwege zu erschließen. Die Einzelziele erstrecken sich dabei von der Unternehmenskommunikation über die Wertschöpfungsprozesse bis zum Verkaufsvorgang über alle Marktphasen [Foch97].“*

Aus dieser Definition soll der technische Aspekt in unsere Definition von E-Commerce einfliessen.

- *„Electronic Commerce ist ein Konzept zur Nutzung von bestimmten Informations- und Kommunikationstechnologien zur elektronischen Integration und Verzahnung unterschiedlicher Wertschöpfungsketten oder unternehmensübergreifender Geschäftsprozesse und zum Management von Geschäftsbeziehungen [Weba01a] (akademische Definition).“*

- *„E-Commerce bedeutet, etwas über das Internet zu verkaufen, Informationen über das Internet auszutauschen und dem Kunden über das Internet eine umfassende Betreuung zu bieten [Weba01b] (praxisorientierte Definition).“*

Aus den letzten beiden Definition lassen sich die verschiedenen Arten von Geschäftsvorfällen ableiten, welche ebenfalls in unserer Definition berücksichtigt werden sollen. Abschliessend fasst Definition 2 nun die obigen Aspekte zusammen.

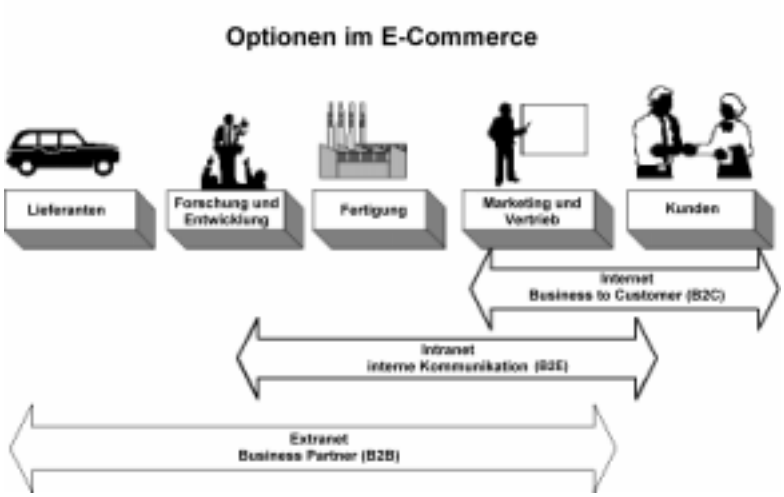
Definition 2 Electronic - Commerce (E-Commerce) / Handelssystem im Web

Unter E-Commerce versteht man den Prozeß der Kommunikation über elektronische Datennetze (inclusive Internet) zum Zwecke des Austausches von Informationen, Abschluß von Geschäften oder die über den Geschäftsvorfall hinausgehende Betreuung/ Support des Kunden. Der Vorgang kann zwischen Anbieter und Kunde (Business to Customer / B2C), unternehmensintern oder zwischen Geschäftspartnern (Business to Business / B2B) stattfinden.

Ein Handelssystem im Web ist ein System, welches E-Commerce im Internet ermöglicht.

Zur Veranschaulichung von Definition 2 dient Abbildung 8. Dort werden noch einmal die drei Anwendungsfelder veranschaulicht, in denen E-Commerce zum Einsatz kommen kann: *Business to Customer* (B2C), *Business to Business* (B2B) und zusätzlich *Business to Employe* (B2E) bei der unternehmensinternen Kommunikation zwischen den Mitarbeitern.

Abbildung 8 E-Commerce / Web-basiertes Handelssystem



Das **Internet** ermöglicht die Kommunikation mit dem Kunden (B2C) auch über grosse Enfernungen hinweg. Unter dem Begriff Internet versteht man ein globales Netzwerk, welches viele Millionen Computer miteinander verbindet. Der Großteil der im Internet angebotenen Dienste wie WWW, FTP, Mail und News ist frei zugänglich und kostenlos nutzbar. Damit eröffnen sich im E-Commerce Möglichkeiten, Marketing und Serviceleistungen als Wege zur Kundengewinnung und -bindung zu nutzen. Mittels E-Commerce können Firmen ihre Absatzprozesse in einem elektronischen Markt zusammenzuführen. Neben der Unterstützung anderer Unternehmensbereiche bietet E-Commerce die Schaffung eines neuen Absatzweges, der eine vollkommene Integration von Marketing, Absatz und Service in ein elektronisches Medium erlaubt. Die Globalität des Mediums Internet ermöglicht dabei den Eintritt in bislang unerreichbare Märkte [Foch97].

Zur unternehmensinternen Kommunikation (B2E) dient das **Intranet**. Unter dem Begriff Intranet versteht man ein privates, nicht öffentlich zugängliches Netzwerk meistens innerhalb von Firmen und Organisationen. Der Zugang wird nur Mitgliedern bzw. Angestellten gewährt. Man erreicht so z. B. die Verkürzung der Entwicklungszeiten, Prozeßautomatisierung oder die Schaffung eines gemeinsamen Wissensspeichers für die Mitarbeiter. In die durch das Intranet unterstützten Prozesse fließen auch Informationen aus den Entwicklungsabteilungen, Vertriebsabteilungen und aus der Kundenkomunikation ein [Foch97].

Im Bereich B2B kommt das **Extrane**t zum Einsatz. Extranet ist der Begriff für ein für bestimmte Bereiche nach aussen geöffnetes Intranet. Es dient weiterhin der Erweiterung der B2E Beziehungen, um den Mitarbeitern einen ortsunabhängigen Zugang zu Unternehmensdaten und Systemen zu gewähren. Man ermöglicht so, nach entsprechender Authentifizierung durch ein Login und Passwort, den externen Zugriff auf dafür bestimmte interne Daten. Da ein Unternehmen neben internen Prozessen auch eine Vielzahl von Beziehungen zu anderen Unternehmen und Partnern unterhält, ist es nützlich, auch diese in die elektronische Kommunikation einzubinden. Das Extranet stärkt die Beziehungen zwischen kooperierenden Unternehmen. Für die Verbesserung der Koordination mit Lieferanten und Abnehmern, die große Unternehmen bereits heute auf der Basis von *Electronic Data Interchnage* (EDI) betreiben, steht damit eine kostengünstige Alternative oder aber eine neue Plattform für EDI-Übertragungen zur Verfügung. Die Unternehmen können auf diese Weise z. B. ihre Lagerverwaltung weiter optimieren und mithin ihre Marktreaktionszeiten verbessern. Da Extranets teilweise die Nutzung von Internet-Infrastruktur einbeziehen, besteht auch hier bereits ein Bedarf an Sicherheitstechnologie. Um Unternehmensdaten vor fremden Zugriff zu schützen, werden bei der Übertragung Verschlüsselungs-Protokolle wie das im 3.1 bereits erwähnte HTTPS eingesetzt [Foch97].

Nun fehlte für diese Web-basierten Handelssysteme eine passende Plattform, welche das Konzept optimal unterstützt. Dieser Bedarf hat massgeblich zur Entwicklung der heute weit verbreiteten Portale beigetragen.

3.3.2 Portal

Der Portalbegriff wird häufig mit den grossen Internet Suchmaschinen assoziiert, welche ihr Angebot um E-Mail, News, Börsenkurse und ähnliche Funktionalitäten erweitert haben. Einige Unternehmen haben nach diesem Ansatz ihre eigenen firmeninterne Portale, die oben genannten *Enterprise Information Portal* (EIP) oder *Intranet-Portale* entwickelt. Anhand verschiedener Definitionen soll nun die Bedeutung des Portal-Begriffs erläutert werden. Abschliessend soll eine eigene Definition zusammengestellt werden.

Die erste Definition beschreibt fast philosophisch die ursprüngliche Bedeutung des Begriffs Portal und zeigt damit den eigentlichen Zusammenhang zwischen dem Begriff und der beabsichtigten Funktionalität der Web-Portale auf einer nicht technischen Ebene.

- *„A door or gate; hence, a way of entrance or exit, especially one that is grand and imposing and passes you on to other grand destinations. [Kno01].“*

Die zweite ist eine recht allgemeine Definition, welche nicht näher auf die angesprochenen Ausstattungsmerkmale eingeht. Ein wichtiger Aspekt ist hierbei die möglichst grosse Zielgruppe.

- *„Ein Portal besteht aus einer Startseite, welche über umfangreiche Navigationsmöglichkeiten verfügt. Weiterhin verfügt ein Portal über besondere Ausstatungsmerkmale und Fähigkeiten, welche vom Portalbetreiber direkt, durch Partner oder durch Dritte erbracht werden. Dabei hat ein Portal eine möglichst große Zielgruppe [Abo01].“*

Die zu erbringenden Leistungs- und Ausstattungsmerkmale werden in den folgenden drei Definitionen näher ausgeführt wobei in den beiden letzten Definitionen die Aspekte der Personalisierung hervorgehoben werden.

- *„Web site or service that offers a broad array of resources and services, such as e-mail, forums, search engines, and on-line shopping malls. The first Web portals were online services, such as AOL, that provided access to the Web, but by now most of the traditional search engines have transformed themselves into Web portals to attract and keep a larger audience [Webo01].“*

- *„Das ideale Portal eröffnet einen gemeinsamen, personalisierten Zugang zu Daten, Expertiesen und Anwendungen; Portale bieten Funktionen wie Navigation, Daten-Integration, Personalisierung, Notifikation, Wissensmanagement, Workflow, Anwendungsintegration und Infrastrukturdienste [Wol01].“*

- *„Portals are designed to be gateways to the Internet. They may have a link to a search engine, a subject directory, and provide other services such as news, weather, entertainment information, stock market information, shopping, and other services. Many portal sites provide an option to customize the site according to the user's personal interest. Portals are sponsored by the major search engine and browser providers, and may include alliances with other major players on the Internet [SLH01].“*

Mit Portalen soll zusätzlich eine Ortsunabhängigkeit geschaffen werden. Egal wo man sich gerade befindet soll der Zugriff auf die gleiche personalisierte Arbeitsumgebung möglich sein. Als Benutzeroberfläche scheint somit der Web-Browser geradezu ideal. Er ist heutzutage auf fast jedem Computer verfügbar und die meisten Computerbenutzer sind den Umgang mit ihm bereits gewöhnt. Im Hinblick auf die Mobilität der Nutzer müssen die selben Inhalte aber auch an verschiedene Endgeräte ausgeliefert werden können wie z. B. WAP-Handys. Daher sollen Portale auch die Funktionalität des **Multi-Channel Delivery** unterstützen. Nun sollen die in den obigen Definitionen angedeuteten Funtionalitäten der Portale etwas näher betrachtet werden [Wol01].

Bei der **Navigation** soll das Portal den Nutzer bei der Suche nach Inhalten unterstützen. Dazu gehören kombinierbare Suchkriterien welche nicht nur Dateinamen sondern auch Inhalte finden können. Im Suchraum sollten dabei sowohl Web-Dokumente als auch, z.B. bei Intranet-Portalen, für das Unternehmen relevante Informationen einbezogen werden können.

Das Portal soll weiterhin die Fähigkeit besitzen, auf Informationen aus den unterschiedlichsten Datenquellen zugreifen und dem Benutzer präsentieren zu können (**Datenintegration**).

Um die so zur Verfügung gestellte Vielzahl von Daten optimal präsentieren zu können, sollten sich diese mittels **Personalisierung** auf den für den Nutzer interessanten Teil reduzieren lassen. Auch die Art der Präsentation und das Aussehen der Umgebung sollte durch den Benutzer anpassbar sein. Weiterhin soll die Möglichkeit seitens des Portalbetreibers bestehen, die Inhalte auf das Verhalten des Nutzers bezüglich seiner bevorzugt besuchten Seiten abstimmen zu können.

Die **Notifikation** wird vom Portal durchgeführt, wenn bestimmte Ereignisse, wie z. B. das Erreichen einer bestimmten Marke eines Aktienkurses oder die nicht termingerechte Beendigung eines Projektes, eintreten. Der Nutzer oder der entsprechende Ansprechpartner erhält dann via E-Mail oder SMS[1] eine kurze Benachrichtigung.

Grundlage für die Notifikation bildet eine weitere Funktionalität, das **Wissensmanagement**. Man versteht darunter eine formales, strukturiertes Konzept zur Verbesserung der Erzeugung, Verteilung und Nutzung von Wissen in einer Organisation [Sah00].

Die **Anwendungsintegration** ermöglicht den Einsatz von Portalen zu Zwecken, welche vorher nur mit bestimmen, eigenständigen Applikationen und Clients möglich waren. Weiterhin können so E-Commerce Anwendungen im Web realisiert werden.

In Unternehmens-Informations-Portalen kommen noch der Bereich **Workflow** hinzu, um die Abläufe innerhalb eines Unternehmens zu verbessern.

Aspekte der funktionalen und nicht funktionalen Anforderungen an Portale werden in Kapitel 4 noch einmal ausführlich behandelt.

Da sich der Portalbegriff im Laufe der Zeit enwickelt und verändert hat und es sicherlich in Zukunft auch noch weiter tun wird, ist eine statische Definition nicht möglich. Eine prägnante Definition liefert Definition 3. Sie ist als aktuelle Beschreibung des Portal-Begriffs zu verstehen.

Definition 3 Portal

Ein Portal ist ein zentraler Sammelpunkt von verteilten Informationen und Anwendungen (Daten- und Anwendungsintegration). Es bietet sowohl eine Umgebung für die Zusammenarbeit von Gruppen innerhalb beliebiger Organisationen, als auch die Möglichkeit beliebig große Zielgruppen anzusprechen. Die Inhalte sind dabei auf die Anforderungen der Benutzer personalisierbar.

Zusammenfassend lässt sich also sagen, das ein Portal ein Web-orientiertes Anwendungssystem ist, in dem Inhalte und Dienstleistungen übersichtlich für eine bestimmte Zielgruppe angeboten werden. Es soll als Eingang in die Informationsvielfalt des Internet dienen. Durch die Nutzung in einem Web-Browser wird ein ortsunabhängiger Zugang geschaffen, welcher durch eine Multi-Channel-Delivery Funtionalität auf verschiedene Endgeräte erweiterbar ist. Ein Portal ist dynamisch aufgebaut und somit mehr als eine reine Linksammlung. Zu den Inhalten und in das Portal integrierten Dienstleistungen gehören neben Nachrichten, E-Mail, Foren, Suchmaschinen auch

1. Short Message Service; Dabei wird eine kurze Textnachricht an ein Mobiltelefon übermittelt.

Navigation, Daten-Integration, Personalisierung, Notifikation, Wissensmanagement, Workflow, Anwendungsintegration, Infrastrukturdienste und insbesondere Web-basierte Handelssysteme. Als besonders wichtig gilt die Möglichkeit zur Personalisierung um die angebotenen Inhalte und Dienstleistungen an die Bedürfnisse und Anforderungen des Benutzers anpassen zu können.

Im folgenden Abschnitt sollen nun ausgewählte Portale nach ihren Eigenschaften grob klassifiziert werden.

3.4 Beispiele für Portale und Handelssysteme

Das Hauptziel der meisten Portale ist die Benutzerfreundlichkeit. Der Besucher hat einen zentralen Einstiegspunkt. Wie schon erwähnt werden reichhaltige Navigationsmöglickeiten in Form von Hyperlinks angeboten. Die Portale lassen sich grob in drei Kategorien einordnen. *Internet-Portal*, *Intranet-Portal* und *Vortal* (Vertical Portal).

3.4.1 Internet-Portal

Ein Beispiel für ein Internet-Portal ist die Seite von NBC [1](Abbildung 9 auf Seite 21). Wie in der Abbildung zu sehen, sind die angebotenen Informationen breit gefächert. Das Angebot reicht von Nachrichten, Wirtschaftsinformationen und Gesundheitstipps, über E-Mail Newsletter und multimediale Inhalte, bis hin zu personalisierbaren Informationen. Man hat bereits auf der Startseite die Möglichkeit über die Angabe der eigenen Postleitzahl einige der angezeigten Informationen regional zu beschränken. Weitere Personalisierungsmöglichkeiten ergeben sich nach einer Anmeldung und der Zuteilung einer Benutzerkennung. Die im Internet-Portal integrierten Datenquellen sind von allgemeiner Natur und in der Regel auch ohne die Benutzung des Portal erreichbar.

3.4.2 Intranet-Portal

Das Portal in Abbildung 10 auf Seite 22 zeigt auf den ersten Blick die gleichen Eigenschaften wie ein Internet-Portal. Es handelt sich hierbei um ein bei Intranets.com[2] erstelltes kostenloses Portal. Gegen Bezahlung erhält der Kunde weitere Leistungsmerkmale wie Support, Verzicht auf die Einblendung von Werbebannern und mehr Speicherplatz für Informationen.

Beim Intranet-Portal sind die Inhalte organisationsspezifischer. So sollen die Angestellten in ihrer Arbeit unterstützt werden und Informationen, welche sie sonst über mehrere Quellen haben anfordern müssen, schneller finden können. Des Weiteren spielt hier die Sicherheit eine grössere Rolle. Über Gruppenstrukturen erhält der jeweilige Mitarbeiter nur Zugang zu den Bereichen, die für seine Arbeit erforderlich sind.

1. http://www.msnbc.com
2. http://www.intanets.com

Abbildung 9 Screenshot www.msnbc.com

Auf diese Art wird ebenso die Zusammenarbeit in Arbeitsgruppen gefördert. Wie man in Abbildung 10 auf Seite 22 sieht, existieren Anwendungen zur Unterstützung der Arbeit in einer Gruppe und zwischen Teams. Ein Beispiel für diese Groupware-Aspekte ist die Kalenderfunktion. Im Dokumenten- und Diskussionsbereich wird die dokumentenorientierte Darstellung der Informationen ermöglicht, sowie weiterhin der Austausch von unstrukturierten oder wenig strukturierten Informationen. Durch die Möglichkeiten zur Personalisierung kann sich jeder Benutzer das Arbeitsumfeld auf seine Vorlieben und Bedürfnisse anpassen. Ein weiterer Unterschied zu Internet-Portalen liegt beim Content-Management welches hier wesentlich flexibler ist. Auch wenn das Portal an sich von einem Drittanbieter kommt, ist der Kunde für das Bereitstellen der Inhalte verantwortlich. Die Datenquellen sind im Gegensatz zum Internet-Portal nicht zwangsläufig öffentlich zugänglich. Auffällig ist auch, dass der Inhalt funktioneller präsentiert wird. Man verzichtet auf aufwändige Effekte bei der Darstellung.

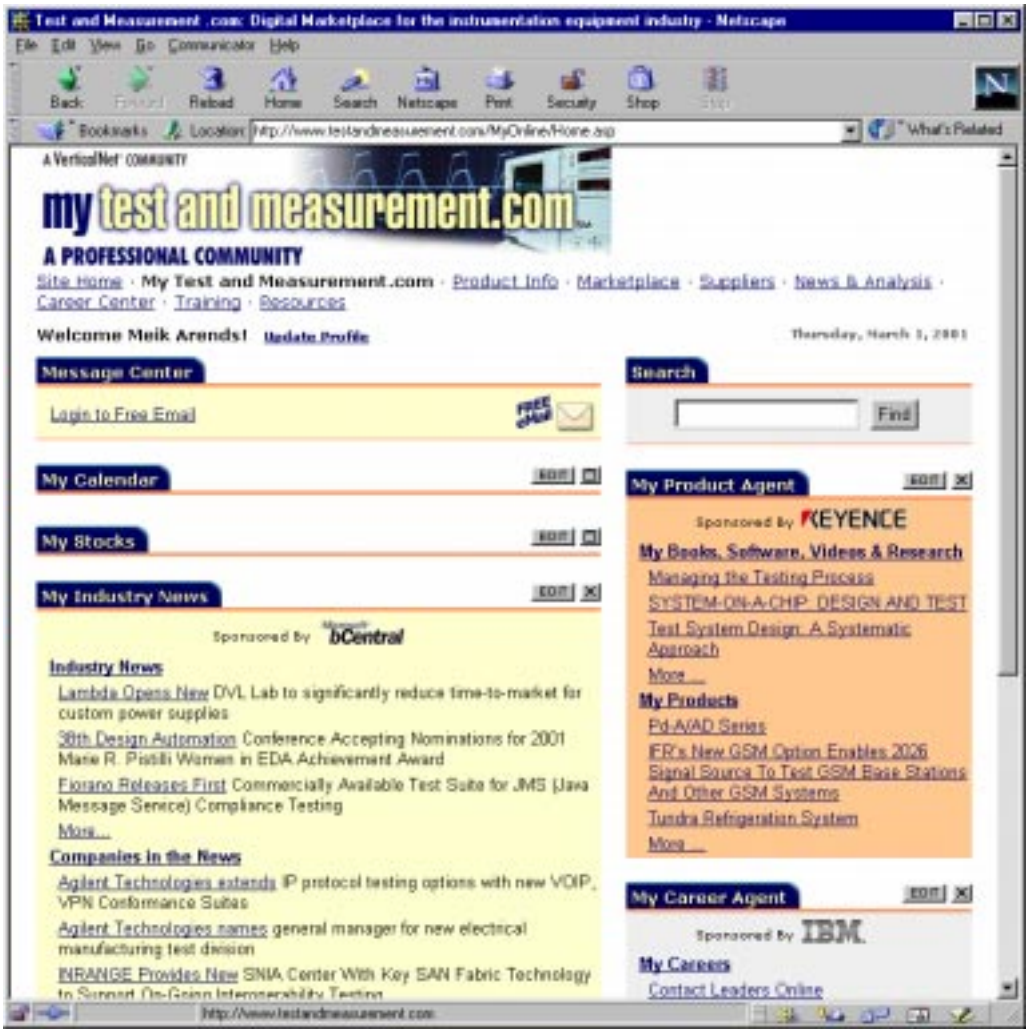
Abbildung 10 Screenshot she-ffm.intranets.com

3.4.3 Vortals

Ein Vortal (vertical portal) stellt eine Mischung aus Inter- und Intranet-Portal dar. Vortale legen, wie Intranet-Portale, ihren Fokus auf spezielle Themen und sollen die Zusammenarbeit zwischen den Nutzern ermöglichen. Andererseits bleibt das Angebot für die breite Öffentlichkeit zugänglich und kann Schnittstellen zu anderen Angeboten wie z. B. Suchmaschinen enthalten. Ein Beispiel zeigt Abbildung 11 auf Seite 23. Hier liegt der Fokus auf Informationen zum Thema „test and measurement“.

Abschliessend muss bemerkt werden, dass neben den so kategorisierten Portal-Arten auch Mischformen auftreten können. Weiterhin sind Klassifizierungen nach weiteren Gesichtspunkten möglich, welche aber hier nicht näher betrachtet werden sollen.

Abbildung 11 Screenhot www.testandmeasurement.com



3.4.4 Vision

Der Idealfall eines Portals, wäre ein Potal, in welches auf generische Art und Weise Anwendungen und Informationsquellen integriert werden könnten. Dabei soll das Portal für den Benutzer weiterhin personalisierbar bleiben. Auch der Portalbetreiber soll genügend Einfluss haben, um gezielt seine Waren anbieten zu können.

Als Negativbeispiel möge an dieser Stelle das Broking-Applet der Consors Discount Broker AG dienen. Die Brokeranwendung besteht aus einem Java-Applet, welches sich als einziges Element in einem Browserfenster befindet (Abbildung 12 auf Seite 24).

Abbildung 12 Screenshot Consors Broking-Applet

Sucht man, die gerade beim Aktien und Optionsscheinhandel, wichtigen Zusatzinformationen, so muß man, zumindest für Werte, die sich nicht im eigenen Depot befinden, auf andere Informationsquellen zurückgreifen. Ideal wäre hier also ein Portal, in welches das Consors-Applet integriert werden kann. Eine Realisierung würde folglich als Vortal betrachtet, da die darin angebotenen Informationen hauptsächlich aus Börseninformationen und Wirtschaftsnachrichten bestehen sollten.

Diese Vision soll in den späteren Kapiteln anhand eines Handeslsystems für Warentermingeschäfte weiterverfolgt werden. Im folgenden Kapitel sollen dazu allgemein die funktionalen und nicht funktionalen Anforderungen an Portale und Handelssysteme betrachtet werden. Diese sollen identifiziert und klassifizieren werden, da eine genaue Analyse und Spezifikation von Anforderungen an ein System für eine spätere Umsetzung sehr wichtig ist.

Kapitel 4 Funktionale und nicht funktionale Anforderungen an Portale und Handelssysteme

Um einen Überblick über funktionale und nicht funktionale Anforderungen an Portale und Handelssysteme zu bekommen, sollen ausgewählte Portalanbieter und deren Produkte dahingehend untersucht werden. Dabei werden die in Abschnitt 3.3.1 auf Seite 15 vorgestellten Anwendungsfelder B2C, B2B und B2E unterschieden. B2X steht dabei zukünftig für die Zusammenfassung der drei Bereiche. Es sei darauf hingewiesen, dass bei der Betrachtung der von den Herstellern formulierten Produktbeschreibungen der Marketing-Aspekt nicht zu vernachlässigen ist. Diese sollten also immer kritisch hinterfragt werden.

Das **Brio.Portal 7.0** der Firma Brio Technology [Brio01] versteht sich als umfassende Lösung zur Verwaltung von strukturierten und unstrukturierten Informationen, zum Wissensmanagement und Business Intelligence Technology im Bereich B2X. Der **Epicentric Foundation Server 3.5** [Epi01] verspricht einen einfachen und schnellen Weg zu einem „full-service e-business center“ und legt dabei seinen Focus auf die Bereiche B2B und B2C. Der **Hyperwave-Information-Server (HIS) 5.5** [Hyp01] stellt eine Softwarelösung für Intranet-, Extranet- und Internet-Anwendungen dar, welche das effiziente Erfassen, Organisieren und Verteilen von Informationen sowohl unternehmensintern als auch im Dialog mit Geschäftspartnern und Kunden ermöglichen soll. Den Server selbst sieht Hyperwave als Durchbruch im Bereich Intranet-Informationssysteme. Der Schwerpunkt des HIS liegt also bei B2E. Dies wird durch Aktivitäten im Bereich *E-Learning* bekräftigt. Oracle vertreibt sein **Oracle Portal 3.0** [Ora00] als Komplettlösung zur Erstellung eines Portals. Es beinhaltet ein erweiterbares Rahmenwerk für standardisierten Anwendungszugriff, „Self-Service“-Werkzeuge zur Personalisierung, Veröffentlichung und Verwaltung von Informationen und Zugriff auf dynamische Daten. Auch dieses Produkt deckt den B2X Bereich ab. IBM beschreibt seinen **WebSphere Portal Server** [IBM01] als Möglichkeit, ein eigens angepasstes Web-Portal erstellen zu können, um so die Anforderungen von Angestellten, Geschäftspartnern und Kunden erfüllen zu können. Auch der **iPlanet Portal Server** [Ipl01] soll ein skalierbares und sicheres Portal für Kunden, Zulieferer, Partner und Angestellte realisieren. Also liegt auch bei IBM und iPlanet der Fokus auf B2X. Die Produkte von Oracle und IBM können derzeit als marktführend angesehen werden.

Neben den bisweilen aufgezählten kommerziellen Produkten soll auch ein Open Source Projekt untersucht werden. **Jetspeed** [Jet01] ist eine Open Source Implementierung eines EIP basierend auf Java und XML im Bereich B2X und ist mittlerweile auch ein Bestandteil des IBM Websphere Portal-Servers.

4.1 Übersicht

Zuerst soll eine kategorisierte Übersicht über die möglichen Anforderungen gegeben werden. Diese werden dann in Abschnitt 4.2 und Abschnitt 4.3 detailiert beschrieben. Die funktionalen Anforderungen werden auf ihre Wichtigkeit in den Anwendungsbereichen B2B, B2C und B2E untersucht. Nicht funktionale Anforderungen lassen sich wegen ihrer Allgemeingültigkeit dem gesamten B2X Bereich zuordnen.

4.1.1 Übersicht funktionale Anforderungen

Multi Channel Delivery (MCD)

- Zugriff auf das Portal in einem Web-Browser.
- Zugriff auf das Portal via Handy oder Personal Digital Assistant (PDA).
- Unterstützung anderer Ausgabeformate.

Integration

- Datenintegration: Anbindung an unterschiedlichste Datenquellen einschliesslich strukturierter Daten aus relationalen Datenbanken, Data-Warehouses und unstrukturierter Daten aus Texten, Dokumenten und Emails.
- Anwendungsintegration: Anbindung und Integration bestehender Anwendungen.

Personalisierung

- Mehrsprachigkeit der Benutzeroberfläche.
- Systemseitige Personalisierung / Benutzerseitige Personalisierung.
- Agenten zum Auffinden neuer und geänderter Informationen mit anschliessender Benachrichtigung des Benutzers.

Portlets

- Integration / Personalisierung mit Portlets.

Suche / Dokumenten- Knowledge- und Contentmanagement

- Dokumentenmanagement mit Versionierung und Versionshistorien.
- Volltextsuche in möglichst vielen Dokumentenformaten, Datenbanken und externen Daten (Internet).
- Suche und Anzeige ähnlicher Dokumente.
- Knowledge-Sharing.
- Content-Management.

Zusammenarbeit / Entscheidungsfindung

- Unterstützung von Gruppenarbeit (Communities).
- Integrierte Analysemöglichkeiten zur Unterstützung der Entscheidungsfindung.

Administration

- Vorkonfigurierte Bestandteile zur möglichst schnellen Erstellung eines Portals.
- Einfache Verwaltung z. B. über den Web-Browser.

4.1.2 Übersicht nicht funktionale Anforderungen

Investitionssicherheit

- Skalierbarkeit, Clusterfähigkeit und Ausfallsicherheit.
- Lauffähigkeit des Servers auf möglichst vielen Betriebssystemen.

Zukunftssicherheit

- Benutzung offener Standards zur Steigerung der Flexibilität.
- Erweiterbarkeit durch offengelegte Programmierschnittstellen.

Sicherheit / Authorisierung

- Sichere Datenübertragung.
- Nutzung bestehender Benutzerverwaltungen z. B. aus NT-Domänen oder LDAP Servern.

Nach der Übersicht über funktionale und nicht funktionale Anforderungen soll nun näher auf diese eingegangen werden.

4.2 Funktionale Anforderungen

4.2.1 Multi Channel Delivery

Unter *Multi Channel Delivery* (MCD) versteht man die Möglichkeit, Informationen auf verschiedenen Endgeräten ausgeben zu können. Die Informationen werden dazu, wie im Folgenden beschrieben, in ein für das jeweilige Endgerät benötigte Format gebracht.

Zugriff auf das Portal in einem Web-Browser

Da der Zugriff auf das Portal ortsunabhängig erfolgen sollte, bietet sich der Zugriff ohne spezielle Clients über einen Standard-Webbrowser an. Er ist heutzutage auf fast jedem Rechner zu finden und der Benutzer ist mit dessen Benutzung vertraut. Diese Anforderung ist für den gesamten B2X Bereich signifikant.

Zugriff auf das Portal via Handy oder Personal Digital Assistant (PDA)

Das Portal sollte entsprechende Anfragen und Protokolle beherrschen, um auch von Unterwegs auf Informationen zugreifen zu können. Handys und auch PDA benutzen dazu das Wireless Application Protocol (WAP) [WAP00]. Es dient der schnellen Übertragung von Daten an mobile Benutzer. Zur Darstellung wird die *Wireless Markup Language* (WML) als Untermenge der *eXtensible HyperText Markup Language* (XHTML) [XHTML01] verwendet. Bei dieser Art das

Ausgabeformat zu verändern, spricht man, in Anlehnung an die Personalisierung, auch von *Customization*. Wegen der unhandlichen Darstellung im WAP-Format und der eher im Unternehmensbereich verbreiteten PDA ist diese Anforderung mehr im Bereich B2E und B2B zu sehen.

Unterstützung anderer Ausgabeformate

Um die erhaltenen Informationen weiterverwenden zu können ist es sinnvoll, diese auch in anderen Ausgabeformaten wie *eXtensible Markup Language* (XML) [XML01] in Kombination mit *eXtensible Style Language* (XSL) [XSL01], *Postscript* (PS) [PS01] oder *Portable Document Format* (PDF) [PDF01] generieren zu können. Diese erleichtern z. B. den direkten Ausdruck auf einem Postscript-fähigen Drucker oder die Weitergabe der Ergebnisse an Kollegen. Mittels XML/XSL erhält man die Möglichkeit den Inhalt der Informationen von ihrer Repräsentation zu trennen. Aufgrund der so gewonnenen Flexibilität ist diese Anforderung im B2X Bereich sinnvoll. Sie wird von keinem der untersuchten Produkte direkt unterstützt. Jedoch bietet IBM als Zubehör zum WebSphere Server einen *Transcoding Publisher* an. Dieser unterstützt aber lediglich die Konvertierung von HTML in andere HTML-Dialekte wie z.B. WML, *Handheld Device Markup Language* (HDML) oder XML zu XML mit unterschiedlichen XSL Beschreibungen [IBM01b]. Brio vertreibt zur Unterstützung des B2B Bereiches unter der Bezeichnung Brio.Report ein Tool zur Extrahierung und Formatierung von Unternehmensdaten in einen Unternehmensreport. Es werden dabei die Ausgabeformate PS, PDF, HTML, XML, PlainText, Email und Fax unterstützt. Weiterhin lässt sich das Produkt in das Brio.Portal integrieren. Inwiefern dadurch generell Ausgaben in verschiedenenen Formaten möglich werden, geht aus der Produktbeschreibung nicht hervor [Brio01b].

4.2.2 Integration

Datenintegration

In den meisten Unternehmen findet sich eine hochgradig heterogene und mit der Zeit und den Anforderungen gewachsene Infrastruktur. Daher ist es besonders wichtig, bestehende Daten jeglicher Art integrieren zu können. Dazu gehört die Anbindung an weit verbreitete Datenbanksystemen wie Oracle, Informix, Sybase, MS SQL Server und DB2. Aber auch unstrukturierte Daten aus Texten, Dokumenten und Emails sollen verfügbar sein. Zusätzlich besteht die Möglichkeit, Informationen von sogenannten *Content Providern* zu beziehen. Diese liefern z.B. News, Wetterinformationen oder aktuelle Börsenkurse. Oracle bietet beispielsweise als Eigenschaft von Oracle9i eine generische Lösung zur Datenintegration. Diese nutzt Treiber welche die Verbindung zu beliebigen *Open-DataBase-Connectivity* (ODBC) [ODBC01] kompatiblen Datenquellen ermöglicht. So wird auch ein transparenter Zugriff auf „Low-End“ Datenquellen wie MS Access, dBase oder nicht relationale Datenquellen wie MS Excel ermöglicht [Ora01].

Anwendungsintegration

Weiterhin sollte die Möglichkeit bestehen, Daten aus in Unternehmen genutzten *Enterprise Resource Planning* (ERP) Systemen wie SAP R/3 in das Portal integrieren zu können. Zusätzlich besteht der Bedarf Anwendungen wie z.B. die Groupwarelösungen Lotus Notes oder Microsoft Exchange einzubetten. Der Aufwand der Integration kann dabei je nach Komplexität der zu integrierenden Anwendung stark schwanken. Oracle versucht die Anwendungintegration durch das Oracle-InterConnect-Tool zu vereinfachen. Es verspricht unkomplizierte Integration vorhandener

Applikationen (*legacy applications*) mittels grafischer Assistenten und bereits vorgefertigten Adaptern für gängige Applikationen [Ora01b].

Integration ist im gesamten B2X Bereich sehr wichtig, da das Angebot von den integrierten Daten und Anwendungen bestimmt wird.

4.2.3 Personalisierung

Mehrsprachigkeit der Benutzeroberfläche

Da das Internet Zugriffe aus aller Welt ermöglicht, soll auch Wert auf die Mehrsprachigkeit der Benutzeroberfläche gelegt werden. Dies erhöht damit auch die Attraktivität des Portals für anderssprachige Benutzergruppen. So lässt sich die bevorzugte Sprache z. B. in den Einstellungen des Browsers oder im Benutzerprofil festlegen, was sich auch auf die Sprache der Inhalte auswirken sollte.

Systemseitige Personalisierung

Bei der systemseitigen Personalisierung kann der Benutzer keinen Einfluß auf die Art der Personalisierung nehmen. Häufig ist dem Benutzer überhaupt nicht bewusst, dass ihm personalisierte Seiten präsentiert werden (*user-unaware*). Eine Möglichkeit der systemseitigen Personalisierung bieten die sogenannten *Clickstreams*. Die vom Benutzer besuchten Seiten werden protokolliert. Die dadurch erhaltenen impliziten Attribute ermöglichen die Zuordnung eines speziellen Benutzerprofils (z.B. Anfänger, Profi, Techniker) und somit die automatische Anpassung des Inhalts auf die durch den Clickstream bestimmten Interessen. Benutzerprofile können aber auch direkt zugewiesen werden. Dies ist besonders im E-Commerce Bereich interessant, da man so zielgerichtet Produkte bewerben kann.

Benutzerseitige Personalisierung

Bei der benutzerseitigen Personalisierung soll der Benutzer nicht nur die Möglichket haben das „look and feel“ des Portals verändern zu können, beispielsweise durch die Angabe expliziter Attribute wie „Schriftfarbe blau“. Auch der Inhalt und dessen Struktur sollten anpassbar sein. So ist es z. B. sinnvoll, Inhalte durch zusätzliche Filter einschränken zu können. Auch eine Kombination von Daten kann erforderlich werden. In dieser Art lassen sich Informationen wie Nachrichten z. B. mit einem Ticker kombinieren. Dabei sollte es möglich sein, vordefinierte Designvorlagen (*Templates, Themes oder Skins)* zu benutzen oder seine eigenen erstellen zu können. Bei EIP muss sich die Funktionalität unter Umständen soweit einschränken lassen, dass das *Corporate Design* (CD) gewahrt bleibt.

Oracle realisiert dies, in dem grundsätzlich alle Komponenten (Portlets, siehe Abschnitt 4.2.4) des Oracle Portals auf einer „per-user“-Basis personalisierbar sind. Der Entwickler der Komponente spezifiziert entsprechende Parameter und erlaubt dann dem Portalbenutzer die für ihn interessanten Paramenter verändern zu können. Systemseitig existiert dann für jedes Benutzer/ Komponenten Paar eine eigene Konfiguration. Ob der Benutzer die Komponente seinen Bedürfnissen anpassen bzw. darauf zugreifen darf, regelt ein Attribut, welches ihm die Privilegien „Execute“ oder „Customize“ zuweist [Ora00b].

Sehr wichtig bei beiden Personalisierungsarten ist die strikte Trennung zwischen Daten, Funktionalität und Benutzerinterface, da so bei jedem Seitenabruf, je nach Randbedingungen wie z. B. den Benutzereinstellungen, an der ensprechenden Stelle eingegriffen werden kann. Zusätzlich kann nocheinmal zwischen Daten, Struktur, grafischer Repräsentation und Funktionalität unterschieden werden. So lassen sich z. B. die MCD Funktionalitäten leichter realisieren. Mehr Informationen dazu findet man in der Architekturbeschreibung des Jetspeed Portals in Abschnitt 5.1.3 auf Seite 41. Die Personalisierung ist für den gesamten B2X Bereich sehr wichtig, da man so die Flexibilität und die Akzeptanz des Portals deutlich steigern kann.

Agenten zum Auffinden neuer und geänderter Informationen mit anschliessender Benachrichtigung des Benutzers

Wenn sich Informationen, die für einen Benutzer von Interesse sind, ändern, soll dieser darüber informiert werden. Sogennate *„Agenten“* überprüfen daher in regelmäßigen Abständen die Informationen auf Veränderungen. Die Benachrichtigung des Benutzers ist z. B. innerhalb seiner Portalseite, via Email oder per SMS auf dessen Handy möglich. Brio bietet diese Funktionalaität innerhalb des Prio.Portals in Form sogenannter „Proactive Agents“ [Brio01].

Diese Anforderung ist im Zuge der sich immer schneller ändernder Informationen ebenso für den gesamten B2X Bereich interessant.

4.2.4 Portlets

Integration / Personalisierung mit Portles

Unter einem Portlet versteht man aus Benutzersicht einen spezialisierten, eingeschränkten Bereich innerhalb der Portalseite, vergleichbar mit einem Fenster. In diesem Bereich werden die gewünschten Informationen wie z. B. Nachrichten oder Wetterinfos angezeigt. Der Benutzer kann den Inhalt, das Aussehen und die Position des Portlets z. B. gemäß seiner Gruppenzugehörigkeit oder einem vom Administrator vorgegebenen Profil personalisieren. Weiterhin besteht die Möglichkeit, das Portlet-Fenster zu bearbeiten, zu maximieren oder zu minimieren. Aus Server-Sicht kann ein Portlet als eine in einer bestimmten Programmiersprache implementierte Komponente gesehen werden, welche statische oder dynamische Inhalte eines speziellen Themenbereichs innerhalb der Portalseite erzeugt. Der Inhalt wird dabei vom Server z. B. durch den Zugriff auf Datenbanken erzeugt oder stammt von sogenannten Content-Providern [BEA01].

Das Portletkonzept erleichtert somit die Integration von Anwendungen und ermöglicht eine umfassendere Personalisierungsmöglichkeit. So hat der Benutzer die Möglichkeit, aus einer Vielzahl von zur Verfügung gestellten Portlets die für ihn relevanten auszuwählen und innerhalb seiner Portalseite zu positionieren. Die Portlets sind dabei entweder bereits im Potal-Produkt enthalten, stammen von Portletprovidern oder können selbst entwickelt werden. Dabei müssen jeweils die Schnittstellen zum produktspezifischen API eingehalten werden. Oracle legt dabei Wert auf „portalfreundliche Portlets“ welche die drei Aspekte „Size“, „Style“ und „Performance“ ausreichend berücksichtigen sollen. So darf laut Oracle ein Portlet nicht zuviel Platz innerhalb der Portalseite benötigen, um keine anderen Portlets zu verdrängen. Da das Portlet in jede Portalseite integrierbar sein soll, muss eine Trennung von Daten und deren Präsentation vorgenommen werden. Dem Portlet können dann die Designattribute der Portalseite vererbt werden. Das Leistungsverhalten sollte nach Oracle auch nicht vernachlässigt werden. Da alle auf einer Portalseite zusammengefassten Portlets den Aufbau der Seite bestimmen, ist das langsamste Portlet das

bestimmende Moment bei der Generierung des Seiteninhalts [Ora00b]. IBM bermerkt in der Produktbeschreibung des Websphere-Portal-Servers, dass Portlets häufg nur einen Teil einer Anwendung wiedergeben und die vielfältigen Ausgaben und Transaktionen einer Applikation nicht vollständig fassen können. IBM sieht daher die Eignung von Portlets eher für den gelegentlichen Zugriff auf die Applikationsdaten [IBM01]. Oracle macht dahingehend keine Einschränkungen sondern ermöglicht die Zuweisung unterschiedlicher Style-Vorgaben. Jenachdem ob das Portlet gerade Bestandteil einer Portalseite mit anderen Portlets ist oder als „Standalone-Application“ im „Full-Page-Mode“ [Ora00b] fungiert. Das Portletkonzept lässt sich im gesamten B2X Bereich einsetzen. Betrachtet man ein Portlet als Daten und deren grafische Repräsentation, entsteht jedoch das Problem, Informationen mehrerer Portlets nur schwer miteinander kombinieren zu können.

4.2.5 Suche / Dokumenten- Knowledge- und Contentmanagement

Dokumentenmanagement mit Versionierung und Versionshistorien

Mittels einer Versionskontrolle kann sichergestellt werden, das Änderungen an Dokumenten festgestellt und protokolliert werden können. So kann man bei Bedarf den alten Stand wiederherstellen. Zusätzlich sind andere Benutzer in der Lage Änderungen nachzuvollziehen. Erwähnenswert ist an dieser Stelle die Technologie, welche der Hyperwave Information Server zur Speicherung und Verknüpfung von Dokumenten einsetzt. Anders als in konventionellen Systemen werden interne Verweise zur Verknüpfung von Dokumenten (Links) beim Hyperwave Information Server als Datenbankobjekte angelegt. Diese Technologie ermöglicht es, "tote Links" auszuschließen sowie Links dynamisch darzustellen, also z. B. nur dann, wenn der Anwender auch Leserechte auf das Zielobjekt hat. Auch beim Verändern oder Verschieben von Zieldokumenten werden automatisch alle betroffenen Links aktualisiert (*bidirektionale Links*). In vielen Bereichen können Links auch automatisch gesetzt werden: So können Links ohne Administrationsaufwand automatisch auf die vom Anwender gewünschte Sprachversion verweisen oder immer zur aktuellsten Dokumentversion führen. Intelligente Mechanismen aus dem Knowledge-Management ermöglichen es dem Anwender, eine automatisch erstellte Liste relevanter Dokumente zu jedem beliebigen Ausgangsdokument einzusehen und auf diesem Weg zu neuen Wissensquellen zu gelangen. Durch die objektorientierte Betrachtung der Dokumente mit Versionskontrolle, Versionsverlauf und Workflow-Mechanismen wird sichergestellt, dass alle Mitarbeiter auf einer korrekten Informationsgrundlage arbeiten [Hyp01].

Volltextsuche in möglichst vielen Dokumentenformaten, Datenbanken und externen Daten

Wenn viele Informationen zur Verfügung stehen, muss es möglich sein, in diesen nach Inhalten suchen zu können. Daher ist bei unstrukturierten Daten eine Volltextsuche sinnvoll. Dabei finden, neben selbst entwickelten Methoden, Produkte von Autonomy[1], Verity[2], Semio[3] und Inktomi[4] Verwendung [Epi01]. Das schnelle Auffinden von Informationen in einer großen Informationsvielfalt wird so ermöglicht. Die Suche sollte auch die logische Verknüpfung von Suchbegriffen

1. http://www.autonomy.com
2. http://www.verity.com
3. http://www.semio.com
4. http://www.inktomi.com

ermöglichen um die Ergebnissmenge einschränken zu können. Bei einer Suche auf strukturierten Daten kann auch die Möglichkeit z. B. direkte SQL Anfragen an das System stellen zu können Vorteile bringen.

Suche und Anzeige ähnlicher Dokumente

Neben der Volltextsuche kann es auch erforderlich sein, ähnliche oder verwandte Inhalte finden zu können. Dazu werden die vorhandenen Dokumente klassifizert. Mit der daraus erhaltenen Taxonomie [Hyp01] lassen sich dann Dokumente mit ähnlichen Inhalten finden. Der Benutzer erhält also Zugriff auf kategorisierte Dokumentengruppen.

Knowledge-Management

Um das gemeinsame Bearbeiten von Dokumenten zu ermöglichen ist es notwendig, dass der einzelne Benutzer die Dokumente mit Kommentaren versehen kann. Weiterhin ist es sinnvoll die Wichtigkeit der Dokumente bewerten zu können. So ermöglicht beispielsweise der iPlanet-Portal-Server [Ipl01] die Kommentierung von Dokumenten in Form einer Newsgruppen ähnlichen Thread-Form. Auch hier findet man wieder Personalisierungspotential. So kann man diese Kommentare, wie bereits gesagt, in Form von Threads oder nach Datum sortiert ausgeben. Die Bewertung von Dokumenten erleichtert weiterhin das Suchen von Dokumenten, da diese nach ihrer Wichtigkeit, also z. B. nach Anzahl der Kommentare, sortiert angezeigt werden.

Content-Management

Beim Content-Management wird die Gestaltung der Webseiten von den redaktionellen Inhalten getrennt, z. B. in einer Datenbank, verwaltet. Dabei sollte die Gestaltung der Seiten möglichst frei bleiben. Um die Konsistenz der zu publizierenden Inhalte zu sichern spielt auch hier ein Versionsmanagement eine wichtige Rolle. iPlanet hat dazu in Kooperation mit der Firma Interwoven[1] den Content-Manager TeamSite in ihr Portal integriert. Dieser ermöglicht z. B. das Testen von geänderten Inhalten in einer virtuellen Testumgebung, ohne dass das eigentliche System davon beeinflusst wird. Weiterhin wird durch den Einsatz von Templates ein konsistentes Design erreicht. Zudem erleichtert man damit auch unerfahrenen Benutzern den Umgang mit dem Content-Manager [Ipl01b].

Das Dokumenten- und Knowledgemanagement findet seine Anwendung hauptsächlich im B2E Bereich. Suchfunktioalitäten wiederum sind für den B2X - Bereich unerlässlich. Auch das Content-Management erleichtert die Pflege der Daten in allen Bereichen.

4.2.6 Zusammenarbeit / Entscheidungsfindung

Integrierte Analysemöglichkeiten zur Unterstützung der Entscheidungsfindung

Um die Entscheidungsfindung zu erleichtern, ist es hilfreich die Vielzahl von Informationen und Daten analysieren zu können. So kann man z. B. durch die Generierung von Charts Informationen übersichtlicher gestalten. Analog zur Benachrichtigung bei geänderten Daten kann der Benutzer sich z. B. auch benachrichtigen lassen, wenn vorgegebene Werte einen zulässigen Bereich verlassen [Brio01].

1. http://www.interwoven.com

Unterstützung von Gruppenarbeit (Communities)

Gruppen sollen mit speziell für sie zusammengestellten Informationen versorgt werden. Weiterhin sollen Aspekte wie Kalender, Messageboards und Chat-Räume die Gruppenarbeit unterstützen. So erleichtert man z. B. das Auffinden eines kompetenten Ansprechpartners für bestimmte Probleme. Hyperwave bietet integrierte Foren an, welche von der Funktionalität mit üblichen Newsgruppen vergleichbar sind. Durch die Integration in den HIS wird gleichzeitig die Suche nach darin enthaltenen Informationen erleichtert. Weiterhin können einzelne Bereiche vor dem Zugriff nicht berechtigter Benutzer geschützt werden. Für kritische oder dringende Situationen bietet Hyperwave die Funktion „Chatting with the expert“. Die Suche nach Informationen führt dann über den Autor gefundener Dokumente direkt zu einem passenden Ansprechpartner. Diesen kann man dann via E-Mail kontaktieren oder ihn mittels der integrierten Microsoft-Netmeeting-Technologie zu einem Chat einladen. Weiterhin lässt sich zur Verbesserung der Zusammenarbeit ein Workflow definieren. So können Dokumente von einer Kette vordefinierter Benutzer verbreitet, geprüft, editiert und freigegeben werden [Hyp01].

4.2.7 Administration

Vorkonfigurierte Bestandteile zur möglichst schnellen Erstellung eines Portals

Um nach dem Entschluss ein Portal einzusetzen möglichst schnell am Ziel seiner Vorstellungen zu sein, ist es hilfreich wenn bereits vorkonfigurierte Komponenten zur Datenanbindung und zur Darstellung der Informationen vorhanden sind. Es können z. B. Templates zur Verfügung gestellt werden welche nur noch geringfügig angepasst werden müssen. Durch die so schon vorhandene Basis lässt sich Entwicklungszeit einsparen.

Management und Verwaltung des Portals im Browser

Um die Administrationskosten senken zu können, sollen alle administrativen Aufgaben zentral im Web-Browser stattfinden können. Zusätzlich ist es für den Groupware-Aspekt wichtig, dass jeder Teilnehmer seinen Bereich selbst Administrieren kann.

Wir haben nun einen Überblick über charakteristische funktionale Anforderungen. Die untersuchten Portale sollen nun auch auf nicht funktionale Anforderungen untersucht werden.

4.3 Nicht funktionale Anforderungen

4.3.1 Investitionssicherheit

Skalierbarkeit, Clusterfähigkeit und Ausfallsicherheit

Da sich im Vorfeld nicht immer bestimmen lässt, wie leistungsfähig ein Portalsystem bezüglich der Zugriffszahlen und des Datendurchsatzes sein muß, ist es wichtig das System dahingehend erweitern (*skalieren*) zu können. So sollte sich das System auch über mehrere Rechner verteilt einsetzen lassen (*clustering*) um die Last gleichmäßig aufteilen zu können. Weiterhin kann man so durch Redundanz die Ausfallsicherheit des Systems erhöhen. Fällt ein Rechner aus, übernimmt ein anderer seine Aufgaben (*fail-over*).

Lauffähigkeit des Servers auf möglichst vielen Betriebssystemen

Wie bereits bei der Datenanbindung erwähnt, findet man in den Unternehmen stark heterogene Systemumgebungen. Um Kunden nicht auf eine spezielle Hardware oder Betriebssysteme festlegen zu müssen, sollte das Portal dahingehend möglichst flexibel ausgelegt sein. Zu den am weitesten verbreiteten Betriebssystemen zählen Linux, SUN Solaris, IBM AIX, HP-UX, DEC UNIX sowie Windows NT 4.0 und Windows 2000.

4.3.2 Zukunftssicherheit

Benutzung offener Standards zur Steigerung der Flexibilität

Das Portal sollte mittels offener Standards entwickelt werden um eine möglichst große Flexibilität bezüglich Erweiterbarkeit und Integration bestehender Anwendungen erreichen zu können. Einige wichtige Standards sind HTTP, SSL, *Lightweight Directory Access Protocol* (LDAP) [Webo01b], WAP, HTML, WML, JavaScript [JS01] und MIME. Besonderes Augenmerk wird auf XML gelegt. XML wird als zukunftsträchtige Entwicklung betrachtet, Daten flexibel austauschen und speichern zu können. Mehr Informationen zu XML finden sich in Kapitel 5.

Erweiterbarkeit durch offengelegte Programmierschnittstellen

Zusätzlich zur Benutzung offener Standards müssen für die Erweiterbarkeit durch den Kunden die Programmierschnittstellen (*Application-Program-Interface*, API) offengelegt werden. Mit Hilfe eines *Software-Developer-Kit* (SDK) kann der Kunde dann selbstständig Erweiterungen und Anpassungen durchführen. Dies wird weiterhin dadurch erleichtert, wenn bei der Programmierung weit verbreitete Sprachen wie Java oder C/C++ zum Einsatz kommen.

4.3.3 Sicherheit / Authorisierung

Nutzung bestehender Benutzerverwaltungen aus NT-Domänen oder LDAP Servern

Durch die Nutzung bestehender Authorisierungssysteme lassen sich redundante Benutzerdaten-Sammlungen vermeiden. Weiterhin muss sich der Benutzer nur ein Login und ein Passwort merken (*single-signon*), was zusätzlich den Administrationsaufwand der Systeme einfacher macht, da die Zugangsdaten z. B. weniger oft von den Benutzern vergessen werden und wieder zurückgesetzt weden müssen.

Sichere Datenübertragung

Da insbesondere bei Unternehmensportalen auch sensible Daten vorkommen können, müssen diese abhörsicher und vor nicht authorisierter Veränderung geschützt übertragen werden. Daher wird dort das HTTPS Protokoll eingesetzt, welches wie bereits erwähnt, SSL zur Verschlüsslung der Daten einsetzt.

Wegen ihrer Allgemeinheit werden die nicht funktionalen Anforderungen dem gesamten B2X Bereich zugeordnet. Zum Abschluss verdeutlicht Tabelle 2 auf Seite 35, welche Produkte welche Anforderungen erfüllen und in welchem Anwendungsbereich die Anforderungen wichtig sind. Wie sich ablesen lässt, sind die Produkte von Hyperwave und Oracle, bezüglich der von uns untersuchten Anforderungen, am besten ausgestattet. Schlußlicht bildet das Open-Source Projekt Jetspeed welches sich aber in fortlaufender Entwicklung befindet und mittlerweile auch in den IBM Websphere Portal-Server integriert wurde.

Tabelle 2 Funktionale und nicht funktionale Anforderungen der Untersuchten Portale

	Zugriff in einem Web-Browser	Zugriff via Handy oder PDA	Unterstützung anderer Ausgabeformate	Datenintegration	Anwendungsintegration	Mehrsprachige Benutzeroberfläche	Benutzerseitige / Systemseitige Peronalisierung	Auffinden geänderter Informationen	Portlets	Dokumentenmanagement	Volltextsuche	Suche ähnlicher Dokumente	Knowledge-Management	Content-Managemnt	Gruppenarbeit	Integrierte Analysemöglichkeiten	Vorkonfigurierte Bestandteile	Einfache Verwaltung z.B. im Web-Browser	Skalierbarkeit, Clusterfähigkeit	Ausfallsicherheit	Lauffähigkeit auf möglichst vielen Plattformen	Benutzung offener Standards	Offengelegte Programmierschnittstellen	Sichere Datenübertragung	Nutzung bestehender Benutzerverwaltungen
Brio.Portal 7.0	✓	✓	❍	✓	✓		✓	✓			✓	❍	✓		❍	✓		✓	✓			✓	✓		✓
Epicentric Foundation Server 3.5	✓	✓		✓	✓	✓	✓				✓	✓					✓	✓	✓	✓	✓	✓	✓		
Hyperwave Information Server 5.5	✓	✓		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓
IBM Websphere Portal Server 1.2	✓	✓	❍	✓	✓		✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
iPlanet Portal Server	✓	✓		✓			✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	✓	✓
Oracle Portal 3.0	✓			✓	✓	✓	✓		✓	✓	✓	❍	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
Jetspeed 1.3a1	✓	✓		✓	❍	✓	✓	✓	✓					❍	❍		✓		❍	❍	✓	✓	✓	❍	
ANWENDUNGSBEREICH	B2X	B2B, B2E	B2X	B2X	B2X	B2X	B2X	B2X	B2X	B2E	B2X	B2E	B2E	B2X	B2B, B2E	B2B, B2E	B2X	B2X	B2X	B2X	B2X	B2X	B2X	B2X	B2X

Anforderungen	
✓	vorhanden
❍	in Ansätzen
	nicht vorhanden

Nachdem nun funktionale und nicht funktionale Anforderungen an Portale herausgearbeitet wurden, sollen in Kapitel 5 Architekturen vorgestellt werden um diese Anforderungen optimal realisieren zu können.

Kapitel 5 Basistechnologien

Wie wir bereits gesehen haben, besteht die Hauptaufgabe bei der Enwicklung von Portalen darin, Daten aus verschiedensten Informationsquellen für den Benutzer personalisiert aufzubereiten. So kommen beim Portalentwurf viele Softwaretechnologien zum Einsatz. Zum Beispiel aus den Bereichen Datenbanken, Systemintegration und verteilte Systeme. Einige typische Komponenten in der Softwarearchitektur von Portalen dienen beispielsweise der Web-Präsentation, der Personalisierung und der Anbindung an bereits bestehende Systeme wie die ERP Software SAP R/3 [Has01]. Diese Komponenten lassen sich wie im Abschnitt “Das n - Schichtenmodell” auf Seite 13 beschrieben in Schichten anordnen. Im Folgenden soll dies Anhand von drei Beispielarchitekturen verdeutlicht werden. Zum einen soll eine von IBM enwickelte Architektur [Wol01][IBM01] vorgestellt werden. Zum Vergleich die des Hyperwave Information Servers (HIS) und die des Open-Source Projekts Jetspeed. Abschliessend soll versucht werden anhand der Beispiele eine allgemeine Portalarchitektur zu erarbeiten.

5.1 Beispiele für Portalarchitekturen

5.1.1 IBM Portalarchitektur

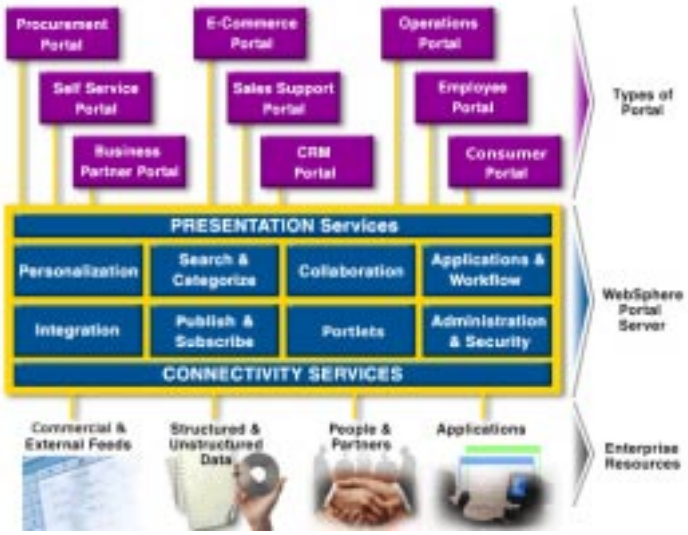
IBM unterscheidet in seiner Portalarchitektur vertikale und horizontale Portale. Ein **horizontales Portal** stellt dabei die in Kapitel 4 vorgestellten funktionalen Anforderungen zur Verfügung. **Vertikale Portale** unterscheiden sich davon durch ihren engen Fokus auf spezielle Themen und werden oft nur einzelnen Funktionalitäten von horizontalen Portalen gerecht. Die darunterliegende Technologie kann jedoch die eines vollständigen horizontalen Portals enthalten. Die IBM Architektur lässt die vertikalen Portale auf der Architektur der horizontalen Portale aufsetzen. Somit stellen die vertikalen Portale eine Verfeinerungen dar. Die horizontalen Portale erhalten ihre Daten dabei aus den verschiedensten Datenquellen wie Anwendungen, Partnern oder externen Daten. Eine Veranschaulichung findet sich in Abbildung 13 auf Seite 38.

Nun soll die Schicht „Dienste eines horizontalen Portals“ näher betrachtet werden. In ihr befinden sich die oben erwähnten Funktionalitäten eines horizontalen Portals welche von allen Portalen erfüllt werden sollen. Die mittlere Schicht unterteilt sich dabei weiter in die Bereiche „Dienste zur Darstellung (Presentation Services)“, die eigentlichen Module zur Realisierung der Funktionen und die „Dienste zur Anbindung (Connectivity Services)“. Der untere Bereich realisiert also die funktionalen Anforderungen der Datenintegration und soll in der Lage sein, mög-

lichst beliebige Datentypen anbinden zu können. Dazu gehören insbesondere externe Daten aus dem Internet, unternehmensinterne operative Daten, von Mitarbeitern und Partnern zugesteuerte Daten und die in Anwendungen wie SAP R/3 oder anderen ERP Systemen enthaltenen Informationen. Nähere Informationen über die Anwendungsintegartion findet man in Kapitel 7.

Abbildung 13 Unterschied zwischen vertikalen und horizontalen Portalen

Abbildung 14 IBM Portalarchitektur (Websphere Portal Server 1.2)



Die Präsentation und die Darstellung aller Daten und Anwendungen übernimmt der Bereich „Dienste zur Darstellung“. Hier finden sich die funktionalen Anforderungen des Bereichs MCD.

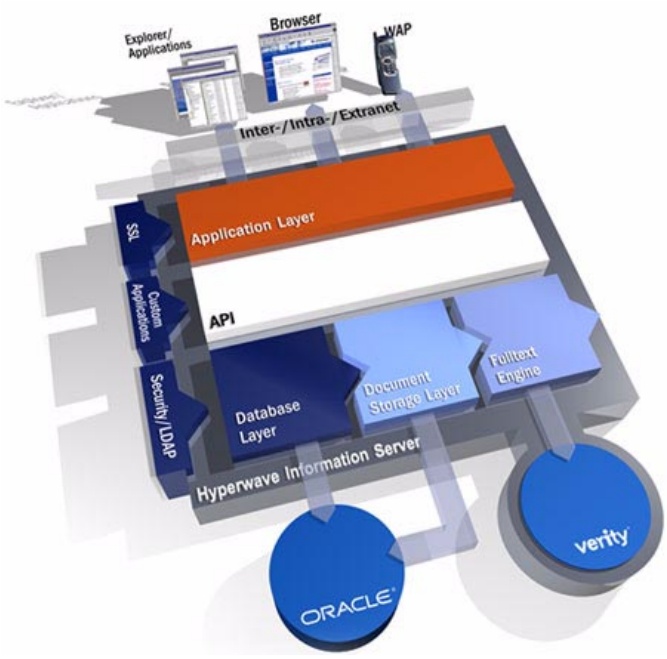
Die von den Funktionsmodulen generierten Ausgaben werden bei Bedarf an den enstpreche-neden Client angepasst. Die restlichen Anforderungen werden in der Schicht zwischen den Presen-tation- und Connectivity Services realisiert. Eine komplette Übersicht über die IBM Portalarchitektur zeigt Abbildung 14. Die IBM Architektur zeigt auf einem relativ hohen Abstraktionsgrad, wie sich vertikale Portale aus einem horizontalen Portal ableiten lassen.

5.1.2 Hyperwave Information Server Portalarchitektur

Die Architektur des Hyperwave Information Servers (HIS) besteht ebenfalls aus verschiedenen Schichten. In der obersten Schicht, dem *Application Layer*, wird die Funktionalität des Portals realisiert. Weiterhin können, neben der Absicherung der Daten durch Benutzer-Authentifizierung (z.B. via LDAP), mit Hilfe eines SSL Moduls Ausgaben, welche über das Inter-, Intra- oder Extranet übermittelt werden, geschützt übertragen werden. Die Ausgaben werden hier auch an den entsprechenden Client, wie z. B. einen Web Browser oder ein WAP Handy, angepaßt.

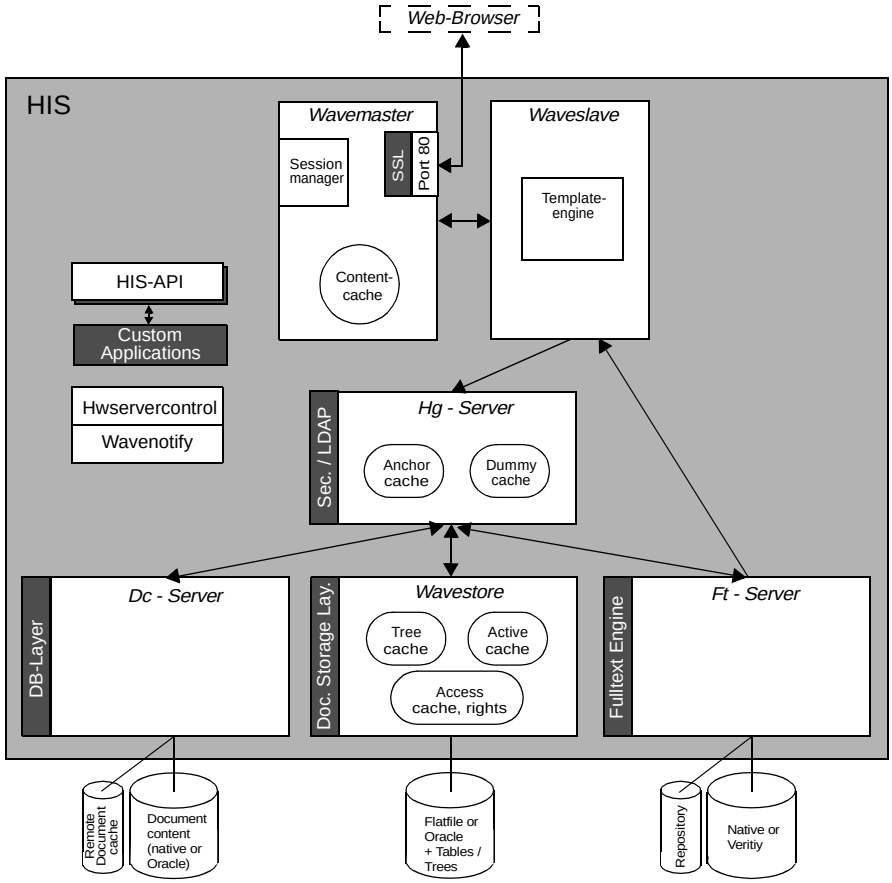
In der nächsten Schicht wird ein API bereitgestellt. Bei dem von Hyperwave zur Verfügung gestellten API und dem dazugehörigen SDK handelt es sich um eine Multi-Betriebssystem und Multi-Programmiersprachen Sammlung von HIS Funktionen, die zur Erstellung oder Integration von eigenen Applikationen oder Clients benutzt werden können.

Abbildung 15 Architektur des Hyperwave Information Server



Zugriff auf die Daten erfolgt über den *Database Layer*, den *Document Storage Layer* oder über eine *Fulltext Engine*. Der Database Layer ermöglicht direkten Zugriff auf in einer Datenbank (z.B. Oracle) abgelegte Daten. Erfolgt der Zugriff auf die Datenbank über den Document Storage Layer so kann z.B. eine Versionskontrolle von Dokumenten erreicht werden. Einen indirekten Zugriffsweg stellt der Weg über die Fulltext Engine von Verity dar. Diese durchsucht und indiziert die Datenbank nach enthaltenen Texten und stellt Referenzen auf die Daten bereit. So wird eine schnelle Volltextsuche über die vorhandenen Daten realisiert. Abbildung 15 zeigt eine Übersicht über die Architektur des HIS. Sie ist im Vergleich zur IBM Architektur weniger allgemein. Der HIS realisiert dabei auch ein horizontales Portal das je nach Anwendungsfeld zum vertikalen Portal werden kann. Abbildung 16 veranschaulicht die interne Realisierung. Die grauen Bereiche mit weißer Schrift zeigen die Relationen zur Architekturübersicht in Abbildung 15.

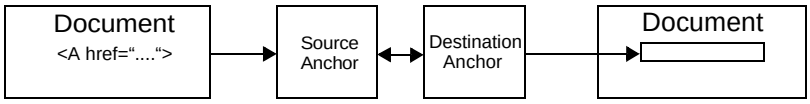
Abbildung 16 Realisierung der Architektur



Der HIS besteht intern aus den Komponenten *Wavemaster*, *Waveslave*, *Hg-server*, *Ft-Server*, *Wavestore*, *Dc-Server*, *Hwservercontrol*, *Wavenotify* und *HIS-API*. **Wavestore** dient zur Sicherung und Administration aller Attribute und Metadaten (access cache, rights), speichert den Pfad (DC-Pfad) zu den in der Dc-Server Datenbank befindlichen (tree cache), sowie den derzeit geöff-

neten Dokumenten (active cache) und implementiert die Suche nach Schlüsselwörtern. Kommunikation findet nur mit dem Hg-Server statt. Dieser erfragt z.B. bei Zugriffen auf Dokumente deren Attribute um die Zugriffsrechte prüfen zu können. Der **Dc-Server** speichert und administriert die eigentlichen Dokumente. Dabei können die Dokumente im lokalen Datenbestand enthalten sein oder in einem Cache, welcher Dokumente aus anderen Datenbeständen vorhält. Damit können Aspekte der Datenintegration realisiert werden. Der **Ft-Server** erstellt und beinhaltet einen Index um eine Volltextsuche implementieren zu können. Um den Zugriff zu beschleunigen, wird zusätzlich ein Repository angelegt, in dem Attribute wie Sprache, MIME-Type und der DC-Pfad der Dokumente zu finden sind. Der **Hg-Server** verteilt die vom Waveslave erhaltenen Kommandos zu Wavestore, sowie dem Ft- und Dc-Server. Weiterhin wird hier die Benutzeridentifikation koordiniert. Wie bereits in Kapitel 4 erwähnt, legt der HIS Referenzen auf Dokumente als bidirektionale Links an. Ein Dokument besitzt dabei jeweils einen *Source-Anchor* und einen *Destination-Anchor*. Dies veranschaulicht Abbildung 17.

Abbildung 17 Source- und Destination-Anchor beim HIS



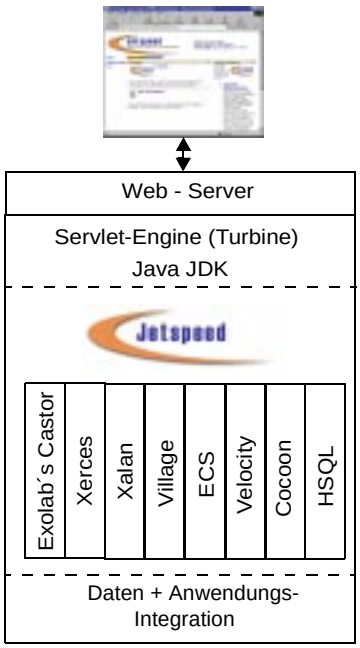
Der **Hg-Server** enthält neben einem Anchor-Cache, in dem die Source-Anchor gespeichert werden, einen Dummy-Cache. Ein Dummy-Anchor verweist auf Dokumente in anderen Servern da der HIS auch in einem Server-Pool von mehreren HIS-Servern betrieben werden kann. Der HIS ist also skalierbar und man erhält zusätzlich eine erhöhte Ausfallsicherheit. **Wavemaster** und **Waveslave** verwalten Benutzersessions, holen alle benötigten Informationen und verarbeiten Templates. An dieser Stelle setzen die Personalisierung und die Anpassung an verschiedene Ausgabeformate an. Der Content-Cache innerhalb des wavemasters sorgt für die schnelle Bereitstellung häufig angefragter Dokumente. Die **Hwservercontrol** Komponete stellt den ordnungsgemäßen Start aller benötigten Prozesse und deren Überwachung sicher. **Wavenotify** benachrichtigt Benutzer via E-Mail über veränderte Dokumente und realisiert somit die in Kapitel 4 vorgestellte Agentenfunktionalität. Weitere Möglichkeiten zur Personalisierung und zur Anwendungsintegration liefert das bereits erwähnte SDK in Verbindung mit dem **HIS-API**.

5.1.3 Jetspeed

Jetspeed ist eine Open Source Implementierung eines EIP. Die Jetspeedarchitektur basiert auf einer Sammlung von anderen, bereits bestehenden Open Source Projekten und ist Bestandteil des Jakarta Projekts[1]. Die Zielsetzung dieses Projekts ist die Entwicklung von qualitativ mit kommerziellen Produkten vergleichbaren Open Source Lösungen. Jetspeed setzt bei seiner Realisierung auf XML und Java und erreicht so eine weitgehende Plattformunabhängigkeit.

1. http://jakarta.apache.org

Abbildung 18 Jetspeed Architektur



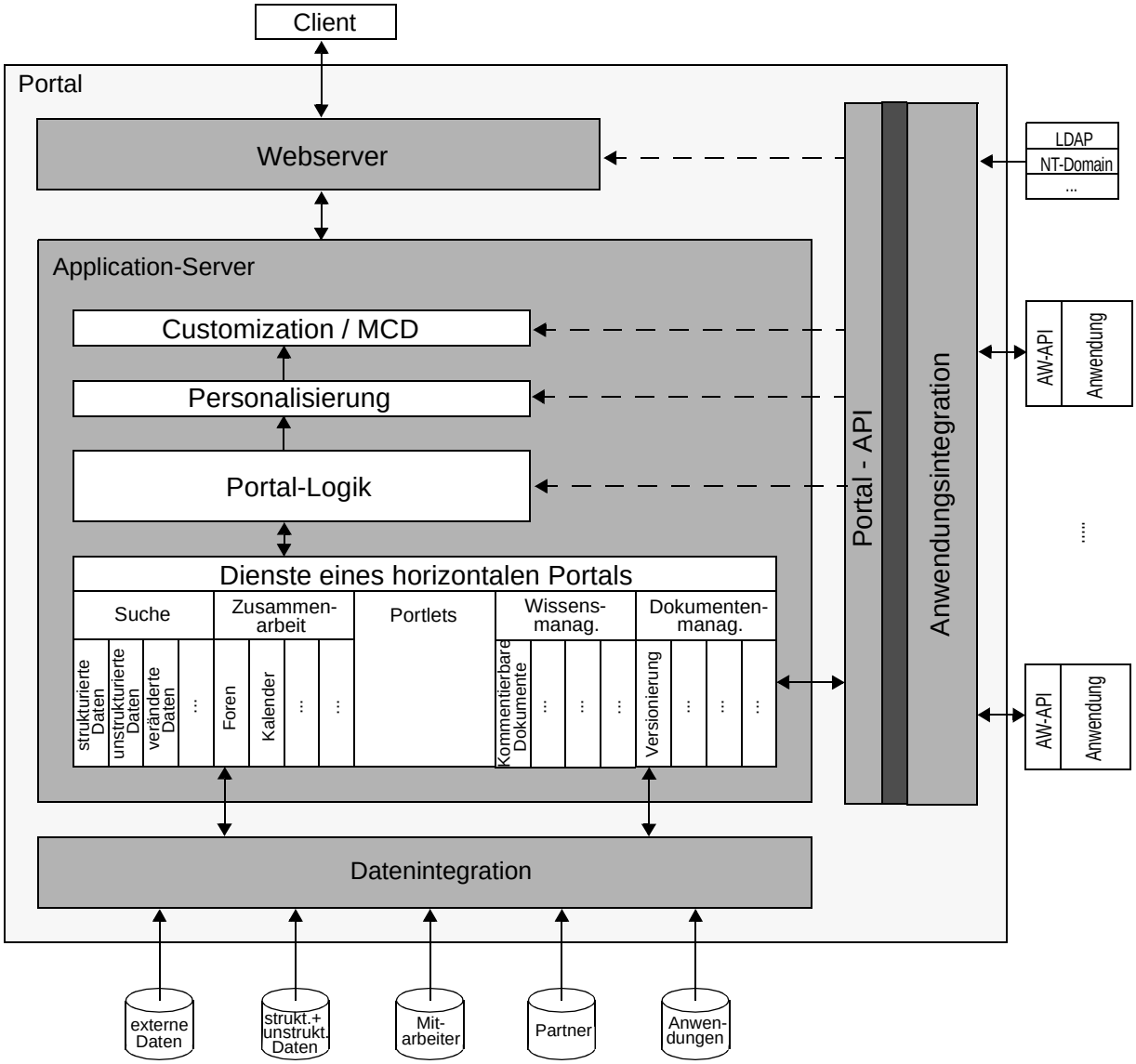
Die eigentliche Jetspeed-Anwendung ist eine Servletsammlung und wird in einer Servlet-Engine abgearbeitet. Ein Servlet ist ein Java Programm, welches in einer virtuellen Maschine, der Servlet-Engine, ausgeführt wird. Die Servlet-Engine ist plattformunabhängig und sorgt so für einen Freiraum bei der Auswahl der einzusetzenden Servertechnologie. Servlets sind vergleichbar mit den heute häufig eingesetzten Java-Applets, mit dem Unterschied, dass Applets Client-seitig im Browser abgearbeitet werden. Ein Servlet ist somit eine Alternative für CGI Programme. Weiterhin läuft ein Servlet permanent und kann gleichzeitig mehrere Verbindungsanfragen bearbeiten. Der Prozess muss also nicht laufend gestartet und beendet werden. Dies wirkt sich positiv auf die Geschwindigkeit des Systems aus. Die Servlet-Engine benötigt dabei eine Java-VM, z. B. aus dem *Java Development Kit* (JDK). Für den Datenaustausch mit dem Web-Browser sorgt ein Web-Server. Einen stark abstrahierten Überblick über die Architektur liefert Abbildung 18. Eine genaue Betrachtung der Jetspeed-Architektur sowie deren Bestandteile findet man in Kapitel 6.

5.1.4 Allgemeine Portalarchitektur

Nachdem nun drei Architekturen betrachtet wurden, soll daraus eine allgemeine Architektur entworfen werden. Wie bei den Betrachteten soll auch hier die in Abschnitt “Das n - Schichtenmodell” auf Seite 13 vorgestellte Schichtenarchitektur Verwendung finden. Eine grafische Übersicht der allgemeinen Architektur liefert Abbildung 19. Die Grundlage für ein Portal bildet immer ein *Webserver* welcher im oberen Bereich der Architektur angesiedelt ist. Oft übernimmt dieser auch die Aufgaben der Benutzerauthentifizierung, wobei, wie beim HIS, zusätzlich LDAP-Server oder andere bestehenden Benutzerverwaltungen integriert werden können. Als zweite allgemeine

Komponente lässt sich ein *Application-Server* identifizieren. In diesem werden die eigentlichen Portalfunktionalitäten realisiert. Bei Jetspeed wird dieser z. B. durch die Servlet-Engine repräsentiert. Der Application-Server enthält die Bestandteile zur Personalisierung sowie zum Anpassen der Ausgaben an verschiedene Ausgabegeräte. Weiterhin werden diese Funktionalitäten in der Regel durch ein Portal-API unterstützt, wodurch ein Eingriff auf Programmiersprachenebene ermöglicht wird. Dies erhöht zusätzlich die Flexibilität. Anwendungsintegration findet ebenfalls unter Benutzung des Portal-API sowie der anwendungsspezifischen API statt.

Abbildung 19 Allgemeine Portalarchitektur



Neben den Bestandteilen zur Realisierung der funktionalen Anforderungen enthält der Application-Server noch die Portal-Logik, die den Charakter des Portals bestimmt. Sie variiert von Anbieter zu Anbieter. Der untere Bereich der Architektur widmet sich der *Datenintegration*. Dieser soll die Integration vieler unterschiedlicher Datenquellen ermöglichen.

Nach der Erstellung einer allgemeinen Portalarchitektur soll nun weiter auf die Implementierung des Beispielsystems für das Handelssystem für Warentermingeschäfte hingearbeitet werden. Dazu wird im folgenden Kapitel Jetspeed näher vorgestellt um nach der Analyse der Anforderungen des Beispielsystems in Kapitel 7 die Eignung von Jetspeed für die spätere Realisierung des Handelssytems abschätzen zu können.

Kapitel 6 Jetspeed

Das Open Source Project „Jakarta“[1] versucht Software zu erstellen, welche sich qualitativ mit kommerzieller Software vergleichen lässt. Ein Teil dieser Bemühungen ist Jetspeed, ein Open-Source Enterprise-Information-Portal (EIP). Bei einem EIP liegt der Einsatzschwerpunkt bei Unternehmen, welche darin Informationen für alle am Geschäftsprozeß teilnehmenden Parteien, vom Kunden über Geschäftspartner bis hin zum Angestellten präsentieren können. Dabei wird besonderes Augenmerk auf die Suche und Kategorisierung von Informationen, die Personalisierung sowie die Anwendungsintegration und Sicherheit gelegt.

Jetspeed bietet dazu eine Benutzerverwaltung mit der Möglichkeit einer Einteilung in Rollen, die mit individuellen Berechtigungen augestattet werden können. Ein Gruppenkonzept befindet sich derzeit in der Entwicklung. Die Portalseite setzt sich aus Portlets zusammen, welche vom Benutzer individuell ausgewählt werden können. Die Seite lässt sich weiterhin mit dem sogenannten „Customizer“ personalisieren. Dabei lassen sich das Aussehen, der Inhalt und die Anordnung der Portlets anpassen. In der Version 1.2b des Jetspeed-Portals ist der iCalendar als Groupwaretool enthalten. In der aktuellen Entwicklung steht dieser jedoch nicht zur Verfügung. Jetspeed hat derzeit keine besonderen Mechanismen zur Anwendungsintegration. Es finden sich weder komponentenbasierte Ansätze zur Anwendungsintegration noch irgendwelche Middleware. Lediglich als Anwendung gekennzeichnete Portlets erfahren eine Sonderbehandlung bei der Darstellung. Die Integration von Anwendungen muss also vollständig selbst vorgenommen werden. Einen SOAP-basierten Integrationsansatz findet man daher in Kapitel 11.

Die Realisierung des Jetspeed-Portals erfolgt als Sammlung von Java Servlets und ist XML unterstützt. Leider findet man, im Gegensatz zu anderen kommerziellen Produkten, derzeit kaum aktuelle Dokumentation und auch Installationsroutinen sucht man vergeblich. Eine Installationsanweisung für Linux findet sich daher in Anhang A. Die hier untersuchte Version stammt aus dem CVS-Bereich und hat den Entwicklungsstand 06.11.2001. Die für die Untersuchungen eingesetzte Jetspeed-Umgebung läuft unter Linux und nutzt Jakarta-Tomcat 4.0b7 als Servlet-Engine und Web-Server. Java-seitig kommt J2SDK 1.4.0 zum Einsatz. In diesem Kapitel sollen die Bestandteile näher vorgestellt werden. Weiterhin soll nach der Analyse des Informationsflusses eine detailierte Architekturübersicht gegeben werden.

1. http://jakarta.apache.org

6.1 Bestandteile

Die Jetspeed Servlets bedienen sich Bibliotheken anderer Open Source Projekte. Es kommen „Exolabs Castor“, „Cocoon“, „Turbine“, „Xerces“, „Xalan“, „Village“, „ECS“, „HSQL“ und „Velocity“ zum Einsatz, welche im Folgenden kurz beschrieben werden sollen. Die letzten zuverlässigen Informationen über den Einsatz der einzelnen Komponenten in Jetspeed stammen aus einem älteren Artikel [Jet00]. Einen tieferen Einblick über den aktuellen Stand erhält man durch das Lesen der entsprechenden Mailing-Listen [TMA01].

Exolab´s Castor

Exolab´s Castor [Exo01] ermöglicht die Erstellung von Java-Objekten aus XML-Daten und umgekehrt. Jetspeed benutzt Castor um Registrierungen wie die PSML-Portlet-Registry oder die zur Personalisierung verwendeten PSML-Informationen in Java-Klassen zu konvertieren. Castor dient also zur Konfiguration des Portals und ermöglicht eine für jeden Benutzer individuelle Personalisierung (siehe Abschnitt “PSML” in Kapitel “6.2.2“).

Xerces & Xalan

Xerces und Xalan werden im Apache XML Projekt [XML01b] entwickelt. Xalan ist ein *eXtensible Stylsheet Language Transformations* (XSLT) [XSLT01] Prozessor für Java, der XML-Dokumente mittels Stylesheets in HTML-, Text- oder andere XML-Dokumente transformieren kann. Weiterhin sind in Xalan die W3C [W3C01] *XML Path Language* (XPath) Empfehlungen implementiert. XPath dient zum Verweis auf bestimmte Stellen in XML-Dokumenten. Xerces ist ein XML-Parser für Java, welcher unter anderem die Standards „*Simple API for XML“* (SAX) [SAX00] und *Document Object Model* (DOM) [DOM01] unterstützt. Xerces und Xalan werden beispielsweise in der Jetspeed-Javaklasse „SimpleTransform“ genutzt um ein XML-Dokument mittels XSL [XSL01] zu transformieren.

Cocoon

Cocoon ist ebenfalls Teil des Apache XML Projekts. Es realisiert die Generierung von Seiten aus XML und XSL Dokumenten. Cocoon trennt dabei Inhalt, Darstellung und Logik der Dokumente mit Hilfe von XSLT. Intern basiert Cocoon auf dem in Abbildung 20 auf Seite 48 gezeigten Reactor-Design-Pattern welches sich aus *Request, Producer, Reactor, Processor, Formatter* und *Response* zusammensetzt:

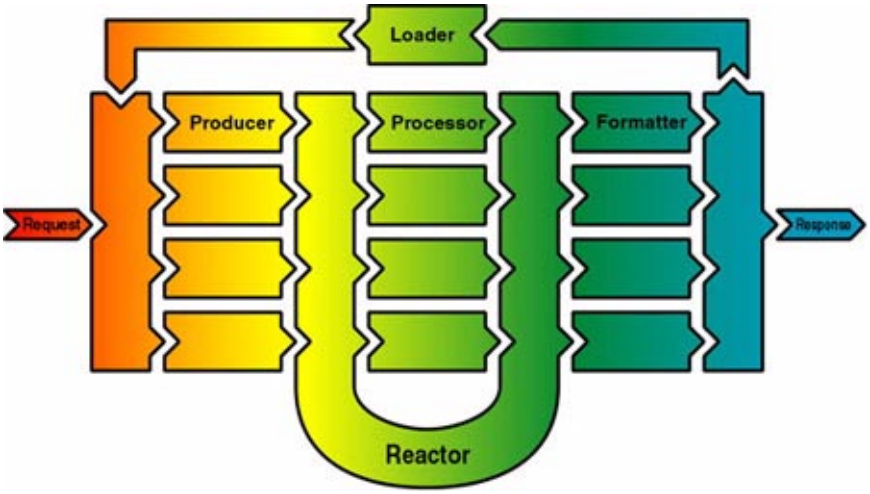
- **Request:** Enthält die Anforderung vom Client und alle für die Verarbeitung notwendigen Informationen. Wichtige Informationen sind die Art des anfragenden Clients, die URI[1] der Anfrage und welcher Producer zur Verarbeitung eingesetzt werden soll.
- **Producer:** Verarbeitet die angegebene URI und erzeugt ein XML-Dokument. Producer haben einen Plugin-Charakter und orientieren sich am Abstract-Factory-Design-Pattern [Gam96]. Jeder kann so seine eigenen Producer erstellen. Das durch den Producer erstellte XML-Dokument kommt dann zur weiteren Verarbeitung in den Reactor.

1. Uniform Resource Identifier: Referenz auf ein Objekt

- **Reactor:** Wertet aus, welcher Prozessor entsprechend den XML-Bearbeitungsinstruktionen (engl. processing instructions, PI) auf das Dokument angewendet werden soll. Das Reactor-Pattern unterscheidet sich hier von einer herkömmlichen Abarbeitung in einer Pipeline, da der Reactor dynamisch ausgewählt werden kann. So muss nur der gewünschte Prozessor aufgerufen werden. Dies wirkt sich positiv auf die Verarbeitungsgeschwindigkeit aus. Nach der Bearbeitung im Reactor muss dieser das Dokument an einen passenden Formatter weiterleiten.
- **Formatter:** Transformiert die im Speicher befindliche XML-Repräsentation für die Übermittlung an den Client. Der Typ des generierten Dokuments hängt von der Wahl des Formatters ab, um die speziellen Möglichkeiten der verschiedenen Endgeräte unterstützen zu können (HTML, WML, ...).
- **Response:** Ergänzt das formatierte Dokument mit weiteren zur Übermittlung notwendigen Eigenschaften, wenn nicht schon vorhanden (z.B. Länge, MIME-Type, etc.).
- **Loader:** Lädt ein formatiertes Dokument wenn es sich um ausführbaren Code handelt. Dies ermöglicht die Nutzung von Serverseiten, bei denen vom Inhalt separierte Logik in einem Producer kompiliert und zusammengeführt wird. In diesem Falle wird das Dokument nicht direkt an den Client zurückgesendet, sondern erneut geladen und als Dokument-Producer ausgeführt.

Cocoon lässt sich dabei „Offline“ von der Kommandozeile, als Servlet oder als eigenständigen Server betreiben. Hauptsächlich kommt Cocoon zur Generierung von dynamischen HTML Dokumenten zum Einsatz. Diese enstehen durch die Verarbeitung statischer oder dynamisch generierter XML Dateien. Cocoon kann aber auch komplexere Transformationen durchführen. So können z. B., je nach angewendetem Formatter, PDF Dateien erzeugt oder die Ausgabe an WAP-Devices angepasst werden. Weiterhin können so auch mehrere Repräsentationen desselben Inhalts erstellt werden. Von Cocoon generierte Inhalte lassen sich mit Hilfe von Turbine ähnlich den von Velocity oder JSP erzeugten Templates in Jetspeed integrieren. Die Fähigkeiten von Cocoon ermöglichen z. B. die im vorigen Kapitel vorgestellten MCD Funktionalitäten. Die Rolle von Cocoon innerhalb der jetzigen Jetspeed-Entwicklung ist, wie sich in den Mailing-Listen erfahren lässt, derzeit aber als gering anzusehen. Cocoon war jedoch in früheren Versionen des Jetspeed-Portals von Bedeutung.

Abbildung 20 Reactor-Design-Pattern



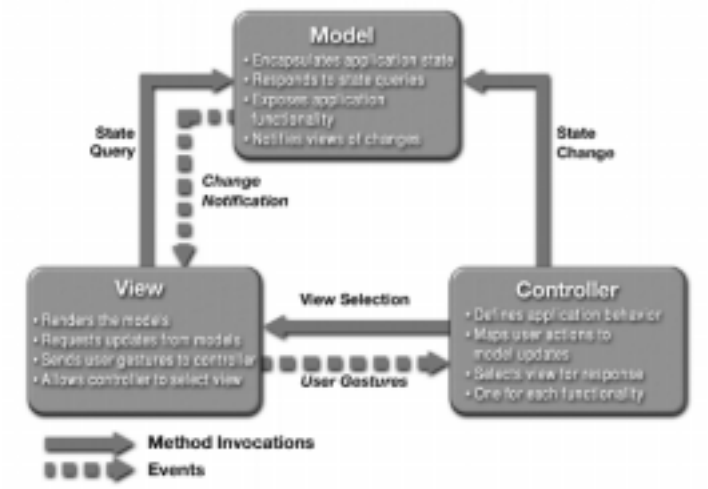
Turbine

Turbine [Turb01] ist ein auf Servlets basierendes Gerüst zur Entwicklung von Web-Applikationen. Es stellt dem Benutzer wiederverwendbare Klassen zur Verfügung. Dieser kann so seine eigenen Anwendungen um Basisfunktionen wie Datenbankanbindung, Mechanismen zur Zwischenspeichertung (Cache), Verwaltung von Sitzungen (Sessions) und vieles mehr erweitern. Da Turbine eine der Hauptsäulen von Jetspeed darstellt, muss man zur eigenen Weiterentwicklung von Jetspeed ein eingehendes Verständnis für Turbine mitbringen. Turbine legt das Fundament für die Daten- und Anwendungsintegration und wird neben der Authentifizierung der Benutzer auch für das Seitenlayout genutzt. Turbine ermöglicht die Integration von Template-Engines wie Velocity oder Java Server Pages (JSP). Ein Template liefert eine vordefinierte Struktur in die Inhalte an entsprechenden Stellen eingefügt werden können. Turbine generiert so genannte „Screens“ in denen die vorgegebenen Templates ausgewertet werden. Dies unterstützt die Entwicklung nach dem *Model-View-Controller* (MVC) [MVC01] Design-Pattern. Web-Designer können sich bei der Entwicklung eines Projekts auf ihre eigentlichen Design-Aufgaben konzentrieren während die Programmierer eine entsprechende Umgebung schaffen können.

Beim MVC Design-Pattern unterscheidet man zwischen *Model* (Modell/Daten), *View* (Sicht) und *Controller* (Steuerung). Das Modell repräsentiert die Anwendungsdaten und die Regeln, wie die Daten genutzt werden können. Eine Sicht beschreibt den Inhalt oder einen Ausschnitt eines Modells. Die Sicht sorgt ebenfalls für eine bezüglich der Daten konsistente Ausgabe. Ändern sich die Daten im Modell, so ändert sich auch die entsprechende Ausgabe. Die Steuerung dient als Vermittler zwischen den Benutzern und dem Modell. Benutzereingaben werden in durch das Modell auszuführende Aktionen umgesetzt. Weiterhin legt die Steuerung die Sicht auf das Modell fest. Das MVC-Pattern wird in Abbildung 21 gezeigt. Ein nach dem MVC-Pattern erstelltes Sytem könnte z.B. XML als Modell, XSLT als Definition für die Sicht und ein Java-Servlet zur Steuerung haben.

Das MVC-Design-Pattern ist bezüglich der Trennung von Daten, Logik und Präsentation abstrakter als das bei Cocoon vorgestellte Konzept. Weiterhin finden sich im MVC Informationen über die Beziehungen zwischen den Komponenten Modell, Sicht und Steuerung.

Abbildung 21 Das MVC Design-Pattern



Village

Village [Vil01] ist ein Java API welches auf dem *Java DataBase Connectivity* (JDBC) API aufbaut. Es dient zur vereinfachten Interaktion mit von JDBC unterstützten relationalen Datenbanken. Die Ergebnismengen des Cursor-basierten JDBC Zugriffs einer Anfrage wird zur Auswertung und Bearbeitung von Village im Speicher materialisiert. Diese Funktionalität ist aber mittlerweile auch in JDBC2 vorhanden.

Element Construction Set

Das Element Construction Set (ECS) [ECS01] ist ebenfalls ein Java API zur Generierung verschiedener Markup Language Tags. Derzeit wird HTML und XML als Ausgabeformat unterstützt. Wie die Erzeugung eines HTML Fragments mit ECS aussehen kann zeigt Beispiel 2.

Beispiel 2

```
Nur Java:
out.println("<FONT SIZE=\"+1\" FACE=\"Times\" COLOR=\"#FFFFFF">);
out.println("The big dog &amp; the little cat chased each other.");
out.println("</FONT>");

Java + ECS:
addElement(new Font().setSize("+1")
    .setColor(HtmlColor.WHITE)
    .setFace("Times")
.addElement("The big dog & the little cat chased each other.")));
```

Velocity

Velocity [Vel01] ist eine Java-basierte Template-Engine. Ähnlich wie ECS unterstützt Velocity die Erstellung von HTML oder XML Templates in Java. In der Aktuellen Jetspeed-Entwicklung ist Velocity die bevorzugte Template-Engine. In früheren Versionen von Jetspeed wurden diese Aufgaben hauptsächlich vom ECS übernommen. Velocity unterstütz ebenfalls das bereits bei Turbine vorgestellte MVC-Design-Pattern. Bei der Entwicklung wird dazu der Java-Code von den eigentlichen Web-Seiten separiert.

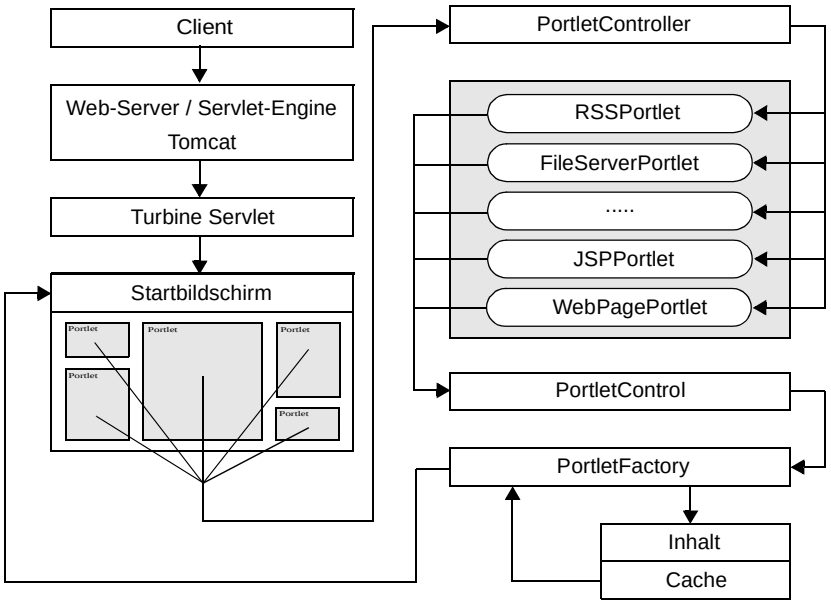
Hypersonic-SQL

Neben den bereits erwähnten Komponenten kommt zum Zweck der Speicherung von Benutzerinformationen, Berechtigungen und ähnlichen Informationen ein Datenbanksystem zum Einsatz. Jetspeed setzt dazu standardmässig *Hypersonic-SQL* [HSQL01] ein. Man kann jedoch auch andere Datenbanksysteme nutzen. Die entsprechende Konfiguration ist innerhalb der Turbine-Engine vorzunehmen, da diese für die Datenbankanbindung verantwortlich ist.

6.2 Informationsfluss in Jetspeed

Um eine detailiertere Architekturübersicht erstellen zu können, soll zuerst der in Abbildung 22 gezeigte Informationsfluss innerhalb des Jetspeed-Portals betrachtet werden. Hierzu wird weiterhin auf die Rolle von Portlets in Jetspeed eingegangen.

Abbildung 22 Informationsfluss in Jetspeed



6.2.1 Portlets

Die Jetspeed-Entwickler sehen in Portlets und deren Personalisierungsmöglichkeiten einen standardisierten Weg Informationen zu beziehen und deren Darstellung zu kontrollieren. Man kann dem Benutzer so gezielt und fein unterteilt Informationen und Anwendungen anbieten. Portlets nutzen die vom Jetspeed-Portal zur Verfügung gestellte Infrastruktur und lassen sich individuell an die Vorlieben des Nutzers anpassen. So kann der Nutzer beispielsweise die Hintergrundfarbe, Symbole, die Position oder die Menge der im Portlet enthaltenen Informationen beeinflussen. In Jetspeed lassen sich die Portlets auch nach verschiedenen Gesichtspunkten anordnen. Jetspeed besitzt dafür einen sogenannten Customizer. Wie dies im Einzelnen funktioniert, zeigt die Betrachtung des Informationsflusses durch das Portal.

Anforderungen vom Client (HTTP oder WAP) werden vom Web-Server entgegengenommen. Dieser veranlasst nun in der Servlet-Engine die Ausführung des Turbine Servlets, welches für den Aufbau der Portalseite verantwortlich ist. Die Hauptelemente sind dabei der *PortletController*, die *PortletControl* und die *PortletFactory* und die eigentlichen *Portlets*. Der PortletController organisiert die Positionierung der Portlets innerhalb der Portalseite. Der Benutzer hat die Auswahl zwischen verschiedenen Anordnungsmöglichkeiten. So kann man z. B. eine 3 Spalten Anordnung mit einer Bildschirmaufteilung von 25%/50%/25% wählen oder eine einfache tabellarische Anordnungsform. Die PortletControl erstellt im nächsten Schritt den Rahmen um die entsprechend positionierten Portlets und bestimmt deren Aussehen. In früheren Versionen wurde zur Gestaltung hauptsächlich das ECS eingesetzt. Die Tendenz in der untersuchten Version zeigt einen vermehrten Einsatz von Velocity. Dies geht auch aus Artikeln der Jetspeed Mailing-Listen hervor. Die PortletFactory instanziiert und initialisiert letztendlich das Portlet, holt den darzustellenden Inhalt und legt ihn zusätzlich in einem Zwischenspeicher (Cache) ab. Ist das Caching nicht gewünscht, lässt sich dieses auch individuell für jedes Portlet abschalten.

Jetspeed beinhaltet fünf Standard-Portlets [Jet01b]: das *RSSPortlet*, *FileServerPortlet*, *XSLPortlet*, *JSPPortlet*, sowie das *WebPagePortlet*. Alle diese Portlets bauen auf einem *AbstractPortlet* auf. Dieses sorgt für eine standardmässige Integration der darauf aufbauenden Portlets in das Portal, kann aber bei Bedarf, z.B. in Folge eigener grundlegender Änderungen am Portal, durch eine selbstentwickelte Version ersetzt werden. Das *RSSPortlet* gibt die aus RSS Datenquellen erhaltenen Informationen wieder. RSS steht für *RDF Site Summary* [RSS01] und ist ein von Netscape entwickeltes auf XML basierendes Format. Netscape setzt RSS zum betreiben von Kanälen innerhalb des Netscape Netcenters ein. Ziel von RSS ist die Verbreitung von geänderten Seiteninhalten. Der Datenverkehr kann so auf das Nötigste eingeschränkt werden. Verändert sich der Inhalt werden alle registrierten Seiten über die Änderung informiert. Analog zum RSSPortlet gibt das *FileServerPortlet* Inhalte aus einer angegebenen URL wieder, welche eine Datei auf der lokalen Festplatte, eine Anforderung via HTTP oder FTP sein kann. Das *XSLPortlet* gibt ein mittels XSL-Stylesheets transformiertes XML Dokument aus. Entsprechend verarbeitet das *JSPPortlet* eine mit JSP erstellte Seite und gibt diese wieder. Das *WebPagePortlet* ermöglicht die Darstellung einer anderen Webseite innerhalb eines Portlets. Dazu werden entsprechende HTML Elemente aus der darzustellenden Seite entfernt um die Ausgabe innerhalb des Portals ermöglichen zu können.

Neben den fünf Standard-Portlets besitzt Jetspeed weitere zur Steuerung und Administration. Zur Steuerung dient beispielsweise das *PortletInfoPortlet*. Es ermöglicht dem Benutzer Metainformationen über die von ihm abonnierten Portlets zu erhalten. Das *PortletBrowserPortlet* gibt dem Benutzer eine Übersicht über alle zur Verfügung stehenden Portlets. Das *ApplicationsPortlet* ermöglicht die Darstellung von Portlets, welche als Anwendungen gekennzeichnet sind. Es hat aber keinen Einfluss auf deren Inhalt. Das besondere bei als Anwendungen gekennzeichneten Portlets ist, dass sich mehrere zu einem einzigen zusammenfassen lassen. Dieses wird dann in einem gesonderten Bereich, meistens jedoch als Vollbildansicht, innerhalb der Portalseite dargestellt.

Weiterhin sind auch Portlets vorhanden, welche von der eigentlichen Jetspeed-Entwicklung unabhängig sind. Ein Beispiel ist das *CocoonPortlet*. Es nutzt Cocoon zur Transformation von XML Dokumenten. Auf diese Weise kann das Jetspeed-Projekt von durch den individuellen Bedarf seiner Nutzer entstandenen Portlet-Eigenentwicklungen profitieren.

6.2.2 PSML

Bei der Realisierung von Jetspeed spielt XML eine wichtige Rolle. Eine aus XML für Jetspeed entwickelte Beschreibungssprache, die *Portal-Structure-Markup-Language* (PSML), wird unter zwei Gesichtspunkten eingesetzt, wobei die zur Verfügung stehenden Sprachelemente unterschiedlich sind.

Übersicht

Die Registrierung von Portlets innerhalb von Jetspeed findet in der sogenannten „Portlet-Registry“ statt. Es existiert ein Verzeichnis in dem diese Konfigurationen abgelegt werden müssen. Alle darin enthaltenen Registrierungsdateien werden dann ausgewertet. Wie ein Registry-Eintrag aussehen kann zeigt Beispiel 3 auf Seite 54. Abbildung 23 auf Seite 53 liefert eine UML[1]-Übersicht über die zur Verfügung stehenden Sprachelemente.

Weiterhin dient PSML zur Steuerung der Personalisierung. Jeder Benutzer hat zu diesem Zweck seine eigenen PSML-Dateien. Eine für den Portalzugriff über den Web-Browser und eine zweite für den Zugriff mit WAP-fähigen Geräten. Ist man nicht als Benutzer angemeldet, gibt es für beide Ausgabetypen systemweite Default-Dateien. Beispiel 4 auf Seite 56 veranschaulicht die Personalisierung mit PSML. Auch hier findet man mit Abbildung 24 auf Seite 55 eine UML-Übersicht über die Sprachelemente.

Jetspeed-Konfiguration mit PSML

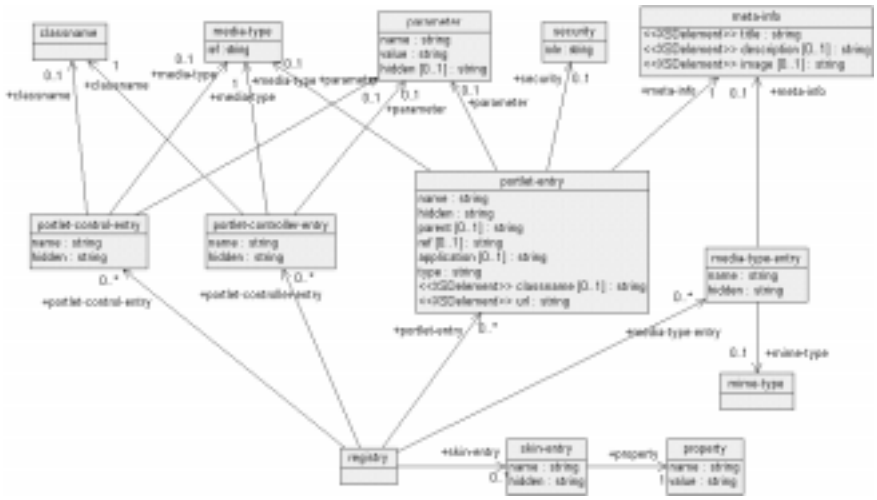
Mit Hilfe von PSML werden vorhandene Portlets innerhalb der eigentlichen Jetspeed-Anwendung registriert. Diese Informationen werden dann z.B. vom PortletBrowserPortlet genutzt um alle verfügbaren Portlets anzuzeigen. Wie ein solcher Eintrag aussehen kann zeigt Beispiel 3. Die Portlets werden in der Registry als „Portlet-Entry“ beschrieben. Man unterscheidet dabei 3 unterschiedliche Typen:

1. Unified Modeling Language (UML), Sprache zur Modellierung und Visualisierung komplexer Strukturen, (http://www.jeckle.de/uml_spec.htm)

- **Instance:** Basiseintrag und für jedes Portlet erforderlich. Ein solcher Eintrag enthält alle zur Instanziierung notwendigen Informationen (z. B. den Klassennamen).
- **Abstract:** Ein solcher Eintrag kann wegen unzureichenden Informationen nicht direkt instanziiert werden. Er dient zur Definition allgemeiner Eigenschaften, die sich z. B. auf eine Gruppe von Portlets übertragen lassen.
- **Ref:** Hiermit lassen sich auf anderen Einträgen (Instance, Abstract, Ref) basierende Registry-Informationen erstellen und erweitern. Zeigt ein Ref-Eintrag auf einen anderen Ref-Eintrag wird solange dereferenziert, bis ein Eintrag vom Typ Instance oder Abstract gefunden wird. Die im Ref-Eintrag vorhandenen Parameter können eventuell schon im Abstract- oder Instance-Eintrag vorhandenen Parameter überschreiben.

Innerhalb des **Portlet-Entry** können dann verschiedene **Parameter** festgelegt werden. Mit dem **Media-Type** Eintrag lassen sich Angaben zu den innerhalb des Portlets angezeigten Datenformaten machen. In der Regel ist dieser „html“ oder „wml“. Mit dem **Security** Eintrag kann eine Rolle angegeben werden, welche auf das Portlet zugreifen darf bzw. berechtigt ist, Veränderungen an dessen Aussehen vornehmen zu können. In der Administration des Portals lassen sich dafür jedem Benutzer individuelle Rollen zuweisen und auch eigene Rollen erstellen. Diesen Rollen können dann wiederum Berechtigungen zugeteilt werden, beispielsweise „Maximize“, um ein Portlet auf Vollbildansicht umschalten zu dürfen. Berechtigungen lassen sich ebenfalls selbst erstellen und können so auf die Bedürfnisse des Portlets zugeschnitten werden. Zum jetzigen Zeitpunkt wird auch an einer Gruppenverwaltung gearbeitet welche jedoch erst ansatzweise vorhanden ist. Im **Meta-Info** Eintrag finden sich weitere Informationen zum Portlet wie z. B. der in der Titelzeile anzuzeigende Text, oder eine informelle Beschreibung dessen, was das Portlet eigentlich macht. Neben dem Portlet-Entry finden sich in der Registrierung auch die Einträge **Portlet-Control-Entry**, **Portlet-Controller-Entry** und **Skin-Entry** welche sich ähnlich wie der Portlet-Entry zusammensetzen hier aber nicht näher betrachtet werden sollen. Änderungen an der Registrierung erfordern einen Neustart der Ausführungsumgebung, in unserem Fall also Tomcat.

Abbildung 23 UML Repräsentation des Portlet-Registry PSML-Schemas



Beispiel 3 PSML-Registry: Portlet-Entry vom Typ „instace“

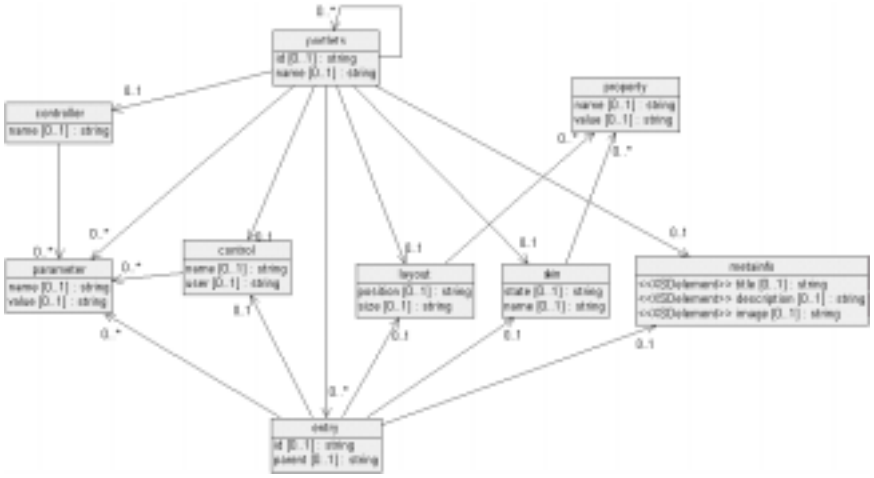
```
<registry>
...
<portlet-entry name="Demo" hidden="false" type="instance" application="false">
        <meta-info>
            <title>Demo Portlet</title>
            <description>Portlet zu Demozwecken eines Registry-Eintrags</description>
        </meta-info>
        <classname>org.apache.jetspeed.portal.portlets.DemoPortlet</classame>
        <parameter name="font" value="Arial" hidden="false"/>
        <parameter name="bgcolor" value="gray" hidden="false"/>
        <media-type ref="html"/>
</portlet-entry>
...
</registry>
```

Personalisierung mit PSML

Neben der Konfiguration wird mit PSML festgelegt, welche Portlets für bestimmte Benutzer angezeigt werden, wie sie innerhalb der Portalseite angeordnet sind und welche Darstellungseigenschaften genutzt werden sollen. Die dazu genutze PSML-Sprache besitzt 4 Hauptelementtypen:

- **Entry:** Hier werden Portlet-Elemente beschrieben die sich mit jenen aus der Portlet-Registry vergleichen lassen. Jedoch sind alle Einträge vom Typ Ref und können keine grundlegenden Eigenschaften wie z. B. den Klassennamen beschreiben. Sie bieten die Möglichkeit zusätzliche, nicht in der Registry enthaltene Eigenschaften wie Skins oder andere Layouteinstellungen hinzuzufügen.
- **Portlets:** Ähnlich dem Abstract-Eintrag in der Registry ermöglicht dieser Eintrag die Beschreibung einer Gruppe von Portlets (PortletSet). So lassen sich z. B. gemeinsame Layout- und Präsentationsoptionen bestimmen.
- **Controller:** Dieser Eintrag weist einem PortletSet einen PortletController zu. Dieser kümmert sich dann, wie bereits beschrieben, um die Positionierung der Portlets innerhalb der Portalseite. Man kann dabei z. B. den Bildschirm in verschieden große Spalten aufteilen.
- **Control:** Hiermit wird eine ensprechende PortletControl zugewiesen welche dann für die grafische Repräsentation wie Rahmen und Überschrift des Portlets sorgt.

Abbildung 24 UML Repräsentation des zur Personalisierung genutzen PSML-Schemas

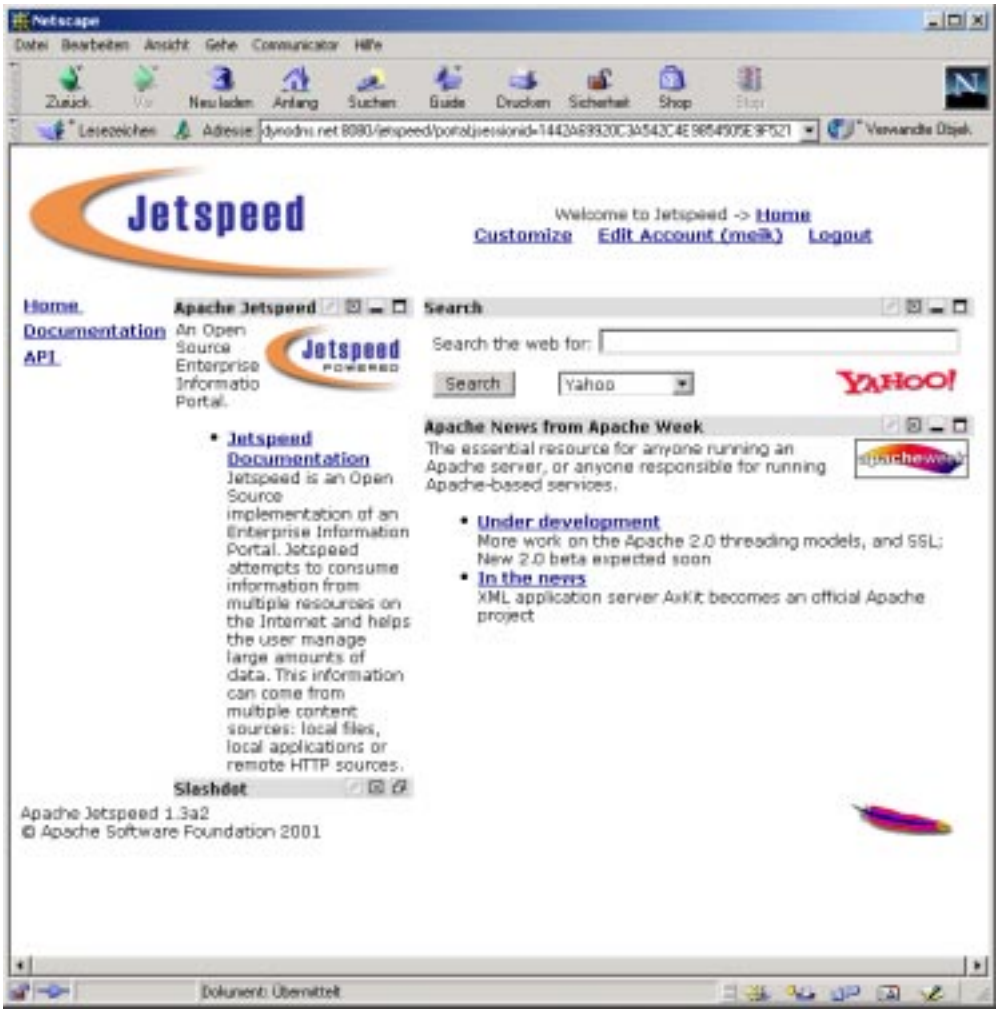


Der Benutzer kann, soweit dies der Autor des Portlets zulässt, mit Hilfe des in Jetspeed enthaltenen Customizers die in der Portlet-Registry vorgegebenen Einstellungen für das entsprechende Portlet erweitern und verändern. Der Customizer ermöglicht dem Benutzer, neben der Veränderung der Darstellung, ebenfalls die Anordung seiner Portlets zu bestimmen. Alle diese Benutzerspezifischen Einstellungen werden dann in Form einer PSML-Datei abgelegt. Diese lässt sich im Gegensatz zur Registrierung der Portlets auch ohne Neustart der Ausführungsumgebung verändern. Beispiel 4 zeigt eine durch den Customizer erstellte PSML Datei für einen Benutzer. In dieser ist als PortletController „TwoColumns“ festgelegt. Die einzelnen Portlets bekommen ihre Position dabei durch die Layout-Property „row“ und „column“ zugewiesen. Das Farbschema für die Darstellung ist durch den Skin „orange-grey“ angegeben. Weiterhin findet man Parameter wie „itemdisplayed“ oder „_display“ welche festlegen, wieviele Einträge innerhalb eines Portlets angezeigt werden sollen bzw. den aktuellen Darstellungsmodus wie beispielsweise „minimized“. Wie sich diese Einstellungen auf die entsprechende Portalseite auswirken, zeigt Abbildung 25. Wie bei der Registrierung soll auch hier mit Abbildung 24 auf Seite 55 eine UML-Übersicht über das PSML-Schema zur Personalisierung gegeben werden.

Beispiel 4 PSML Site Markup

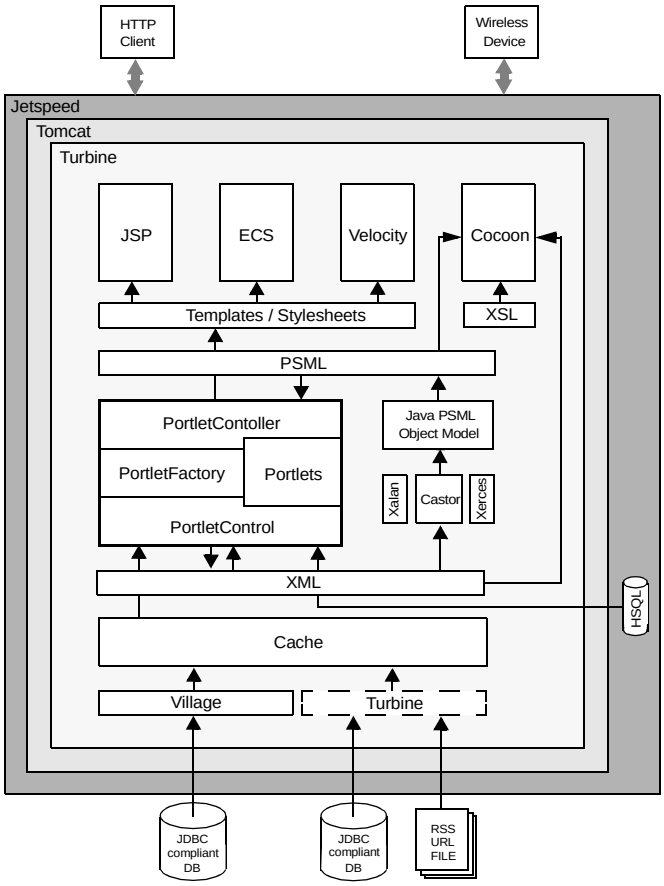
```
<?xml version="1.0" encoding="UTF-8"?>
<portlets xmlns="http://www.apache.org/2000/02/CVS">
    <controller name="TwoColumns"/>
    <skin name="orange-grey"/>
    <entry parent="Jetspeed">
        <layout>
            <property name="column" value="0"/>
            <property name="row" value="0"/>
        </layout>
    </entry>
    <entry parent="Search">
        <layout>
            <property name="column" value="1"/>
            <property name="row" value="0"/>
        </layout>
    </entry>
    <entry parent="http://www.slashdot.org/slashdot.rdf">
        <parameter name="itemdisplayed" value="4"/>
        <parameter name="_display" value="minimized"/>
        <layout>
            <property name="column" value="0"/>
            <property name="row" value="1"/>
        </layout>
    </entry>
    <entry parent="http://www.apacheweek.com/issues/apacheweek-headlines.xml">
        <parameter name="itemdisplayed" value="2"/>
        <layout>
            <property name="column" value="1"/>
            <property name="row" value="1"/>
        </layout>
    </entry>
    <portlets>
        <metainfo>
            <title>Test-Pane</title>
        </metainfo>
    </portlets>
</portlets>
```

Abbildung 25 Screenshot einer mit Jetspeed personalisierten Seite



Die Betrachtungen dieses Kapitels leiten abschliessend zu einer detailierten Jetspeed-Architektur welche in Abbildung 26 auf Seite 58 dargestellt ist. Analog zur allgemeinen Portalarchitektur in Abbildung 19 auf Seite 43 läßt sich im unteren Bereich der Jetspeed-Architektur mit Village und Turbine die Datenintegration erkennen. Die Portal-Logik wird durch den PortletController, die PortletFactory und die PortletControl repräsentiert und in Kombination mit PSML ergeben sich die Möglichkeiten zur Personalisierung. Die Portlets stellen die Dienste eines horizontalen Portals dar. Die Erstellung verschiedener Ausgabeformate (Customization / MCD) befindet sich im oberen Bereich der Architektur. Durch die zur Verfügung stehenden Template-Engines wie z. B. Velocity lassen sich dafür die Ausgaben an einen Web-Browser (HTTP-Client) oder ein Wireless-Device (z. B. WAP-Handy) anpassen. Zum jetzigen Stand der Jetspeed-Entwicklung existieren noch keine speziellen Komponenten zur Anwendungsintegration. Wie sich trotzdem Anwendungen integrieren lassen, wird Kapitel 11 zeigen.

Abbildung 26 Jetspeed Architektur im Detail



Im folgenden Kapitel soll das zu implementierende Beispielsystem, ein Web-basiertes Handelssystem für Warentermingeschäfte, vorgestellt werden. Die daraus hervorgehenden Anforderungen sollen in späteren Kapiteln prototypisch mit Jetspeed realisiert werden.

Kapitel 7 Anforderunganalyse für ein Web-basiertes Handelssystem für Warentermingeschäfte

In diesem Kapitel sollen die Anforderungen an ein Web-basiertes Handelssystem für Warentermingeschäfte vorgestellt und analysiert werden. Ein Teil dieser Anforderungen soll dann in Kapitel 10 und Kapitel 11 mit dem im vorigen Kapitel vorgestellten Jetspeed-Portal umgesetzt werden.

7.1 Motivation und Hintergrund

Mit Hilfe einer neuartigen Internet-Handelsplattform sollen Unternehmen ihre Produkte (Waren, Dienstleistungen) terminbezogen und preisflexibel vertreiben können. In Anlehnung an Warentermingeschäfte sollen frühzeitig Kontrakte unter Angabe von Anzahl, Termin, Erfüllungsort, etc. abgeschlossen werden können. Dadurch wird auf Anbieterseite die Kapazitätsplanung erleichtert, auf Nachfragerseite der Kaufpreis flexibilisiert.

Voraussetzung ist jedoch, dass der Käufer das Produkt innerhalb ausreichend großer (Termin-) Toleranzen beschaffen kann. In Abhängigkeit von den beim Verkäufer-Unternehmen vorhandenen Ressourcen wird dabei versucht die Nachfrage über den Angebotspreis zu steuern. Je größer die Restkapazitäten sind, desto preisgünstiger wird das Produkt angeboten, je geringer die Kapazitäten, desto teurer wird das Produkt. Um einen dynamischen Handel zu erreichen, gibt der Verkäufer Order in kleiner Stückelung in das Handelssystem ein. Diese Order treffen auf Kauforder von Kunden. In einem automatisierten Preisfeststellungsverfahren werden Kauf- und Verkaufsorder zusammengeführt (gematcht).

Ein Ziel einer solchen Handelsplattform liegt in einer potentiellen Kostensenkung innerhalb der gesamten Wertschöpfungskette. Ein weiterer Motivationsfaktor liegt in der Möglichkeit der Marktteilnehmer, vielfältige Informationen über die aktuelle Marktsituation, Angebots- und Nachfrageentwicklungen sowie über die Lieferanten und Kunden zu gewinnen.

Auf diesem Hintergrund soll eine Umgebung geschaffen werden, welche diese Anwendung optimal unterstützt. Aufgrund der Zielsetzung und den in den vorigen Kapiteln gewonnenen Erkenntnissen über Portale erscheint hier der Einsatz eines Portals als sinnvoll [Weid01].

7.2 Prozesse innerhalb des Handelssystems

Zuerst sollen die für die Realisierung eines solchen Systems notwendigen Prozesse zusammengestellt und analysiert werden. Dabei werden im Folgenden die drei Bereiche *Registrierung, Produkt- und Marktinformation* sowie *Login, persönliche Dienste und Orderverwaltung* unterschieden. Eine Übersicht der Prozesse liefern Abbildung 27 und Abbildung 28.

7.2.1 Registrierung

Da das Web-basierte Handelssystem finanzielle Transaktionen voraussetzt und personalisierbare Inhalte enthalten soll, ist eine Registrierung der Benutzer erforderlich. Nach Anzeige der Startseite des Systems soll es nach der Einwilligung in die allgemeinen Geschäftsbedingungen möglich sein, eine Benutzerkennung zu erhalten. Nach Eingabe der Stammdaten und Vergabe einer eindeutigen Benutzer-Kennung inklusive des gewünschten Passworts soll eine Bestätigung via E-Mail an den Benutzer versendet werden. Weiterhin sollen die Daten des Benutzers in eine Datenbank übernommen werden (siehe Abbildung 27 oben).

7.2.2 Produkt- und Marktinformationen

Innerhalb des Handelssystems soll eine Produktsuche möglich sein. Nach der Eingabe der Suchkriterien soll eine Produkt- und Auktionsübersicht generiert werden. Weiterhin sollen aktuelle Marktinformationen zu den Produkten gegeben werden. Dazu gehört beispielsweise der aktuelle Preis. Auch die historische Entwicklung der Marktinformationen soll bereitgestellt werden. Dies dient z. B. der Abschätzung von Nutzen und Risiko der möglicherweise hoch spekulativen Warentermingeschäfte. Neben der produktspezifischen Suche soll auch eine beliebige Suche bei verschiedenen Suchmaschinen möglich sein (siehe Abbildung 27 unten).

7.2.3 Login, persönliche Dienste und Orderverwaltung

Nachdem der Benutzer seine Benutzerkennung erhalten hat, sollen ihm neben allgemeinen Informationen vor allem seine Daten innerhalb des Handelssystems präsentiert werden. Dazu gehört beispielsweise das Anzeigen und Verändern der Stammdaten, eine Depoteinsicht sowie eine Übersicht der laufenden Auktionen (Orderbuch). Weiterhin soll die Abgabe, Änderung und Streichung einer Order möglich sein (siehe Abbildung 28).

7.2.4 Sonstige Anforderungen

Neben den bis jetzt aufgeführten Anforderungen soll das System auch eine Notifikations-Funktion aufweisen. Dazu gehört die Benachrichtigung bei Veränderungen am Orderstatus oder bei anderen die Auktion betreffenden Nachrichten. Wie in Kapitel 4 gesehen, kann dies beispielsweise durch den Versand einer entsprechenden E-Mail realisiert werden. Denkbar wäre weiterhin eine direkte Anbindung an ein bestehendes PPS-System[1] des Anbieters oder auch beim Interessenten selbst. Zur einfacheren Administration wäre ein Redaktions-Frontend als Hilfsmittel zum

1. System zu **P**roduktions**p**lanung und -**s**teuerung (PPS)

Content-Management sinnvoll. Eine Eingriffsmöglichkeit in das System zu Prüfungs- und Aufsichtszwecken durch ein Überwachungsgremium ist ebenfalls wichtig.

Abbildung 27 Prozesse innerhalb des Handelssystems (1. Teil)

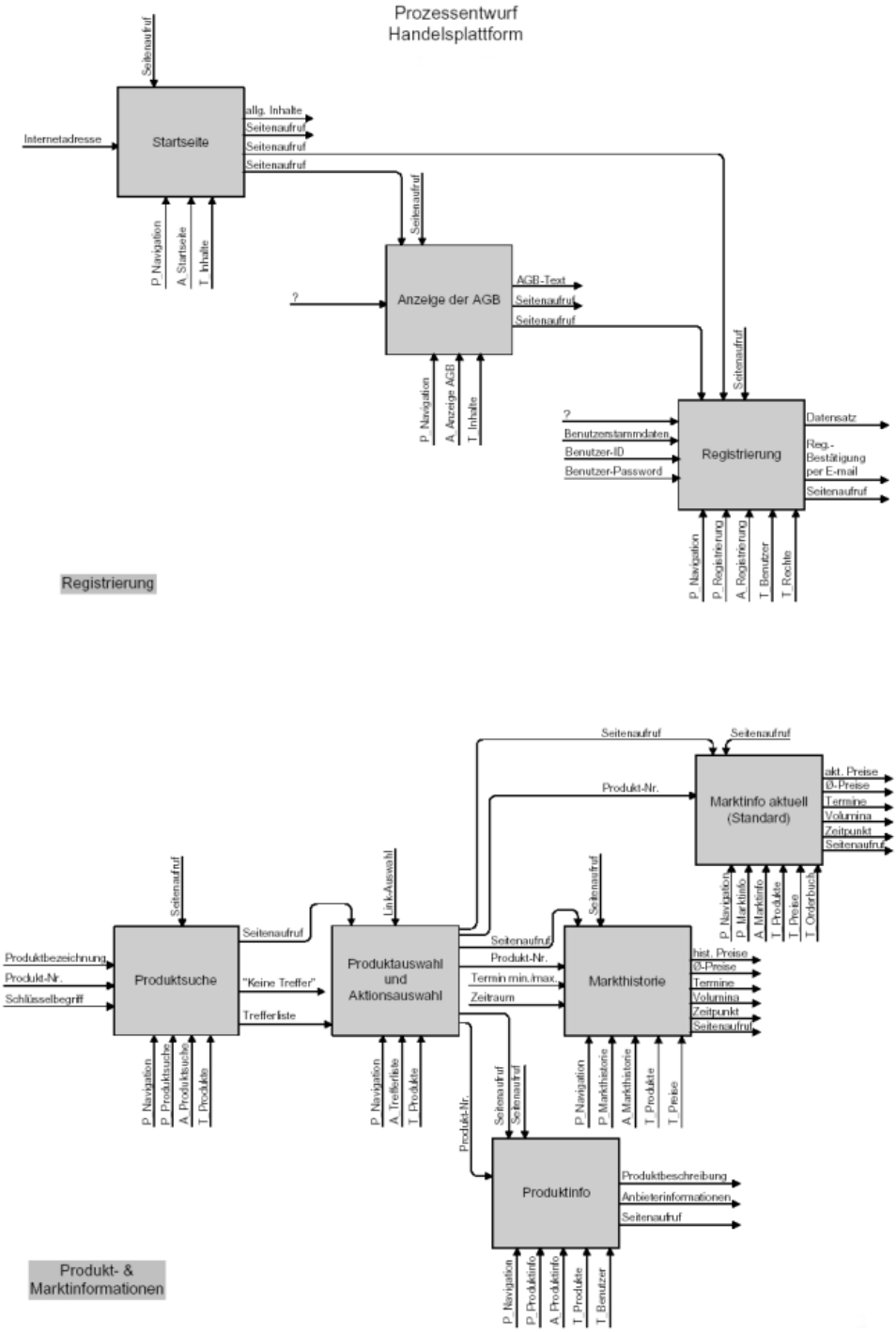
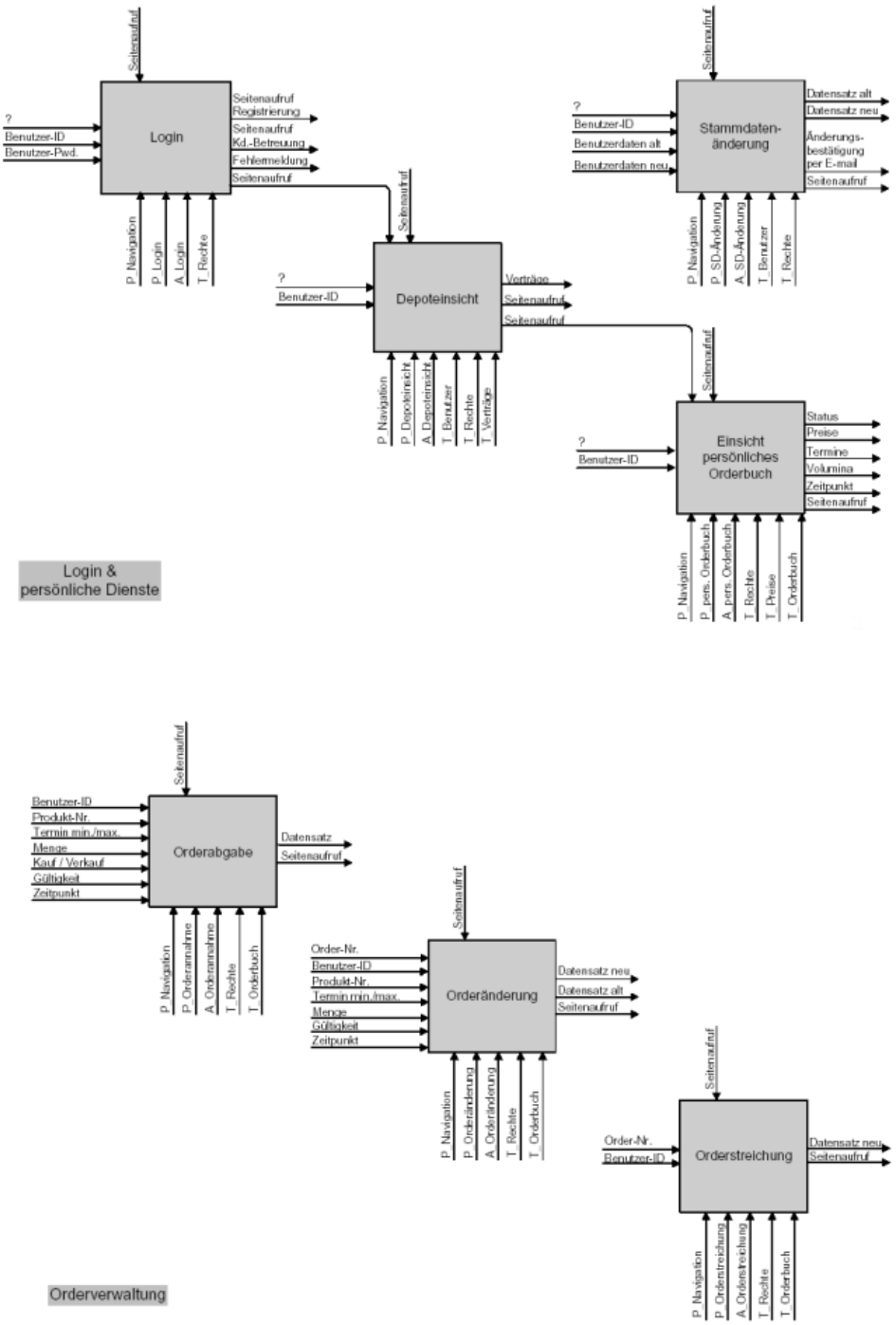


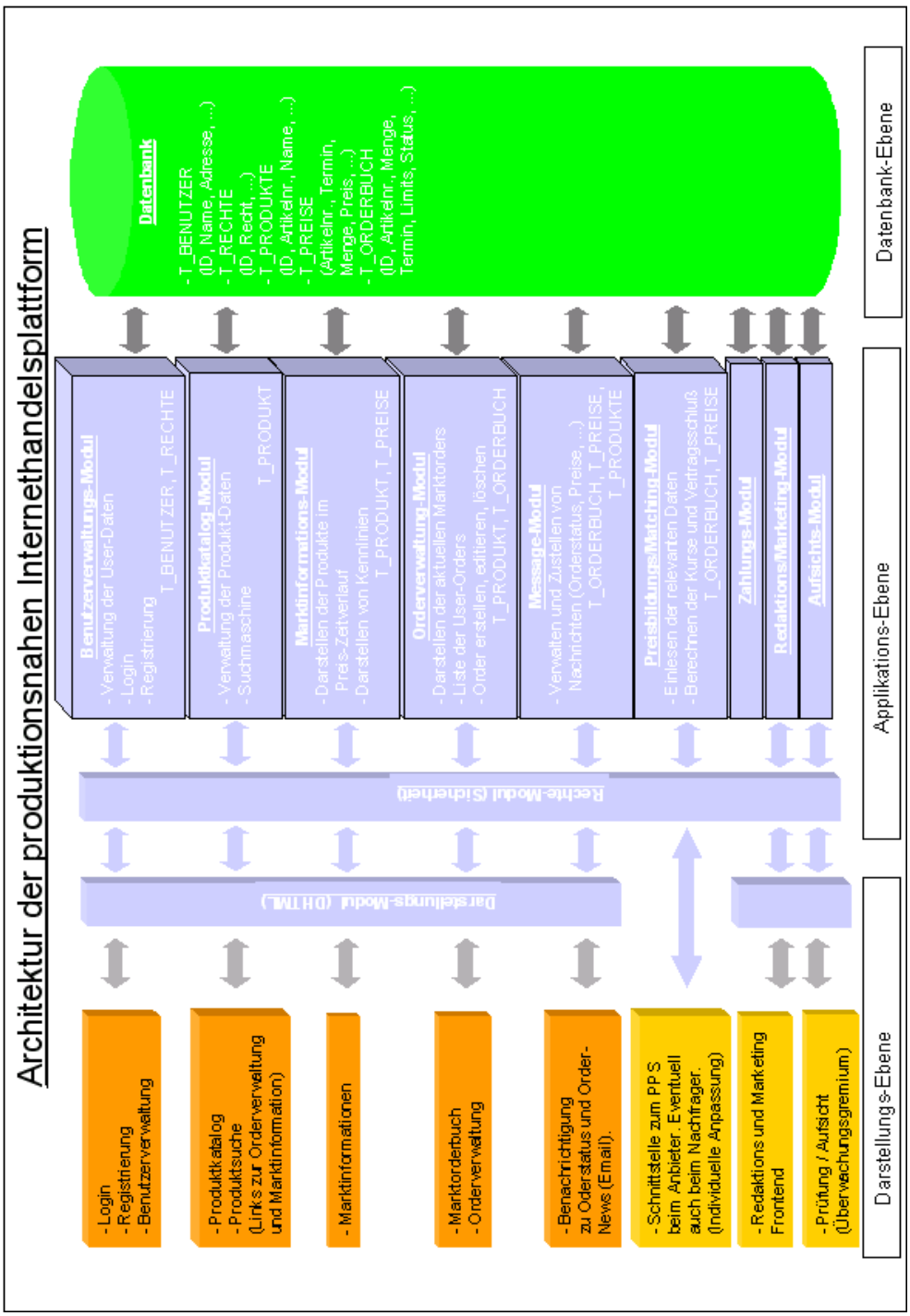
Abbildung 28 Prozesse innerhalb des Handelssystems (2. Teil)



7.3 Architektur des Handelssystems

Aus den Anforderungen und der Prozessübersicht lässt sich eine Modularisierte Architektur der Handelsplattform erstellen. Eine grafische Repräsentation zeigt Abbildung 29.

Abbildung 29



Die Architektur unterscheidet die *Darstellungs-*, *Applikations-* und *Datenbank-Ebene*. Diese Darstellung enspricht dem im Abschnitt “Das n - Schichtenmodell” auf Seite 13 vorgestellten 3-Schichten-Modell. Die **Darstellungs-Ebene** zeigt noch einmal alle Anforderungen an das System auf. Die Applikations-Ebene enthält eine modularisierte Struktur der zur Realisierung der Anforderungen notwendigen Komponenten. Das System wird in der **Applikations-Ebene** in zehn Module aufgeteilt:

- **Rechte-Modul:** Zur Realisierung von Sicherheitsaspekten (Authorosierung etc.).
- **Benutzerverwaltungsmodul:** Zur Registrierung, Verwaltung der Benutzerdaten und zum Login.
- **Produktkatalog-Modul:** Zur Verwaltung der Produktdaten und zur Produktsuche
- **Marktinformations-Modul:** Zur Darstellung der Produkte im Preis/Zeit-Verlauf.
- **Orderverwaltungs-Modul:** Zum Erstellen, Editieren, Löschen und Auflisten von Order.
- **Message-Modul:** Zum Verwalten und Zustellen von Nachrichten.
- **Preisbildungs/Matching-Modul:** Zum Einlesen der relevanten Daten und anschliessender Berechnung der aktuellen Kurse.
- **Zahlungs-Modul:** Zur Abwicklung erforderlicher Zahlungen beispielsweise mit einer Kreditkarte.
- **Redaktions/Marketing-Modul:** Zur Bearbeitung der statischen Inhalte.
- **Aufsichts-Modul:** Zur Prüfung und Aufsicht im System durch ein Überwachungsgremium.

Naheliegend im Bezug auf die Realisierung wäre die Implementierung dieser Module als Portlets. Eine Ausnahme bildet das Preisbildungs/Matching-Modul, da die Preisfeststellung im Hintergrund stattfindet. Hier wäre eine Realisierung als eigenständiger, im Hintergrund auf dem System laufenden Prozess denkbar. Dieser muss dann in festzulegenden Zeitabständen die Berechnungen zur Preisbildung der einzelnen Produkte durchführen.

Die **Datenbank-Ebene** enthält die Datenbank zur Speicherung aller relevanten Daten wie Benutzer- oder Orderinformationen sowie die entsprechenden Mechanismen zu deren Anbindung an die Applikations-Ebene.

Neben der Datenintegration, die beispielsweise beim Marktinformations-Modul durchgeführt werden muss, soll für dieses System auch eine Anwendungsintegration durchgeführt werden. Das Zahlungs-Modul stellt eine solche Anwendungsintegration dar.

Wie in Kapitel 6 bereits erwähnt, besitzt das zur Realisierung vorgesehene Jetspeed-Portal keine speziellen Mechanismen zur Anwendungsintegration. Daher soll in Kapitel 9 mit SOAP eine Möglichkeit zur Anwendungsintegration aufgezeigt werden. Zuerst beschreibt das folgende Kapitel die Bedeutung der Anwendungsintegration im Allgemeinen und zeigt Vorgehensweisen und Integrationsmechanismen auf.

Zusammenfassend lässt sich zu diesem Zeitpunkt sagen, das Jetspeed prinzipiell zur Realisierung aller oben genannten Anforderungen in der Lage ist. In den späteren Kapiteln wird festgelegt, welche der Funktionalitäten beispielhaft mit Jetspeed implementiert werden sollen.

Kapitel 8 Anwendungsintegration

Neben der Datenintegration und der Personalisierung spielt ferner bei Portalen die Anwendungsintegration eine entscheidende Rolle. Die Anforderungen an die Anwendungsintegration steigen sogar vor dem Hintergrund folgender Entwicklungen noch weiter an [PWC00]:

- E-Business basiert auf integrierten Prozessen die Datenaustausch zwischen Unternehmen gewährleisten müssen. Dabei wird ein effizienter Datenzugriff in Echtzeit immer wichtiger, d. h. der Austausch von Daten muss innerhalb kürzester Zeit erfolgen können (im Gegensatz zur Kommunikation via Briefpost oder ähnlichem).
- Globalisierung und weltweite Vernetzung erfordern 24-Stunden-Verfügbarkeit der beteiligten Systeme und Daten.
- Ständige Veränderungen der Geschäftsprozeße erforderen eine Integration in neue oder bestehende Anwendungen.
- Zeitnahe Umsetzung funktionaler Anforderungen durch immer kürzerwerdende Realisierungszyklen.
- Wachsende Anzahl zu integrierender „Black-Box“-Software (Legacy-Systeme).
- Bedarf einer effizienten und raschen Integration neuer technologischer Trends.

Techniken und Produkte zur Anwendungsintegration sollen für diese Herausforderung einen möglichst einfachen Lösungsansatz zur Komplexitätsreduzierung liefern.

8.1 Allgemeine Aspekte der Anwendungsintegration

Eine Anwendungsintegration (AI) wird häufig bei der Erweiterung, Migration oder Erneuerung einer bestehenden Systemlandschaft erforderlich. In unserem Fall also bei der Erweiterung der Systemlandschaft um ein Portal. Die Ziele einer Anwendungsintegration lassen sich dabei durch die Beschreibung der Integrationsbreite und Integrationstiefe definieren [PWC00].

Integrationsbreite

Unter Integrationsbreite versteht man den Umfang, in dem Anwendungen integriert werden. Integriert man z. B. eine bestehende Adressdatenbank mit einer reinen Suchfunktionalität, ist die Integrationsbreite vergleichsweise gering. Kommt dazu noch eine Möglichkeit die Adressen bearbeiten zu können oder eine Postleitzahlensuche zur Vereinfachung der Eingaben wächst die Integrationsbreite. Mit zunehmender Integrationsbreite steigt also die Komplexität der Integrationsaufgabe.

Integrationstiefe

Die Integrationstiefe entspricht dem Grad der semantischen Integration und lässt sich in drei Ebenen unterteilen. Bei der *Integration auf Datenebene* wird durch den Einsatz von Datentransferprotokollen für die Übertragung von Daten von einem System zum anderen gesorgt. Dies stellt zwar sicher das Daten von System A zum System B übertragen werden, jedoch ist eine korrekte Interpretation durch System B damit noch nicht garantiert. Dazu wird die *Integration auf Objektebene* notwendig. Man definiert dazu Objekte. Sie bestehen aus Daten und einer standardisierten Schnittstelle, den sogenannten Methoden, mit denen auf die Daten zugegriffen werden kann. Die *Integration auf Prozessebene* umfasst letztlich die anspruchvollste Variante semantischer Integrationsmaßnahmen. Sie beschreibt die Unterstützung von Prozessen, in deren Verlauf ggf. verschiedene Objekte be- und verarbeitet werden. Sind also die Daten übermittelt und im richtigen Format werden sie auf dieser Ebene „verstanden“ und können in einen gemeinsamen Kontext gebracht werden.

8.1.1 Wege zur Anwendungsintegration

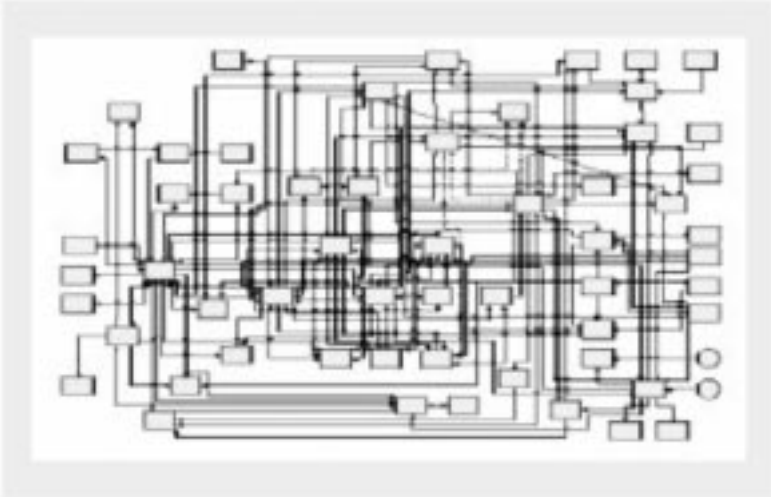
Die Integration kann über unterschiedliche Wege realisiert werden. Hier sollen im Folgenden das Vorgehen bei *Point-to-Point Verbindungen*, die *ERP basierte Anwendungsintegration* und der *Middleware-orientierte Integrationsansatz* vorgestellt werden. In der Realität finden sich Aufgrund der oft in mehreren Schritten durchgeführten Anwendungsintegration häufig Mischformen.

Eine Übersicht über die einzelen Integrationsebenen liefert Abbildung 32 auf Seite 69. Dort wird die wachsende Komplexität der Integration auf den unterschiedlichen Ebenen durch den dazu parallel steigenden „Wert der Integration“ ausgedrückt.

Point-to-Point Verbindungen

Als Point-to-Point Verbindung bezeichnet man eine Verbindung zwischen zwei gleichberechtigten Anwendungen bzw. Systemen. Bei der Integration mehrerer Anwendungen oder Systeme besteht also jeweils eine eigenständige Verbindung von jedem System zu jedem System. Diese Architektur ist häufig das Resultat gewachsener Strukturen durch regelmäßige Ergänzung der Systemlandschaft durch weitere Anwendungen und Applikationen. Abbildung 30 zeigt ein Beispiel für eine solche Netzwerkstruktur.

Abbildung 30 Beispiel einer Point-to-Point dominierten Netzwerkstruktur



Dieser Ansatz hat jedoch einige gravierende Nachteile:

- Hoher Betriebsaufwand für die Wartung der Schnittstellen.
- Hoher Aufwand bei der Integration zusätzlicher Anwendungen.
- Suboptimale Ressoursennutzung.
- Inhomogene und inkonsistente Daten.
- Zeitliche Verzögerungen bei der Datenbereitstellung.

Daher kann eine Point-to-Point-Verbindung den Anforderungen an die Anwendungsintegration, also Geschwindigkeit der Informationsflüsse in Verbindung mit der Bereitstellung von Daten in Echtzeit sowie Flexibilität bei der Integration neuer Anwendungen, nur in den seltensten Fällen genügen.

ERP basierte Anwendungsintegration

In den letzten Jahren haben viele Unternehmen mit der Einführung von ERP-Systemen wie SAP R/3 große Fortschritte bei der Anwendungsintegration gemacht. Solche Systeme besitzen spezielle Schnittstellen für die Integration von in Unternehmen häufig genutzten Anwendungen. Jedoch muss davon ausgegangen werden, das auch zukünftig selten mehr als 40% der benötigten Anwendungunterstützung durch ERP-Systeme abgedeckt werden kann[1]. Branchen- oder unternehmensspezifische Anforderungen werden also auch weiterhin durch spezielle Lösungen ergänzt werden. Dies wird dadurch erhärtet, das man in der Lage bleiben möchte, schnell auf neuartige branchenübergreifende Anforderungen reagieren zu können. ERP-Systeme werden bei vielen Unternehmen die Rolle einer Kernanwendung übernehmen wobei ergänzend Einzelanwendungen für bestimmte Zwecke erhalten bleiben. Der ERP-basierte Integrationsansatz ist

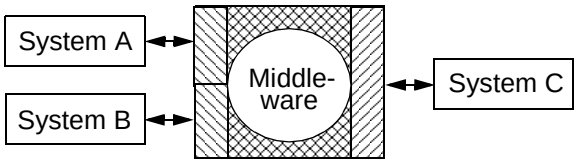
1. Vgl. Conspectus EAI-Report 1999 (http://www.conspectus.com)

damit auch nur dann geeignet wenn die Satellitenanwendungen keinen direkten Datenaustausch erfordern. Ansonsten erhält man die gleichen Einschränkungen wie bei Point-to-Point Anwendungen.

Middleware-orientierte Integrationsansatz

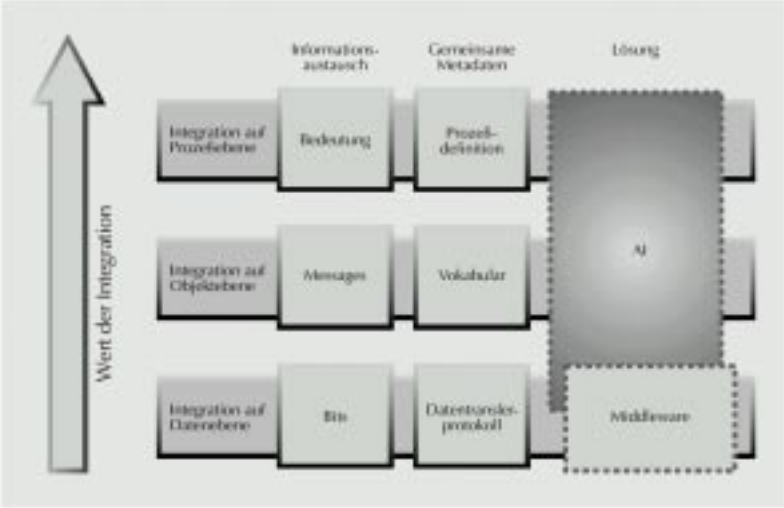
Bei der Middleware soll die Integration verschiedener Systeme in standardisierter Form ermöglicht werden. Middleware-Produkte werden zwischen zwei oder mehrere Systeme geschaltet und ermöglichen es, Systemen und Anwendungen verschiedener Hersteller auch plattformunabhängig zu kommunizieren und Daten auszutauschen. Vielfach findet man Middleware in der oben beschriebenen Point-to-Point Umgebung, die sogenannte Message-Oriented Middleware. Middleware-Produkte erzielen ihre Leistungen im Wesentlichen bei der Integration auf Datenebene. Abbildung 31 zeigt die Middleware-basierte Kommunikation zwischen den Systemen A, B und C wobei System C durch die Middleware an A und B angepasst wird.

Abbildung 31 Middleware-basierte Kommunikation



Eine weitere Form der Middleware ausserhalb einer Point-to-Point Umgebung, stellt die sogenannte *Distributed-Object Middleware* dar. Diese Middleware wird auch oft als *Message Broker Architecture* bezeichnet. Mit dessen Hilfe lassen sich Objekte über mehrere Systeme hinweg integrieren. Ein Beispiel für einen solchen Message Broker ist die *Common Object Request Broker Architecture* (CORBA) [Omg01]. CORBA ermöglicht Objekten, mit anderen zu kommunizieren, gleichgültig welche Programmiersprache verwendet wurde oder auf welcher Plattform diese ausgeführt werden.

Abbildung 32 Die Prozess-, Objekt- und Datenebene



8.1.2 Integrationsmechanismen

Man findet derzeit eine Vielzahl von Mechanismen um die Übertragung von Informationen zwischen verschiedenen Systemen und Anwendungen zu realisieren. Drei häufig vorkommende Integrationsmechanismen sollen hier kurz vorgestellt werden. *Nachrichtenbasiert, Datenzugriff bzw. Filetransfer* und *Call Interfaces*. In der Praxis findet man hiervon oft auch diverse Mischformen.

Nachrichtenbasiert (Message-Oriented Middleware, asynchron)

Bei der Nachrichtenbasierten Integration erfolgt die Übertragung von Informationen durch den Versand von Nachrichten. Die Integration besteht dabei in der Regel aus asynchronner Kommunikation inerhalb der verteilten Systemumgebung. Diese Art von Kommunikation wird z.B. im IBM Produkt MQSeries oder beim Java Persistent Message Queuing (JPMQ) [SZ98] genutzt. Bei MQSeries schreibt ein Quellsystem einen Datensatz in eine Output-Warteschlange (*Queue*). Diese wird dann von MQSeries ausgelesen und in die Input-Queue eines Zielsystems geschrieben. Das Zielsystem kann den Datensatz dann auslesen. Die bereitgestellten Warteschlangen sind im allgemeinen persistent. Betrachtet man die Operationen des Ein- bzw. Auslagerns in die Warteschlangen als atomar und bietet die Warteschlange Möglichkeiten zur Festlegung einer Isolationssemantik und zur Prüfung der Konsistenz erhält man ein Transaktionssytem mit ACID-ähnlichen Eigenschaften [SZ98][Hr83].

Datenzugriff und Filetransfer

Datenzugriff und Filetransfer stellt Anwendungsintegration auf unterster Ebene (Datenebene) dar. Der Unterschied zur reinen Datenintegration besteht darin, das hier auch die Semantik der zu integrierenden Systeme bereitgestellt werden muss. Bei der Anwendungsintegration findet also eine Schemaintegration und eine Nachbildung des Zielsystems statt. Ein Beispiel für die Anwen-

dungsintegration durch Datenzugriff stellen Lese- und Schreiboperationen auf eine Datenbank dar. Bei verteilten Systemen kann es sich dabei auch um mehrere Standorte mit mehreren Datenbeständen handeln welche gegebenenfalls durch Replikationsmechanismen mit der zentralen Datenbank abgeglichen werden müssen. Der einfachste Fall für einen Filetransfer stellt eine von Anwendung A auf einem Netzlaufwerk erstellte Datei dar, welche dann von Anwendung B gelesen werden kann. In jedem Fall muss für die Konsistenz der Daten gesorgt werden.

Funktions-Schnittstellen (Distributed-Object Middleware, synchron)

Anwendungen können anderen Anwendungen sogenannte *Funktions-Schnittstellen* (Application Program Interface, API) bereit stellen. Diese können beinhalten:

- Objektschnittstellen (u. a. COM, CORBA, JavaBeans),
- Schnittstellen für Packaged-Applications[1] (u. a. SAP BAPI) oder
- Transaktionsschnittstellen (u. a. IBM CICS, BEA Tuxedo).

Beispielsweise kann Anwendung A bestimmte Methoden eines Business-Objects wie „Kunden hinzufügen“ oder „Order einstellen“ zur Verfügung stellen. Anwendung B ruft dann eine diese Funktionen über das API auf und kann sie für sich selbst nutzen. Von Vorteil ist dabei die synchrone Funktionsintegration. Nachteilig wirkt sich der erhöhte Programmieraufwand für die Bereitstellung der Funktions-Schnittstellen aus.

8.2 Probleme bei der Anwendungsintegration

Zur Nutzung der im vorigen Abschitt vorgestellten Integrationsmechanismen müssen vorab einige Probleme überwunden werden. Dazu gehören:

- Klare Identifikation der zur Verfügung stehenden Schnittstellen.
- Analyse eventueller struktureller Heterogenitäten der auszutauschenden Informationen, der zu integrierenden Systemen bezüglich der Plattform und des Programmiermodells.
- Konsistenzprobleme beim Zugriff mehrer Anwendungen auf einen Datenbestand.
- Wahl eines optimalen Integrationsmechanismus.

Die Identifizierung der Schnittstellen kann zu einem K. O. Kriterium bei der Anwendungsintegration werden. Nutzt eine Anwendung beispielsweise ein binäres Datenformat und fehlen die Informationen zu dessen Generierung bzw. Dekodierung ist eine Integration nicht möglich. Weiterhin muss festgestellt werden, ob ein direkter Zugriff auf die Anwendungsdaten möglich ist, oder ob dieser nur über die Anwendung realisiert werden kann.

Die Anwendungsintegration kann auch durch strukturelle Heterogenitäten erschwert werden. Beherrscht beispielsweise ein zu integrierendes System einen von anderen Systemen genutzten Datentyp nicht, muss dieser transformiert werden. Eine entsprechende Rücktransformation kann aber nicht immer gewährleistet werden (z. B. Rundungsfehler bei der Transformation Real->Integer->Real). Strukturelle Heterogenitäten jeglicher Art können auch durch die Integration ver-

1. Größtenteils fertig programmiert gekaufte Softwarepakete, die durch die Einstellung von Parametern auf spezifische Anforderungen angepasst werden.

schiedener Serverplattformen auftreten. Die Plattformen können sich dabei sowohl im Betriebssytem als auch in ihrer Architektur (Intel, Alpha oder SUN, bzw. 32 oder 64 Bit-Architektur) unterscheiden. Auch unterschiedliche Programmiermodelle können Probleme verursachen. So entstehen beispielsweise Konflikte beim Einsatz objektorientierter Programmiermodelle, wenn Seiteneffekte von Funktionen oder Rückgabewerte von Methoden keine Beachtung finden. Generell gilt, dass eine korrekte und vollständige Transformation von Daten eines semantisch mächtigeren Modells in ein semanisch ärmeres Modell nicht durchgeführt werden kann [HST97].

Ein weiteres Problem stellt die Frage der Konsistenzhaltung der Daten dar. Beschränkt sich also der Zugriff auf einen Datenbestand nicht auf eine einzelne Applikation, sollte der Zugriff unter der Einhaltung der ACID-Eigenschaften stattfinden. Eine Möglichkeit der Realisierung ist die im vorigen Abschnitt vorgestellte MOM.

Abschliessend bleibt die Wahl eines für das betrachtete Integrationsproblem passenden Integrationsmechanismus. Dabei sollten auch bereits absehbare Entwicklungen in der Anwendungsumgebung mit einbezogen werden um Anpassungen bei der Anwendungsintegration so gering wie möglich zu halten. Generische Integration ist Aufgrund der Vielzahl der existierenden Szenarien überaus komplex und kaum zu realisieren.

Nachdem nun verschiedene Wege und Mechanismen zur Anwendungsintegration vorgestellt wurden, soll im folgenden Kapitel eine konkrete Möglichkeit zur Anwendungsintegration, das *Simple Object Access Protocol* (SOAP) [SOAP01], behandelt werden. SOAP ist ein auf XML und bestehenden Netzwerkprotokollen basierendes Protokoll zur Ausführung von *Remote Procedure Calls* (RPC) [RPC92] innerhalb einer Client/Server-Architektur. SOAP dient also zur Nutzung von durch Server-Systeme bereitgestellte Funktions-Schnittstellen welche auf das SOAP-Protokoll zugeschnitten sein müssen. Mit den Erkenntnissen dieses und des folgenden Kapitels zeigt dann Kapitel 11 eine einfache Anwendungsintegration in das im Kapitel 7 vorgestellte Beispielsystem eines Web-basierten Handelssystems für Warentermingeschäfte.

Kapitel 9 Simple Object Access Protocol (SOAP)

Da Jetspeed nicht über integrierte Mechanismen zur Anwendungsintegration verfügt, diese aber zur Realisierung des in Kapitel 7 vorgestellten Handelssystems benötigt werden, soll hier das *Simple Object Access Protocol* (SOAP) vorgestellt werden. Es kann als Nachrichten-Transport-Protokoll für die Anwendungsintegration genutzt werden.

9.1 Übersicht

Der Wunsch, ein beliebiges Drittsystem in die Kommunikation zwischen zwei Systemen mit einbeziehen zu können, schafft einen Bedarf an standardisierten Protokollen. Protokolle die diesem Zweck dienen lassen sich in zwei Bereiche aufteilen [Geb01]:

1.) Protokolle die eine bestimmte, eng definierte Aufgabe erfüllen. Diese sind in der Regel textbasiert und vollständig dokumentiert (Beispielsweise *Simple Mail Transfer Protocol* (SMTP)[1] zum Transport von Email).

2.) Universell einsetzbare Protokolle. Diese besitzen meistens ein binäres Format und gehören oft zu Enwicklungssystemen für verteilte Software (beispielsweise das IIOP Protokoll von CORBA [Omg01]).

Durch die Entwicklung von XML wurde die Realisierung neuer Protokolle erleichtert. Einen besonderen Vorteil liefert XML durch die Fähigkeit, beliebige Daten sebstbeschreibend darzustellen. Das *Simple Object Access Protcol* (SOAP) [SOAP01] ist ein Protokoll zur Durchführung von *Remote Procedure Calls* (RPC) [RPC92] und der damit verbundenen Übermittlung von Nachrichten in verteilten Systemen mit XML. Die aktuelle SOAP Spezifikation umfasst in der Version 1.1 folgende Elemente:

- die Struktur des XML-Dokuments, das eine SOAP-Nachricht enthält
- einen Mechanismus zur Kodierung von gängigen Datentypen (z. B. Float, Integer, etc.) in XML-Konstrukten
- den Einsatz von HTTP als Transportprotokoll für SOAP-Nachrichten
- und eine Konvention zur Durchführung von RPC.

1. SMTP RFC 821; http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc0821.html

SOAP-Nachrichten sind also ein Verbund von Nutzdaten und der zur Auswertung und Transport benötigten Informationen. Der Einsatz von SOAP vereinfacht so deutlich die Fehlersuche in auf SOAP basierenden Anwendungen, da sich die Kommunikation ohne zusätzliche Hilfsmittel beobachten lässt. Weiterhin ist SOAP mit seiner textbasierten Kommunikation und der Möglichkeit zur Kopplung an gängige Protokolle wie HTTP in Bezug auf Firewalls einfacher handhabbar, da die Kommunikation für diese Protokolle in der Regel freigegeben ist. Die einzelnen Komponenten sollen im Folgenden näher betrachtet werden.

9.2 Struktur einer SOAP-Nachricht

9.2.1 Grundlagen

Eine SOAP-Nachricht besteht aus *Envelope*, *Header* und dem *Body.*

Envelope

Der Envelope enthält als Wurzelelement den gesamten Inhalt einer SOAP-Nachricht. Dabei werden Namespaces[1] verwendet um Konflikte zwischen den im SOAP-Standard definierten und den eigenen XML-Elementen zu vermeiden. Namespaces sorgen so für eindeutige Element- und Attributnamen innerhalb von XML-Dokumenten. So wird im Envelope als Namespace die URI „http://schemas.xmlsoap.org/soap/envelope/“ angegeben. Header und Body sind Subelemente des Envelope wobei der Header optional ist und der Body die eigentliche Information der SOAP-Nachricht enthält.

Header

Im Header werden Metainformationen für den Empfänger oder eventuelle Zwischenstationen (z. B. Proxies bei der Nutzung des HTTP-Protokolls) beim Transport der Nachricht hinterlegt. In ihm sind keine Informationen welche die aufzurufende Methode betreffen enthalten, sondern enthält Informationen, die zur Erweiterung der Funktionalität genutzt werden können. Da im SOAP-Standard keine Authorisierungs- oder Transaktionsverfahren spezifiziert sind, können im Header beispielsweise eine Transaktion-ID oder Sicherheitsinformationen übertragen werden. Weiterhin ist innerhalb der SOAP-Spezifikation weder die Adressierung noch der Weg einer Nachricht zu ihrem Empfänger näher spezifiziert. Lediglich mit dem *actor* Attribut kann die Verarbeitung auf dem Weg zum endgültigen Empfänger beeinflusst werden.

Das *actor* Attribut

Wie bereits erwähnt müssen SOAP-Nachrichten nicht direkt an den Empfänger gesand werden sonderen können über Zwischenstationen (SOAP-Intermediaries) weitergeleitet werden. Diese geben die empfangene Nachricht nicht einfach weiter sondern analysieren diese und werten die für sie bestimmten Inhalte aus. Bevor die Nachricht an die nächste Station weitergeleitet wird, entfernt der aktuelle Intermediary alle von ihm bearbeiteten Teile und fügt gegebenenfalls neue Elemente ein. Jede so am Transport beteiligte Station und auch der endgültige Empfänger sind

1. Namespaces in XML, W3C Recommendation; http://www.w3c.org/TR/REC-xml-names

über eine URI identifizierbar. Das *actor* Attribut enthält eine solche URI und kann in jedem Subelement des SOAP-Headers verwendet werden. Das actor Attribut legt also fest, wer das dazugehörige Element auswerten soll. Fehlt das actor Attribut ist das Element für den endgültigen Empfänger der SOAP-Nachricht bestimmt. Alle Elemente im Body haben keine actor Attribute und sind somit automatisch für den endgültigen Empfänger bestimmt.

Das *mustUnderstand* Attribut

Das *mustUnderstand* Attribut legt fest, ob das korrespondierende Element vom Empfänger zwingend verarbeitet werden muss. Es ist ein boolesches Attribut mit den möglichen Werten „0“ oder „1“. Der Defaultwert des Attributs ist für alle Headerelemente „0“ und für Elemente innerhalb des Bodys unveränderlich „1“. Kann ein Element mit gesetztem mustUnderstand Attribut nicht verarbeitet werden, muss die komplette SOAP-Nachricht verworfen werden. So können kritische Inkompatibilitäten beim Transport und der Auswertung einer Nachricht sofort bemerkt werden. Dem Sender wird eine entsprechende Fehlermeldung zugestellt.

Body

Der Body und dessen Subelemente sind anwendungsspezifisch. Standardmässig ist nur ein Elementtyp, der SOAP-*Fault*, spezifiziert. Er dient der Übermittlung von Fehler- und Statusinformationen. Innerhalb des Fault-Element sind folgende Subelemente festgelegt:

- **faultcode:** Enthält einen Code zur algorithmischen Identifizierung der Fehlersituation. Der SOAP-Standard spezifiziert ein Schema zur Generierung dieser Codes[1].
- **faultstring:** In diesem Element ist eine an den Endbenutzer gerichtete Fehlermeldung in einem Klartextformat enthalten.
- **faultactor:** Dieses Element enthält die URI des SOAP-Intermediaries der den Fehler verursacht hat.
- **detail:** Falls der Fehler durch ein Element im SOAP-Body ausgelöst wurde, müssen in diesem Element applikationsspezifische Detailinformationen zum Fehler enthalten sein.

Die fehlerfreie Übertragung von SOAP-Nachrichten muss durch das unterliegende Transportprotokoll sichergestellt werden. Innerhalb des SOAP-Standards sind dazu keine Mechanismen vorgesehen. Wird dies nicht durch das unterliegende Transportprotokoll sichergestellt (z. B. bei der Verwendung von SMTP), müssen diese Aufgaben von den Anwendungen übernommen werden.

9.2.2 Der Einsatz von SOAP anhand eines Beispiels

Anhand eines Beispiels soll die Verwendung von SOAP und die Bedeutung der eingesetzen Attribute und Elemente schrittweise demonstriert werden [Tar01]. Als Grundlage dient das Java-Interface aus .

1. SOAP Fault Codes (http://www.w3.org/TR/SOAP/#_Toc478383510)

Beispiel 5

```
public interface Hello
{
 public String sayHelloTo(String name);
}
```

Der Aufruf der `sayHelloTo()` Methode durch einen Client soll durch die Übergabe eines Parameters eine personalisierten Begrüssungstext ausgeben. In XML verpackt könnte der Aufruf von `sayHelloTo(„John“)` wie in Beispiel 6 gezeigt aussehen. Der Interfacename dient hier als Wurzelknoten, der Methodenname und der Parametername sind ebenfalls als Knoten realisiert. Dies entspricht vereinfacht der Codierung in SOAP.

Beispiel 6

```
<?xml version="1.0"?>
<Hello>
  <sayHelloTo>
     <name>John</name>
  </sayHelloTo>
</Hello>
```

Nachdem die Anfrage an einen Server übermittelt und ausgewertet wurde, sendet dieser eine Antwortnachricht in der Form von Beispiel 7 zurück. Der Wurzelknoten besteht weiterhin aus dem Interface-Namen `Hello`. Der Methodenname wurde jedoch um `Response` ergänzt, um erkennen zu lassen, dass es sich bei dieser Nachricht um eine Antwort auf eine Anfrage handelt. Bei mehrern gleichzeitigen Anfragen erfolgt die Zuordnung der Ergebnisse also durch das Suchen nach dem Methodennamen zuzüglich des Strings „Response“.

Beispiel 7

```
<?xml version="1.0"?>
<Hello>
  <sayHelloToResponse>
     <message>Hello John, How are you?</message>
  </sayHelloToResponse>
</Hello>
```

Die Codierung der obigen Anfrage in SOAP zeigt Beispiel 8. Hier ist die bereits vorgestellte Envelope-, Header- und Body-Struktur erkennbar. Weiterhin wurde mit `ns1` ein eigener Namespace eingeführt um Überschneidungen mit den im Envelope referenzierten SOAP-Namespaces zu vermeiden.

Beispiel 8

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
                   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
                    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Header>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
   <ns1:sayHelloTo xmlns:ns1="Hello“
          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <name xsi:type="xsd:string">John</name>
   </ns1:sayHelloTo>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Typdefinitionen wie z. B. `xsd:string` basieren auf dem durch das Attribut `encoding-Style` mit der URI „http://schemas.xmlsoap.org/soap/encoding“ eingebundenen Kodierungsschema. Dieses Schema ist jedoch lediglich eine Empfehlung und kann durch andere oder eigene Schemata ersetzt werden. Das Attribut `encodingStyle` kann in jedem Element enthalten sein und ist für dessen Inhalt und alle Subelemente gültig. Das Format der Nachricht ist also weitgehend flexibel gestaltet, solange Client und Server bei der Wahl der Darstellung übereinstimmen und es sich um gültige XML-Konstrukte handelt. Der Interfacename `Hello` gehört nun zum eigenen Namespace `ns1`. Eine SOAP-konforme Antwort auf die obige Anfrage zeigt Beispiel 9. Die Antwort findet sich im Body der Nachricht innerhalb des `return` Parameters.

Beispiel 9

```
<SOAP-ENV:Envelope
           xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
           xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:sayHelloToResponse xmlns:ns1="Hello"
       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <return xsi:type="xsd:string">Hello John, How are you doing?</return>
    </ns1:sayHelloToResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Wie im vorigen Abschitt beschrieben, kann der Header auch zur Übertragung von Transaktionsinformationen genutzt werden. Wie ein solches Header-Element aussehen kann zeigt Beispiel 10.

Beispiel 10

```
<SOAP-ENV:Header>
  <t:Transaction xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
    5
  </t:Transaction>
</SOAP-ENV:Header>
```

Der Namespace `t` stammt aus einer Anwendungsspezifischen URI. `5` steht für die Transaktions-ID zu der die eventuell im Body aufgerufenen Methode gehört. Das MustUnderstand-Attribut sorgt dafür, das die Nachricht verstanden werden muss oder eine Fehlermeldung erzeugt wird. Dies ist besonders im Transaktions-Umfeld wichtig. Eine Fehlermeldung kann bei Bedarf durch das Fault-Element übermittelt werden. Unter Vorraussetzung einer Anfrage wie in Beispiel 8 und der Annahme das der Server solche Anfragen nicht an Dienstagen annehmen darf würde eine Fehlermeldung wie in Beispiel 11 aussehen.

Beispiel 11

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <e:myfaultdetails xmlns:e="Hello">
          <message>
             Sorry, my silly constraint says that I cannot say hello on Tuesday.
          </message>
          <errorcode>
            1001
          </errorcode>
        </e:myfaultdetails>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Hier sieht man beispielhaft die Anwendung der im vorigen Abschnitt vorgestellten Fault-Subelemente. Wird eine Fehlermeldung durch ein MustUnderstand-Attribut ausgelöst fehlt das `detail` Element. Diese Tatsache lässt einen einfachen Rückschluss zu, ob eine Fehlermeldung bei der Verarbeitung des Headers oder des Bodys entstanden ist. Beispiel 12 zeigt eine solche Fehlermeldung.

Beispiel 12

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
   <SOAP-ENV:Body>
     <SOAP-ENV:Fault>
       <faultcode>SOAP-ENV:MustUnderstand</faultcode>
       <faultstring>SOAP Must Understand Error</faultstring>
     </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP lässt sich wie bereits erwähnt, in Kombination mit verschiedenen Protokollen wie HTTP oder SMTP einsetzen. Im Folgenden soll die Kopplung an das HTTP Protokoll betrachtet werden.

9.2.3 SOAP und HTTP

Die Kommunikation über HTTP erfolgt indem der Client über den *POST*-Befehl eine SOAP-Nachricht, welche die gewünschte Anfrage enthält, an den Server schickt. Dieser antwortet anschließend über dieselbe Verbindung und sendet eine SOAP-Nachricht mit dem Ergebnis der Anfrage oder eine Fehlermeldung an den Client zurück. Der Header des POST-Befehls enthält neben den für das HTTP-Protokoll üblichen Feldern *Content-Type* und *Content-Length* ein SOAP-spezifisches Feld *SOAP-Action*. Diese Feld enthält eine URI als Hinweis auf den Zweck der SOAP-Nachricht. Das Feld hat aber keinerlei Einfluß auf deren Interpretation und soll vielmehr Dritten zur Information über den Inhalt der Nachricht dienen. Dies kann beispielsweise in Firewalls und Paketfiltern genutzt werden um die SOAP-Kommunikation besser überwachen und kontrollieren zu können. Beispiel 13 zeigt eine über das HTTP-Protokoll übertragene SOAP-Nachricht. Die entsprechende Antwort wird in Beispiel 14 dargestellt.

Beispiel 13

```
POST http://www.SmartHello.com/HelloApplication HTTP/1.0
Content-Type: text/xml; charset="utf-8"
Content-Length: 587
SOAPAction: "http://www.SmartHello.com/HelloApplication#sayHelloTo"
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Header>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
     <ns1:sayHelloTo xmlns:ns1="Hello" SOAP-ENV:encodingStyle="
        http://schemas.xmlsoap.org/soap/encoding/">
          <name xsi:type="xsd:string">Tarak</name>
     </ns1:sayHelloTo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Beispiel 14

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 615
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:sayHelloToResponse xmlns:ns1="Hello"
         SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <return xsi:type="xsd:string">Hello John, How are you doing?</return>
      </ns1:sayHelloToResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Das Ziel der Anfrage wird mit dem POST-Befehl übergeben. Wenn bei der Übertragung keine Fehler auftreten, enthält die erste Zeile der Antwort den Rückgabewert „`HTTP/1.0 200 OK`“. Ansonsten wird als Rückgabewert „`HTTP/1.0 500 Internal Server Error`“ zusammen mit einer SOAP-Nachricht, die ein entsprechendes SOAP-Fault Element enthält, generiert. Die Möglichkeit einer detailierteren Beschreibung der versandten Nachrichten innerhalb des HTTP-Protokolls bietet das HTTP Extension Framework [NLM00]. Dieses soll hier jedoch nicht näher betrachtet werden.

Kapitel 10 Prototypische Realisierung eines Web-basierten Handelssystems mit Jetspeed

In diesem Kapitel soll Jetspeed genutzt werden, um prototypisch einen Teil der in Kapitel 7 vorgestellten Anforderungen an ein Web-basiertes Handelssystem für Warentermingeschäfte zu implementieren. Nach der Auswahl der zu verwendenden Entwicklungsumgebung sollen die zu realisierenden Komponenten festgelegt werden.

10.1 Festlegen der Entwicklungsumgebung

Da Jetspeed auf Java basiert, ist eine weitgehende Plattformunabhängigkeit gegeben. Vorraussetzung ist lediglich ein Web-Server welcher die Ausführung von Servlets unterstützt. Dabei ist auf eine Kompatibilität zur Servlet 2.2 API zu achten, da Jetspeed dies voraussetzt. Der genutzte Java JDK sollte die Version 1.2 oder höher besitzen. Als Server-Plattform soll ein Linux-System zum Einsatz kommen. Die verwendete Distribution spielt dabei keine Rolle, sollte jedoch so aktuell sein, das die benötigten Software-Komponenten lauffähig sind. Bei der Auswahl der Hardware ist, je nach Ansprüchen an die Performance des Systems, ein leistungsfähiger Rechner mit genügend Arbeitsspeicher zu wählen. Diese Anforderungen sind durch den geplanten Einsatz von Java bedingt.

Im Folgenden wird von dieser Systemkonfiguration ausgegangen:

- Intel Pentium III 550 Mhz
- 512 Mb RAM
- Debian Linux 2.2r3
- Java 2 SDK 1.4.0
- Jakarta Tomcat 4.0-b7 als Webserver und Servlet-Engine
- SOAP 2.2 (mit diversen eigenen Weiterentwicklungen)

Benötigter Festplattenplatz für das System inklusive Entwicklungsumgebung liegt bei ca. 150 Mb. Die Installationsanweisungen für dieses System finden sich in Anhang A.

10.2 Festlegung der zu realisierenden Komponenten

Innerhalb dieses Kapitels sollen folgende Komponenten des Handelssystems mit Jetspeed realisiert werden:

- **Anlegen von Benutzern** und senden einer Bestätigungs-Email mit einem Sicherheits-Code zur Verifizierung der angegebenen Email-Adresse. Dies soll erst nach dem Akzeptieren der Allgemeinen Geschäftsbedingungen möglich sein.
- Möglichkeit zur **Suche im Web** mit unterschiedlichen Suchmaschinen.
- Anzeige von **Marktinformationen** der deutschen Börse (Grafische Darstellung der Indexverläufe des DAX und des NEMAX).
- Einbindung eines **Newstickers**.

Dabei werden die von Jetspeed zur Verfügung gestellten Möglichkeiten zur Personalisierung genutzt. Nach der Anmeldung soll dem Benutzer eine Standard-Umgebung präsentiert werden, die er sich bei Bedarf erweitern und anpassen kann. Das Zusammenstellen dieser Startumgebung für neue Benutzer erfolgt, in dem man die Umgebung des Benutzers „turbine“ entsprechend anpasst. Wird ein neuer Benutzer angelegt, erhält dieser automatisch die zur Personalisierung verwendete PSML-Datei des „turbine“-Benutzers. Das Passwort des Benutzers „turbine“ lautet ebenfalls „turbine“ und sollte nach der Installation von Jetspeed geändert werden. Bei den Benutzernamen und Passwörtern wird Groß- und Kleinschreibung unterschieden. Der Administrations-Account lautet „admin“ mit dem Initialpasswort „jetspeed“. Auch dieses sollte verändert werden. Der Administrations-Account ermöglicht eine Übersicht über das gesamte Portal inklusive der Rechte- und Benutzerverwaltung.

10.3 Grundsätzliches

Die hier verwendete CVS-Version von Jetspeed (Version 1.3a2-dev, Stand 06.11.2001) unterscheidet sich in einigen Bereichen von der letzen offiziellen Release 1.3a1. Dies erschwert die Entwicklung eines eigenen Portals, da sich die nur knapp bemessene Dokumentation auf die Version 1.3a1 bezieht und nur nach und nach angepasst wird. Viele nützliche Tipps finden sich jedoch in den entsprechenden News-Gruppen und Mailing-Listen [TMA01].

Um die Dokumentation zu erleichtern, werden Dateien im Folgenden von den Basisverzeichnissen des entsprechenden Projekts ausgehend referenziert. Bei Jetspeed und SOAP gelten die Verzeichnisse mit den Quellen als Basis, bei Tomcat die des installierten und laufenden Servers. Weiterhin wird beschrieben, ob die Datei geändert (G) oder hinzugefügt (H) wurde. Eine Referenz kann also beispielsweise wie folgt aussehen: `{TC, G, bin/startup.sh}`.

Weiterhin wird davon ausgegangen das alle Installationen wie in Anhang A beschrieben stattgefunden haben. Bei der Entwicklung sollte darauf geachtet werden, das bei Jetspeed und SOAP die Dateien nur innerhalb der Quellen geändert werden, da sonst bei einer Neu-übersetzung die Änderungen verloren gehen. Nach Neu-übersetzungen ist das Jetspeed bzw. SOAP WAR-Archiv wieder, wie in Anhang A beschrieben, innerhalb der Tomcat-Umgebung zu installieren. Tomcat muss danach neu gestartet werden. Alle zur Realisierung veränderten oder hinzugefügten Dateien finden sich in Anhang B.

10.3.1 Registrierung neuer Portlets

Werden neue Portlets erstellt, müssen diese Jetspeed bekannt gemacht werden. Dies geschiet durch das Hinzufügen eines Eintrags in die entsprechende Registrierungsdatei. Genauere Informationen zum Format dieser Datei finden sich in Kapitel 6. Die Registrierungsdateien befinden sich im Verzeichnis `{JS, , webapp/WEB-INF/conf}`. Jetspeed nutzt standardmässig `{JS, G, webapp/WEB-INF/conf/portlets.xreg},` wertet aber alle Dateien mit der Endung `.xreg` aus die sich in diesem Verzeichnis befinden. Um die Übersichtlichkeit zu steigern, sollte man für die selbst entwickelten Portlets eine eigene Datei anlegen. Sollen jedoch Portlets hinzugefügt werden welche auf bestehenden Jetspeed-Portles aufbauen, beispielsweise ein neues RSS-Portlet, muss der `ref`-Eintrag in der Registrierungsdatei stattfinden, in der sich auch der `instance`- oder `abstract`-Eintrag des referenzierten Portlets befindet. Die im Folgenden erstellten Portlets werden somit zum Teil in der Standarddatei oder in der neu erstellten `{JS, H, webapp/WEB-INF/conf/local-portlets.xreg}` registriert.

10.3.2 Entwurfsentscheidung

Wie bereits in Kapitel 6 erwähnt, stehen zum Entwurf neuer Portlets primär ECS, Velocity oder JSP als Template-Engine zur Verfügung. Im allgemeinen ist bei der Entwicklung von Jetspeed eine Tendenz von ECS nach Velocity zu beobachten. Das komplette Erscheinungsbild wird von Velocity-Templates bestimmt. Da die Template-Engines modular in Jetspeed integriert sind, besteht auch die Möglichkeit andere hinzuzufügen. Im Folgenden wird zur Entwicklung Velocity und ECS genutzt.

10.4 Realisierung

10.4.1 Verändern des Erscheinungsbildes des Standard-Jetspeed Portals

Die Nutzung des Jetspeed-Portals zum Erstellen eigener Portale soll für den Endanwender möglichst transparent sein. Durch die Verwendung von Template-Engines lässt sich der statische vom dynamischen Inhalt sehr gut trennen. Daher lässt sich Jetspeed in ein selbst erstelltes Web-Design relativ einfach einbetten. Dazu müssen die entsprechenden Templates angepasst werden. Hier zeigen sich erneut die Vorteile des bei der Jetspeed-Entwicklung genutzen MVC-Design-Patterns. Das Basis-Template `{JS, G, webapp/WEB-INF/templates/vm/layouts/html/default.vm}` enthält das Basis-Layout des Portals. Es beinhaltet den `$screen_placeholder`, welcher später dynamisch durch die vom Portal erzeugte Portlet-Anordnung ersetzt wird. Das Basis-Layout bestimmt also den Rahmen, in dem dann die dynamischen und personalisierten Inhalte präsentiert werden. Darin werden weitere Templates integriert, welche ebenfalls angepasst werden können:

```
{JS, G, webapp/WEB-INF/templates/vm/navigations/html/left.vm}
{JS, G, webapp/WEB-INF/templates/vm/navigations/html/bottom.vm}
{JS, G, webapp/WEB-INF/templates/vm/navigations/html/top.vm}.
```

Um die im Customizer angebotenen Portlets auf das Nötigste, bzw. auf die eigenen Portlets zu reduzieren, wurde die Portlet-Registrierungsdatei sowie die Dateien zur Steuerung der via RSS bzw. Open-Content-Syndication (OCS) referenzierten Daten bearbeitet. Die Referenzen werden in den folgenden Dateien festgelegt:

```
{JS, G, webapp/ocs/local.ocs}
{JS, G, webapp/rss/Jetspeed.rss}
{JS, G, webapp/rss/admin.rss}
{JS, G, webapps/about/index.rss}.
```

Zu beachten ist, dass die hinzugefügten Portlets nur dann im Customizer sichtbar sind, wenn ihnen in der Registrierung ein gültiger Media-Type zugewiesen wurde. Dies kann beispielsweise mit `<media-type ref="html"/>` oder `<media-type ref="wml"/>` geschehen.

Um die zu Beginn eines Portalbesuchs anzuzeigenden Portlets festzulegen, müssen die Dateien
`{JS, G, webapp/WEB-INF/psml/anon/html/default.psml}`
`{JS, G, webapp/WEB-INF/psml/anon/html/en/default.psml}`
angepasst werden. Für diese Diplomarbeit wurde ein Welcome-Portlet mit einem Begrüssungstext erstellt und in diesen Dateien angegeben. Der Quellcode findet sich jeweils wieder in Anhang B.

10.4.2 Anlegen von Benutzern

Die Anforderungen zum Anlegen von Benutzern lassen sich größtenteils mit den Bordmitteln von Jetspeed realisieren. Im dafür vorgesehene Velocity-Template müssen dazu die statischen Inhalte angepasst werden. Zusätzlich muss das Template um eine Abfrage zur Akzeptanz von allgemeinen Geschäftsbedingungen (AGB) erweitert werden. Dazu bietet sich der Einsatz von JavaScript zur Prüfung der Benutzereingaben an. Das Velocity-Template findet man unter `{JS, G, webapp/WEB-INF/templates/vm/screens/html/NewAcount.vm}.`
Der `<form>`-Tag muss um einen Parameter `name` erweitert werden um die zur Abfrage eingesetzte Checkbox prüfen zu können. Die wichtigen Änderungen am NewAccount-Template sind im Anhang B hervorgehoben. Weiterhin muss die in der CVS-Version abgeschaltete E-Mail-Funtionalität reaktiviert werden, welche nach dem Anlegen des neuen Benutzers diesen über den Erfolg informiert und zur Eingabe eines Sicherheits-Codes zur Überprüfung der Email-Adresse auffordert. Dazu müssen die Dateien

```
{JS, G, webapp/WEB-INF/conf/JetspeedResources.properties}
{JS, G, webapp/WEB-INF/conf/TurbineResources.properties}
```

geändert werden. In ersterer müssen dazu die Variablen `confirm.email.from`, `confirm.email.name` auf entsprechende Werte sowie `confirm.email.enable` auf `true` gesetzt werden. In der zweiten Datei sollte `mail.smtp.from` auf eine gültige Email-Adresse verweisen. Damit wäre die erste Anforderung realisiert.

10.4.3 Suche im Web

Die Möglichkeit zur allgemeinen Suche im Web wird direkt von Jetspeed zur Verfügung gestellt. Das Search-Portlet ist eine Referenz auf das HTML-Portlet und stellt die HTML-Datei `{JS, , webapp/search/index.html}` dar. Diese enthält bereits eine Vielzahl von Suchmaschinen, kann aber bei Bedarf auch erweitert werden.

10.4.4 Portlet-Erstellung am Beispiel „Marktinformationen“

Das Marktinformationen-Portlet soll eine grafische Übersicht der deuschen Aktienindizes DAX und NEMAX liefern. Dazu gibt es zwei Möglichkeiten. Die eine wäre eine periodische Sammlung der Werte in einer Datenbank mit anschliessender Generierung der Grafiken bei Bedarf. Da im Internet jedoch bereits viele solcher Angebote existieren, sollen diese Informationen von einem anderen Anbieter bezogen und in Form eines Portlets integriert werden. Als Anbieter wurde dazu die Gatrixx AG aus Berlin ausgesucht. In deren Web-Angebot[1] wurden die URL für entsprechende Grafiken identifiziert und in ein mit ECS erstelltes Portlet integriert. Der Quellcode findet sich ebenfalls in Anhang B. Die URL für die beiden Grafiken lauten:

DAX: `http://gfx.finanztreff.de/charts/cc_gatrixx.gfx?typ=1&string=DAX&borse=9&uvon=0900&ubis=2015&land=276&h=100&b=130&realtime=1`

NEMAX: `http://gfx.finanztreff.de/charts/cc_gatrixx.gfx?typ=1&string=NMKX&boerse=9&uvon=0900&ubis=2015&land=276&h=100&b=130&realtime=1`

Beispiel 15 Beispiel für ein Portlet-Grundgerüst

```
package org.apache.jetspeed.portal.portlets;

import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.turbine.util.RunData;
import <Weitere benötigte Klassen, z. B. ECS oder ähnliches>

public class <Portletname> extends AbstractPortlet
{
   <beliebige Funktionsdefinitionen und Funktionen>
   public ConcreteElement getContent(RunData runData) {
       <Realisierung einer Funktion>
       return <Rückgabewert>;
   }
}
```

Bei der Erstellung des Portlets ist darauf zu achten, dass die von den Jetspeed-Entwicklern vorgegebenen Programmierrichtlinien[2] eingehalten werden. Ansonsten kann es zu Problemen bei der Integration des Portlets in die Jetspeed-Umgebung kommen oder das Portlet verhält sich nicht wie erwartet. Daher sollte das neue Portlet stehts auf dem `AbstractPortlet` basieren und

1. http://www.finanztreff.de
2. http://jakarta.apache.org/jetspeed/site/code-standards.html

den Rückgabetyp `ConcreteElement` liefern. Eine mögliche Grundlage für ein Portlet zeigt Beispiel 15. Dabei zeigt das `package`-Kommando an, wo sich die neue Portlet-Klasse befindet. Der entsprechende Dateibaum findet sich unter `{JS, , src/java}` in dem dann auch an der passenden Stelle der Java-Quellcode mit dem Dateinamen `<Portletname>.java` abgelegt werden sollte. Mit dem `import`-Kommando werden bereits bestehende Klassen importiert, deren Methoden dann im zu erstellenden Portlet genutzt werden können. Die Klasse `org.apache.turbine.util.RunData` bietet beispielsweise Zugriffsmöglichkeiten auf Benutzerinformationen. So liefert `aRunData.getUser().getFirstName()` den Vornamen des gerade angemeldeten Benutzers. Informationen, die dem Portlet mittels Parameter-Einträgen in der Portlet-Registry übergeben werden, können mit `getPortletConfig().getInitParameter("<ParamName>")` ausgewertet werden.

Wie bereits erwähnt, sorgt die Klasse `AbstractPortlet` für die ordnungsgemäße Integration in die Gesamtumgebung. Der Rückgabewert des neu erstellten Portlets vom Typ `ConcreteElement` enthält dabei in der Regel den in die Portalseite in einen Portletrahmen einzufügenden Inhalt. Dieser Inhalt ist, jenachdem für dieses Portlet in der Portlet-Registry angegebenen Media-Typ, in HTML oder WML zu formulieren.

Abschliessend muss das Portlet noch in der Portlet-Registry bekannt gemacht werden. Dies geschieht in der Datei `{JS, H, webapp/WEB-INF/conf/local-portlets.xreg}` und ist für das Marktinformations-Portlet in Beispiel 16 gezeigt.

Beispiel 16 Registry-Eintrag für das MarketInfo-Portlet

```
<portlet-entry name="MarketInfo" hidden="false" type="instance"
    application="false">
    <meta-info>
      <title>Marktionformationen</title>
      <description>DAX und NEMAX Verlauf</description>
    </meta-info>
    <classname>org.apache.jetspeed.portal.portlets.MarketInfoPortlet</classname>
    <media-type ref="html"/>
</portlet-entry>
```

10.4.5 RSS-Content Integration am Beispiel eines Newstickers

Die Integration eines RSS-Content stellt kein grosses Problem dar. Das bei Jetspeed enthaltene RSS-Portlet bietet alle dazu notwendigen Funktionen. So wird hier ein neues Portlet wiederum durch die Referenz auf ein bestehendes Portlet, in diesem Fall das RSS-Portlet erstellt. Dieses Vorgehen wurde bereits beim Portlet für die Websuche angewand. Nachdem die URI für den entsprechenden RSS-Content identifiziert ist, muss er dem Portlet als Parameter übergeben werden. Für den hier ausgewählten Newsticker lautet diese URI `http://headlines.internet.com/internetnews/bus-news/news.rss`. Die Registrierung des Newsticker-Portlets zeigt Beispiel 17.

Beispiel 17 Registry-Eintrag für das HeiseNT-Portlet

```
<portlet-entry name="NTPortlet" hidden="false" type="ref" parent="RSS"
    application="false">
    <meta-info>
       <title>Newsticker</title>
       <description>Aktuelle Business-News</description>
    </meta-info>
    <url>http://headlines.internet.com/internetnews/bus-news/news.rss</url>
</portlet-entry>
```

Damit sind nun alle für dieses Kapitel angestrebten Funktionalitäten realisiert. Eine personalisierte Zusammenstellung aller in diesem Kapitel erstellten Komponenten zeigt Abbildung 33.

Abbildung 33 Personalisiertes Jetspeed mit allen bisweilen realisierten Komponenten

Im folgenden Kapitel soll nun noch ein Beispiel für eine Anwendungsintegration gegeben werden.

Kapitel 11 Anwendungsintegration in das Web-basierte Handelssystem mit SOAP

11.1 Festlegung der zu realisierenden Komponenten

In diesem Kapitel soll als Beispiel für eine Anwendungsintegration ein Zahlungsmodul realisiert werden. Mit diesem Modul soll die Begleichung von durch Transaktionen im Handelssystem entstandenen Verbindlichkeiten mittels Kreditkartenzahlung ermöglicht werden. Dazu wird das Zahlungsmodul die Gültigkeit einer Kreditkarte bei einem Server anfragen und entsprechend den Ausgleich des Kontostandes innerhalb des Handelssystems vornehmen. Dieser Web-Service wird auf der Seite www.iemfamily.com in Form eines MS .NET SOAP-Servers angeboten. Da das Handelssystem an sich nicht existiert, wird ein fester Wert für die Verbindlichkeiten angenommen. Die Anfrage nach der Gültigkeit der Karte wird mit dem in Kapitel 9 vorgestellten SOAP-Protokoll stattfinden.

Das Beispiel für die Anwendungsintegration soll in zwei Schritten realisiert werden. Zuerst soll ein Portlet entwickelt werden, welches ohne Benutzerinteraktion eine fest vorgegebene Kreditkartennummer prüft. Danach soll dieses Portlet zum Zwecke der Interaktion mit dem Benutzer durch ein weiteres Portlet genutzt werden. Die Grundsätze aus Kapitel 10 sind weiterhin gültig.

11.2 Realisierung

Zu Beginn soll bemerkt werden, das der SOAP-Server, welcher die Kreditkarten auf ihren Typ und ihre Gültigkeit hin überprüft, in seiner Konfiguration und in der Funktionalität nicht beeinflussbar ist. Er wird von Mike Iem[1] betreut. Es sind lediglich die Parameter bekannt die übergeben werden müssen. Diese zeigt Tabelle 3 auf Seite 90.

1. mikeiem@hotmail.com

Tabelle 3 Verbindungsparameter zum SOAP-Server

Parameter	Wert
SOAP Endpoint URL	http://www.iemfamily.com/iemfamily/Creditard.asmx
SOAPAction	http://tempuri.org/VerifyCreditCard
Method Namespace URI	http://tempuri.org/
Method Name	VerifyCreditCard
Value-Parameter	CardNumber

Mit diesen Angaben kann nun ein Client programmiert werden, welcher im Parameter `CardNumer` die zu überprüfende Kreditkartennummer übergibt. Dies wurde mit dem `{JS,H,src/java/org/apache/jetspeed/portal/portlets/SOAPDemoPortlet.java}` realisiert. Die Registrierung für dieses Portlet in `{JS, H, webapp/WEB-INF/conf/local-portlets.xreg}` zeigt Beispiel 18. Bei der Realisierung dieses Portlets traten jedoch Probleme auf, die im folgenden Abschnitt besprochen werden sollen.

Beispiel 18 Registrierung SOAPDemoPortlet

```
<portlet-entry name="SOAPDemo" hidden="false" type="instance"
               application="false">
        <meta-info>
           <title>SOAP-Demo</title>
           <description>Beispiel fuer Anwendungsintegration</description>
        </meta-info>
        <classname>org.apache.jetspeed.portal.portlets.SOAPDemoPortlet</classname>
        <media-type ref="html"/>
    </portlet-entry>
```

11.2.1 Interoperabilitätsprobleme

Als sehr problematisch haben sich Interoperatibilitätsprobleme zwischen dem verwendeten Apache-SOAP Client und dem Microsoft .NET SOAP-Server herausgestellt. Leider entstehen bei der Verwendung verschiedener SOAP-Implementierungen aufgrund des Nachrichtenformats immer wieder Probleme beim Datenaustausch. Man unterscheidet *Transport-Probleme*, *XML-Probleme*, und *SOAP-spezifische Probleme* [SOAP01b]. Hier sollen jedoch nur die Transport- und SOAP-spezifischen Probleme näher betrachtet werden, da diese bei der Kreditkartenanwendung aufgetreten sind.

Transport-Probleme

Mit Apache-SOAP lassen sich keine SOAP-Bodys erzeugen, ohne einen Namespace zu benutzen. Ist der .NET-Dienst so konfiguriert, das im Body im sogenantten „Literal“-Format verwendet wird, kann dieser die entsprechenden Anfragen nicht verstehen. Bei der Verwendung des „Literal“-Formats wird der Inhalt einer Nachricht mittels eines XSD-Schemas, welches sich in einer *Web Services Description Language* (WSDL) [WSDL01] Datei befindet, beschrieben. Es wird kein SOAP-Encoding-Schema benutzt. Im Gegensatz dazu das „Encoded“-Format, in dem die Daten im SOAP-Body durch ein Encoding-Schema (z. B. `http://schemas.xmlsoap.org/soap/encoding/`) beschrieben werden. Diese Situation zeigt Beispiel 19 auf Seite 92.

Die einzige Lösung für dieses Problem liegt in der Anpassung der SOAP-Quellen, um einen SOAP-Body ohne Namespace erzeugen zu können. Dazu wurde das Apache-SOAP-API um die Methode `setConnectToDotNet` erweitert. Wird diese für einen aktuellen SOAP-Call mit dem Parameter `true` aufgerufen, wird kein Namespace im SOAP-Body erzeugt. Die dazu nötigen Änderungen finden sich im Anhang B. Eine auf diese Art erzeugte Anfrage mit der entsprechenden Antwort vom Server zeigt Beispiel 20 auf Seite 93. Zwar versteht nun der .NET Server die Anfrage, jedoch kann die Antwort vom Client nicht ordnungsgemäß ausgewertet werden. Dies leitet zu den SOAP-spezifischen Problemen.

Beispiel 19 Inkompatible Anfrage an einen MS .NET SOAP-Server

Anfrage:

```
POST http://www.iemfamily.com/iemfamily/CreditCard.asmx HTTP/1.0
Host: www.iemfamily.com
Content-Type: text/xml; charset=utf-8
Content-Length: 473
SOAPAction: "http://tempuri.org/VerifyCreditCard"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
                    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
     <ns1:VerifyCreditCard xmlns:ns1="http://tempuri.org/"
             SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <CardNumber xsi:type="xsd:string">2323121</CardNumber>
     </ns1:VerifyCreditCard>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Antwort:

```
HTTP/1.1 500 Internal Server Error.
Server: Microsoft-IIS/5.0
Date: Mon, 17 Sep 2001 14:48:56 GMT
MicrosoftOfficeWebServer: 5.0_Pub
Cache-Control: private
Content-Type: text/xml; charset=utf-8
Content-Length: 985

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>System.Web.Services.Protocols.SoapException: Server was unable
       to read request. ---&gt; System.Xml.XmlException: The 'SOAP-ENV:Envelope' s
       tart tag on line '2' does not match the end tag of 'SOAP-ENV:Enve'. Line 8,
       position 3.
       at System.Xml.XmlTextReader.ParseTag()
       at System.Xml.XmlTextReader.ParseBeginTagExpandCharEntities()
       at System.Xml.XmlTextReader.Read()
       at System.Xml.XmlReader.Skip()
       at System.Web.Services.Protocols.SoapServerProtocol.ReadParameters()
       at System.Web.Services.Protocols.SoapServerProtocol.ReadParameters()
       at System.Web.Services.Protocols.WebServiceHandler.Invoke()
       at System.Web.Services.Protocols.WebServiceHandler.CoreProcessRequest()
     </faultstring>
    <detail />
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Beispiel 20 Anfrage nach Erweiterung des SOAP-API

Anfrage:

```
POST /iemfamily/CreditCard.asmx HTTP/1.0
Host: www.iemfamily.com
Content-Type: text/xml; charset=utf-8
Content-Length: 439
SOAPAction: "http://tempuri.org/VerifyCreditCard"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
       <VerifyCreditCard xmlns="http://tempuri.org/"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <CardNumber>4111111111111111</CardNumber>
       </VerifyCreditCard>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Antwort:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 12 Sep 2001 14:54:59 GMT
MicrosoftOfficeWebServer: 5.0_Pub
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 418

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <VerifyCreditCardResponse xmlns="http://tempuri.org/">
      <VerifyCreditCardResult>Valid Visa</VerifyCreditCardResult>
    </VerifyCreditCardResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP-Probleme

Eine Anforderung an jeden durch den Apache-SOAP Client vom Server erhaltenen SOAP-Envelope sind die selbstbeschreibenden Typdefinitionen im entsprechenden Ergebnisparameter. Eine Antwort auf die Anfrage im Beispiel 19 sollte also eigentlich der Antwort in Beispiel 21 entsprechen.

Beispiel 21 Korrekte Antwort auf eine Apache-SOAP Anfrage

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Mon, 17 Sep 2001 14:26:34 GMT
MicrosoftOfficeWebServer: 5.0_Pub
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 624

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
               xmlns:tns="http://tempuri.org/"
               xmlns:types="http://tempuri.org/encodedTypes"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <tns:VerifyCreditCardResponse>
             <VerifyCreditCardResult xsi:type="xsd:string">
                  Invalid Unknown</VerifyCreditCardResult>
        </tns:VerifyCreditCardResponse>
    </soap:Body>
</soap:Envelope>
```

Das Problem bei der Antwort in Beispiel 20 ist nun, dass die SOAP-Engine des Clients den Typ der Ergebnisparameter nicht kennt, um diese in ein Java-Objekt umwandeln zu können. Eine von den Apache-SOAP Entwicklern vorgeschlagene Lösung für dieses Problem zeigt Beispiel 22.

Beispiel 22 Workaround für Deserialisierung der Ergebnisparameter

```
SOAPMappingRegistry smr = new SOAPMappingRegistry ();
StringDeserializer sd = new StringDeserializer ();
smr.mapTypes (Constants.NS_URI_SOAP_ENC, new QName ("http://tempuri.org/",
                                "VerifyCreditCardResult"), String.class, null, sd);
soapCall.setSOAPMappingRegistry (smr);
```

Hier wird das bestehende Encoding um einen Datentyp für den Rückgabeparameter `VerifyCreditCardResult` erweitert. Mit Hilfe des angepassten Apache-SOAP API und der in Beispiel 22 vorgestellten Lösung lässt sich nunmehr ein funktionierender Nachrichtenaustausch wie in Beispiel 20 gezeigt, realisieren. Nachdem das `SOAPDemoPortlet` nun wie erwartet funktioniert, kann die Anwendung um das `CreditCardCheck` Portlet erweitert werden.

Das `CreditCardCheck` Portlet ist dabei lediglich eine Referenz auf das `CustomizerVelocity` Portlet. Dieses ermöglicht unter Hinzunahme des Templates `{JS, H, webapp/WEB-INF/templates/vm/portlets/html/CreditCardCheck.vm}` und eines sogenannten Action-Portlets, in unserem Fall das `{JS, H, src/java/org/apache/jetspeed/modules/actions/portlets/CardCheckAction.java}` Portlet, eine Interaktion mit dem Benutzer. Das entsprechende Portlet und Template findet sich in Anhang B. Das oben erstellte `SOAPDemoPortlet` wird in die `CreditCardAction` importiert. Dies ermöglicht die Nutzung der darin enthaltenen Methode `getCardStatus` mit der zu prüfenden Kre-

ditkartennummer als Parameter. Somit fehlt zur Realisierung der Anwendungsintegration nur noch die Registrierung des neu erstellten Portlets in der {JS, G, webapp/WEB-INF/conf/portlets.xreg}. Sie wird in Beispiel 23 gezeigt.

Beispiel 23 Registrierung CreditCardCheck-Portlet

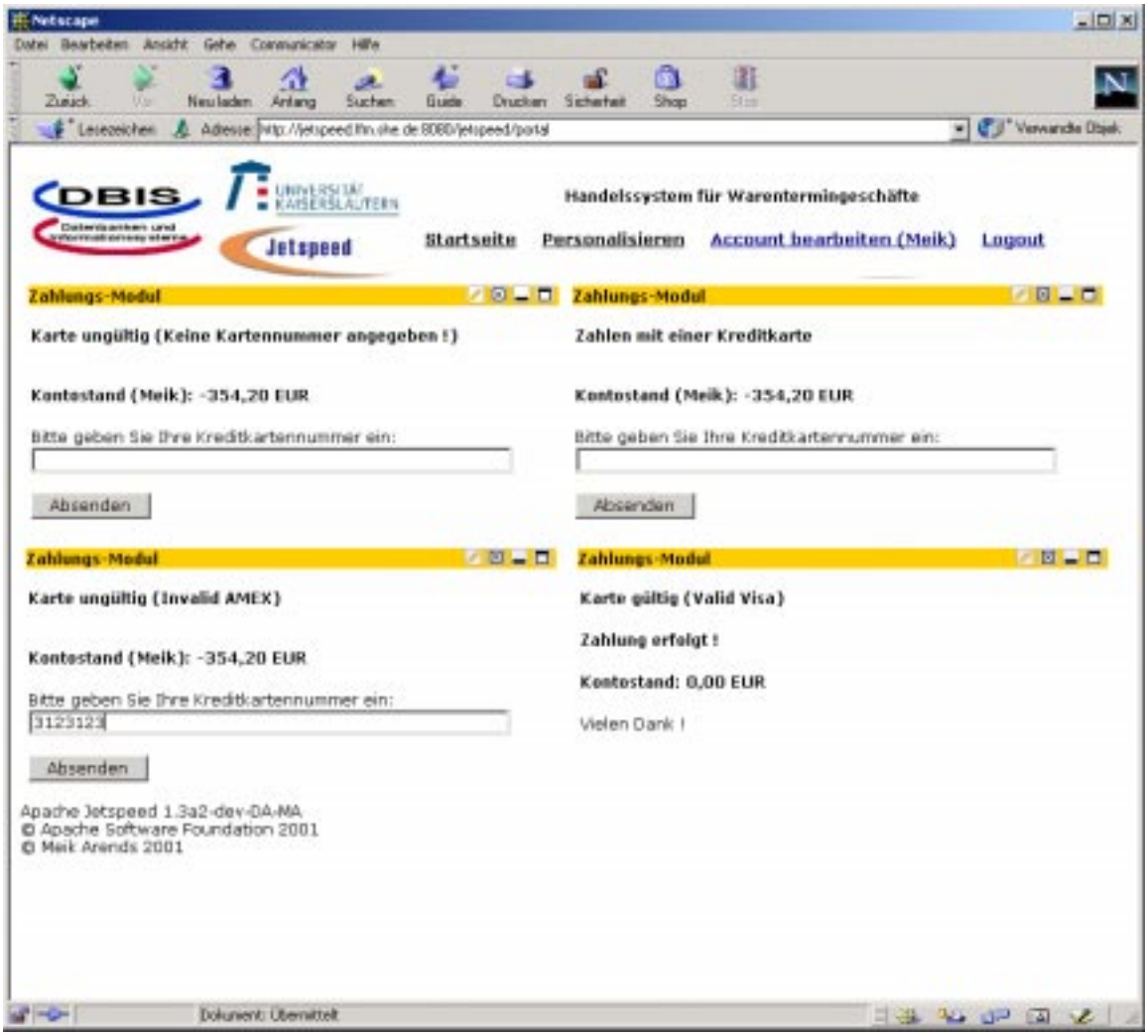
```
<portlet-entry name="CreditCardCheck" hidden="false" type="ref"
        parent="CustomizerVelocity" application="false">
        <meta-info>
            <title>Zahlungs-Modul</title>
            <description>Modul zur Zahlung mit Kreditkarte</description>
        </meta-info>
        <parameter name="template" value="CreditCardCheck" hidden="false"/>
        <parameter name="action" value="portlets.CardCheckAction" hidden="false"/>
        <parameter name="text" value="STATUS: unbekannt" hidden="false"/>
        <media-type ref="html"/>
    </portlet-entry>
```

Abschliessend bleibt noch zu bemerken, dass die Auftretenden Interoperabilitätsprobleme nicht in der SOAP-Spezifikation begründet liegen. Vielmehr ist eine nicht ausreichende oder nicht ordnungsgemässe Implementierung seitens der Client- und Server-Anbieter für diese Probleme verantwortlich.

Weiterhin unterstützt der Betreiber des hier genutzten SOAP-Services mittlerweile auch Zugriffe mit Apache-SOAP Clients. Dazu sind aktuell keine besonderen Vorgehensweisen mehr nötig. Die Problematik wurde jedoch trotzdem dokumentiert, da sie sich in ähnlicher Form jederzeit wiederholen kann. Eine Zusammenstellung des `CreditCardCheck` Portlet in verschiedenen Zuständen zeigt Abbildung 34. So wird bei der Eingabe einer leeren Kreditkartennummer eine entsprechende Warnung generiert. Bei einer ungültigen Kartennummer erfolgt keine Zahlung. Diese ist nur mit einer gültigen Nummer möglich.

In diesem und dem vorigen Kapitel wurde nun eine Beispielsystem realisiert, in dem ein kleiner Teil der Anforderungen an ein Web-basiertes Handelssystems prototypisch implementiert wurde.

Abbildung 34



Kapitel 12 Zusammenfassung und Ausblick

12.1 Zusammenfassung

Diese Arbeit unternimmt den Versuch einen Überblick über den Stand der Entwicklungen im Bereich Portale und Handelsysteme zu geben. Dazu wurde die „Evolution“ von der herkömmlichen Homepage mit statischen Inhalten über Seiten mit dynamischen Inhalten aus Datenbanken, hin zu Portalen und Handelssystemen mit personalisierbaren Inhalten und integrierten Anwendungen untersucht.

Die Client-Server-Architektur des World-Wide-Web wurde zu diesem Zweck näher betrachtet und anhand von Beispielen verschiedene Anforderungen an diese Architektur vorgestellt. Als Möglichkeit zur Realisierung dieser Anforderungen wurde das Stufenmodell zur Datenbankanbindung eingeführt. Wachsende Anforderungen an die Geschwindigkeit der Systeme und eine immer komplexer werdende Logik erforderten jedoch ein mächtigeres Modell zur Beschreibung solcher Systeme. Als Alternative wurde das Schichtenmodell vorgestellt, mit dem sich auch komplexere Systeme beschreiben lassen. Eine Systembeschreibung in Form des Schichtenmodells wurde an Hand eines Business Information Systems (BIS) demonstriert.

Der Trend zu Portalen und Handelssystemen im Web brachte viele verschiedene Definitionen dieser Begriffe hervor. Daher wurden einige dieser Definitionen zusammengetragen um deren Aspekte in eigenen Definitionen kurz und prägnat zusammenfassen zu können. Weiterhin wurden Portale am Beispiel konkreter Realisierungen in die Bereiche Internet-Portal, Intranet-Portal und Vortal klassifiziert. Als eine mögliche Gliederung der Anwendungsfelder für Portale wurden die Begriffe „Business to Customer (B2C)“, „Business to Business (B2B)“ sowie „Business to Employe (B2E)“ eingeführt.

Um funktionale und nicht funktionale Anforderungen an Portal- und Handelssysteme überblikken zu können, wurden gängige Portalprodukte untersucht. Als wichtige Stichworte funktionaler Anforderungen seien hier „Personalisierung“, „Datenintegration“, „Anwendungsintegration“, „Navigation“, „Notifikation“, „Multi-Channel-Delivery“ sowie „Wissensmanagement“ und „Workflow“ genannt. Weiterhin wurde die Integration und Personalisierung mit Portlets angesprochen. Dazu wurde die Bedeutung des Portlet-Begriffs aus verschiedenen Gesichtspunkten dargestellt. Die von den untersuchten Produkten erbrachten funktionalen und nicht funktionalen Anforderungen wurden in tabellarischer Form gegenübergestellt und den möglichen Anwendungsfeldern zugeordnet.

Nach der Identifikation der Anforderungen wurden die Architekturen der betrachteten Portalprodukte analysiert. Ziel war die Erstellung einer verallgemeinerten Portalarchitektur zur optimalen Realisierung der erkannten funktionalen und nicht funktionalen Anforderungen. Als Ergebnis

entstand eine Schichtenarchitektur welche eine Trennung von Daten-, Anwendungs- und Präsentationsschicht vornimmt und zusätzlich einen Bereich für die Anwendungsintegration enthält. Weiterhin wurde festgestellt, dass sich viele der Anforderungen durch eine Separierung der Daten von ihrer Repräsentation einfacher realisieren lassen.

Dieser Gedanke wurde in der näher betrachteten Architekur des Open-Source Portals Jetspeed aufgenommen. Bei der Realisierung dieses Portals kam das „Model-View-Controller (MVC)“ Design-Pattern zum Einsatz, bei dem eine strikte Trennung von Daten, Logik und Repräsentation stattfindet. Zusätzlich wurden die Einzelkomponenten, aus denen sich Jetspeed zusammensetzt, näher vorgestellt, um nach einer Analyse des Informationsflusses durch das Portal eine detailierte Architekturübersicht von Jetspeed geben zu können. Weiterhin wurde das von Jetspeed genutze Portlet-Konzept aufgegriffen und das Verständnis des Portlet-Begriffs innerhalb von Jetspeed dargestellt.

Anschliessend wurde die auf XML basierende, zur Konfiguration und Personalisierung von Jetspeed verwendete „Portlet Structure Markup Language (PSML)“ vorgestellt und deren Einsatzmöglichkeiten in Form von Beispielen demonstriert. So wird mit Hilfe von PSML die Registrierung der in Jetspeed vorhandenen Portlets durchgeführt und die benutzerspezifischen zur Personalisierung erforderlichen Informationen abgelegt. Eine grafische UML-Repräsentation des bei PSML verwendeten Schemas zur Konfiguration und Personalisierung erleichtert dabei den Überblick.

Um mit Jetspeed eine prototypische Portal-Implementierung durchführen zu können, wurde eine Anforderungsanalyse an ein Web-basiertes Handelssystem für Warentermingeschäfte durchgeführt. Nach der Prüfung der möglichen Prozesse innerhalb eines solchen Handelssystems wurden diese in einer Architektur dargestellt.

Da einige für das Handelssystem notwendigen Prozesse eine Anwendungsintegration erfordern, wurde die Rolle der Anwendungsintegration im entsprechenden Umfeld untersucht. Neben allgemeinen Aspekten wurden Wege und Mechanismen aufgezeigt, die eine Anwendungsintegration ermöglichen sollen. So wurden beispielsweise nachrichtenbasierte Ansätze (Message-Oriented Middleware, MOM) mit einer asynchronen Übertragung der Informationen und Funktions-Schnittstellen (Distributed-Object Middleware) mit synchronem Informationsaustausch vorgestellt. Weiterhin wurde auf die bei der Anwendungsintegration eventuell auftretende Probleme hingewiesen und Lösungsmöglichkeiten aufgezeigt.

Da das zur prototypischen Realisierung des Handelssystems ausgesuchte Jetspeed-Portal keine Mechanismen zur Anwendungsintegration bereitstellt, wurde mit dem „Simple Object Access Protocol (SOAP)“ eine Möglichkeit zum Nachrichtenaustausch zwischen Anwendungen gezeigt. SOAP ist ein Protokoll zur Durchführung von „Remote Procedure Calls (RPC)“ und der damit verbundenen Übermittlung von Nachrichten bei verteilten Systemen. Dabei kommt XML zur Formulierung der auszutauschenden Nachrichten zum Einsatz. Als Transportprotokoll wird derzeit entweder HTTP oder SMTP eingesetzt. Der Nachrichtenaustausch zwischen zwei Systemen wurde unter Benutzung von HTTP als Transportprotokoll demonstriert.

Abschliessend erfolgte dann eine prototypische Implementierung des vorgestellten Handelssystems, indem einige der daran gestellten Anforderungen umgesetzt wurden. Das Anlegen von Benutzern, die Suche im Web, eine grafisch aufbereitete Ausgabe von Marktinformationen sowie das Einbinden aktueller Wirtschaftsnachrichten wurden im ersten Schritt umgesetzt. Dabei traten

keine grösseren Probleme auf. Im zweiten Schritt wurde, als Beispiel für eine Anwendungsintegration, ein Zahlungsmodul entwickelt. Dieses Modul sendet eine Anfrage zur Prüfung der Gültigkeit einer Kreditkarte mit Hilfe des SOAP-Protokolls an einen dafür vorgesehenen SOAP-Server und wertet dessen Antwort aus. Die Implementierung erwies sich auf Grund auftretender Interoperatiilitätsprobleme zwischen verschiedenen SOAP-Systemen als schwierig und erforderte einen Eingriff in die aktuelle Apache-SOAP Version. Die Probleme wurden weiter analysiert und Ansätze zu deren Lösung erarbeitet.

Zusammenfassend lässt sich eine positive Bilanz zum Einsatz von Jetspeed zur Realisierung eines Portal-Systems ziehen. Ein erfahrender Programmierer kann trotz Mangel an aktueller Dokumentation in relativ kurzer Zeit ein passables Portal erstellen. Als gewichtiges Manko stellte sich jedoch das geringe Leistungsverhalten bezüglich der Geschwindigkeit durch den Einsatz von Java heraus.

12.2 Ausblick

Es ist festzustellen, dass die Ausbreitung von Portalen ähnlich der Ausbreitung von einfachen statischen Hompages stattfindet. Reichte vor kurzem noch die allgemeine Präsenz im Internet, so gerät mittlerweile die Erweiterung auf Portal-Funktionalität fast zum Muss, um mit konkurrierenden Anbietern mithalten zu können. Die damit oft verbundene Anwendungsintegration erfordert ensprechende Mechanismen. In dieser Arbeit wurde sowohl der komponentenbasierte Ansatz (z. B. CORBA) als auch der Dokumenten- oder Nachrichten-basierte Ansatz (z. B. SOAP) betrachtet.

Eine weiterführende Idee ist die Kombination der Vorteile beider Ansätze in einem komplementären (zusammenführenden) Ansatz [HeRo01]. Um diesen Ansatz näher ausführen zu können, wird im Folgenden die Situation angenommen, dass ein Dienst von zwei unterschiedlichen Dienstnutzern in Anspruch genommen wird. Auf der einen Seite die Web-Clients, auf der anderen Seite die Applikations-Clients. Für die erste Gruppe sieht der Dienstanbieter eine auf HTML basierende Web-Lösung vor. Diese Gruppe von Nutzern hat lediglich einen Ad-hoc-Bedarf an den vom Dienstanbieter angebotenen Informationen. Die zweite Gruppe hat regelmäßigen Bedarf an entsprechenden Informationsangeboten und sieht eine Speicherung der erhaltenen Resultate in ihrer Informationsverarbeitung vor. Hier sieht der Dienstanbieter eine komponentenbasierte Lösung, in diesem Fall EJB [GrTh00], vor. Ein Vorteil dieser Anbindung liegt in den feingranularen Diensten. Jedoch bedingt der bei allen komponentenbasierten Ansätzen zur Kommunikation genutzte Stub/Skeleton-Mechanismus [GrTh00] auf der Dienstnutzerseite einen gewissen Aufwand, um die Kommunikation zu ermöglichen. Die alleinige Nutzung der bestehenden Web-Infrastruktur reicht nicht aus.

Im Vergleich dazu liefert die XML-gestützte Kommunikation die Möglichkeit einer raschen Umsetzung. Die bestehende Web-Infrastruktur gelangt unmittelbar zum Einsatz. Der Dienstnutzer muss lediglich ein Anwendungsprogramm für das Parsen der XML-Dokumente entwickeln. Die Granularität der Dienste ist in diesem Fall grob, das heißt der Dienstnutzer muss sich die Informationen aus einer Fülle von Daten extrahieren, da immer vollständige Dokumente übertragen werden.

Eine Möglichkeit der Zusammenführung besteht nun darin, den komponentenbasierten Ansatz auf Dienstanbieterseite um die Möglichkeit des Datenaustausches über XML-Dokumente zu erweitern. Die Realisierung dieser Variante sieht den Einsatz von Servlets vor, welche als Client des Applikation-Servers bzw. der dortigen Objekte und Methoden fungieren. Diese nehmen die vom Dienstnutzer abgesetzten Anfragen entgegen und führen daraufhin entsprechende Methoden auf dem Applikation-Server aus. Das Resultat des Methodenaufrufs muss dann noch nach XML umgewandelt werden. Dazu kann eine sogenannte Fassade [Gam96] eingesetzt werden, die sich beispielsweise als Stateless Session-Bean realisieren lässt. Diese Fassade verfügt dann über eigene Methoden zur Umwandlung der Ergebnisse in XML. Findet der Zugang wie bislang innerhalb eines komponentenbasierten Ansatzes statt, so nutzt der Dienstnutzer die vom Applikation-Server angebotene Methode direkt.

Eine weitere Tendenz ist die Nutzung von sogenannten Web-Services. Ein Beispiel dafür ist der in Kapitel 11 vorgestellte Service zur Überprüfung von Kreditkarten auf ihre Gültigkeit. Ziel dieser Services ist die direkte Nutzung der auf einem Web-Server zur Verfügung gestellten Informationen ohne diese mühsam aus den mit den Layout vermischten HTML-Dateien extrahieren zu müssen [Wiel01]. Als Protokoll für den Methodenaufruf eines solchen Web-Service scheint sich SOAP durchzusetzen. Ein Aufrufprotokoll alleine reicht jedoch nicht aus, es muss auch eine Terminologie zur Beschreibung der Dienste bereit gestellt werden. Ansonsten kann es, wie bei der Realisierung des Zahlungsmoduls, zu Interoperabilitätsproblemen kommen. Zu diesem Zweck wurde die „Web Service Description Language (WSDL)“ entwickelt, mit deren Hilfe man den Client über die zur verfügungstehenden Methoden und benötigten Parameter informieren kann. Leider unterstützt der in Kapitel 11 genutze Apache-SOAP Client diesen neuen Ansatz noch nicht. Zusammenfassend gilt WSDL aber als Schritt in die richtige Richtung, um Interoperatibilitätsprobleme auf ein Minimum reduzieren zu können. Eine Ausführlichere Betrachtung von WSDL findet man in [Wiel01].

Anhang A Installationsanleitungen

A.1 Jetspeed unter Linux

Die Installation unter Linux soll als Servlet-Engine und Web-Server Jakarta-Tomcat nutzen. Alternativ besteht u.a. auch die Möglichkeit Tomcat durch den Apache-Web-Server mit Servlet-Erweiterung zu ersetzen. Dies ist dann zu empfehlen, wenn eine höhere Performance benötigt wird.

A.1.1 Downloads

Mit den hier referenzierten Versionen funktionierte die Installation in einer auf Debian-Linux[1] basierenden Testumgebung ohne Probleme. Es steht frei andere Versionen zu nutzen. Für die Installation unter Linux müssen erst folgende Komponenten heruntergeladen werden:

Tomcat 4.0beta7:

```
http://jakarta.apache.org/builds/jakarta-tomcat-4.0/
                    release/v4.0-b7/jakarta-tomcat-4.0-b7.tar.gz
```

Aktuelle Jetspeed-Release:

```
http://jakarta.apache.org/builds/jakarta-jetspeed/release/
```

Dort befindet sich jeweils die aktuelle Release als übersetztes WAR-Archiv oder das entsprechende Source-Archiv.

Soll die aktuellste Version aus dem Jetspeed-CVS genutzt werden muss diese aus dem Repository ausgelesen werden. Die geschieht mit dem Linux-Befehl `cvs`. Unter Umständen muss, je nach verwendeter Linux-Distribution, das ensprechende CVS-Packet extra installiert werden.

Anmeldung am CVS-Server:

```
> cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic login
```

Als Passwort ist `anoncvs` zu verwenden.

Checkout der aktuellen CVS-Version:

```
> cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic
                checkout jakarta-jetspeed
```

1. http://www.debian.de

Wurde das Archiv mit den Quellen oder die CVS-Version geladen muss Jetspeed erst übersetzt werden. Hier wird ein entsprechender Java-SDK benötigt, der auch zum Betrieb der Servlet-Engine notwendig ist.

Java 2 Standard Edition 1.4 beta2:

`http://java.sun.com/j2se/1.4/#linux`

A.1.2 Installation

Zuerst wird der Java-SDK installiert. Anschliessend muss eine Umgebungsvariable gesetzt werden. Die Installation aller erforderlicher Software soll im Folgenden in das Verzeichnis `/usr/local/jetspeed` erfolgen.

Dazu in das entsprechende Verzeichnis wechseln und den Java-SDK extrahieren:

```
> cd /usr/local/jetspeed
> sh <Pfad zum Java SDK>/j2sdk-1_4_0-beta2-linux-i386.bin
```

Die Umgebungsvariable JAVA_HOME ist dann auf `/usr/local/jetspeed/j2sdk1.4.0` zu setzen. Je nach benutztem Kommandozeileninterpreter (Shell) erfolgt dies mit den folgenden Kommandos.

tcsh: `> setenv JAVA_HOME /usr/local/jetspeed/j2sdk1.4.0`

bash: `> export JAVA_HOME=/usr/local/jetspeed/j2sdk1.4.0`

Wurde die vorkompilierte Version von Jetspeed heruntergeladen, kann das WAR[1]-Archiv nun in den entsprechenden Ordner der Servlet-Engine extrahiert werden.

```
> cd /tmp
> tar xzvf <Pfad zur Jetspeed-Release>/Jetspeed-1.3a1-war.tar.gz
> mv Jetspeed-1.3a1/jetspeed.war
                /usr/local/jetspeed/jakarta-tomcat-4.0-b7/webapps
```

Ansonsten müssen die Jetspeed-Quellen in ein WAR-Archiv übersetzt werden. Hier wird von den CVS-Sourcen ausgegangen.

```
> cd <Pfad zu den Sourcen>/jakarta-jetspeed/build
> ./build.sh war
```

Nach erfolgreicher Übersetzung muss auch hier das WAR-Archiv in den entsprechenden Ordner der Servlet-Engine verschoben werden.

```
> mv <Pfad zu den Sourcen>/jakarta-jetspeed/bin/jetspeed.war
                /usr/local/jetspeed/jakarta-tomcat-4.0-b7/webapps
```

Nun kann die Servlet-Engine gestartet werden. Diese extrahiert zuerst das WAR-Archiv. Nach einiger Zeit steht dann Jetspeed zur Verfügung.

1. Das WAR-Archiv kann einfach in das entsprechende Verzeichnis einer Servlet-Engine kopiert werden. Diese extrahiert das Archiv dann automaitisch.

```
> cd /usr/local/jetspeed/jakarta-tomcat-4.0-b7/bin
> ./startup.sh
```

Der Zugriff auf Jetspeed erfolgt dann mit `http://localhost:8080/jetspeed`.

Bei Problemen mit der Installation von Tomcat oder Jetspeed findet man in entsprechenden Mailing-Listen [TMA01] und auf den Projekt-Homepages [Tom01][Jet01] Hilfe.

A.2 Installation von Apache SOAP

Zur Installation von SOAP bestehen zwei Möglichkeiten. Zum einen kann eine vorkompilierte Version genutzt werden oder man benutzt die Java-Quellen um eine eigene Übersetzung vorzunehmen. Da Änderungen an den SOAP-Quellen notwendig sind, ist jedoch eine eigene Übersetzung erforderlich. Zusätzlich muss bei der Installation unterschieden werden, ob Apache-SOAP lediglich für Client-Zugriffe genutzt werden soll oder ob eigene SOAP-Dienste angeboten werden. Hier wird lediglich die Installation für den Client-Zugriff auf bestehende SOAP-Server durchgeführt.

A.2.1 Downloads

Zuerst müssen also die Quellen von Apache-SOAP heruntergeladen werden. Hier wurde die Version 2.2 genutzt. Zum Übersetzen wird weiterhin das Tool Ant, welches ebenfalls zum Jakarta-Projekt gehört, benötigt. Dieses kommt auch beim Übersetzen von Jetspeed zum Einsatz, ist dort aber bereits in den Quellen enthalten.

Apache SOAP 2.2:

`http://xml.apache.org/dist/soap/version-2.2/soap-src-2.2.tar.gz`

Ant 1.4:

```
http://jakarta.apache.org/builds/jakarta-ant/release/v1.4/bin/
                    jakarta-ant-1.4-bin.tar.gz
```

A.2.2 Installation

Zuerst die SOAP-Quellen extrahieren, in das Verzeichnis wechseln und darin Ant extrahieren.

```
> tar xzf <Pfad zu den SOAP-Sourcen>/soap-src-2.2.tar.gz
> cd soap-2_2
> tar xzf <Pfad zu Ant>/jakarta-ant-1.4-bin.tar.gz
```

Zum Übersetzen sind noch weitere Aktionen erforderlich.

```
> ln -s jakarta-ant-1.4/bin .
> ln -s jakarta-ant-1.4/lib .
> cp <Pfad zu den Jetspeed-Sourcen>/build/build.sh .
> cp <Pfad zu den Jetspeed-Sourcen>/lib/mail.jar ./lib
> cp <Pfad zu den Jetspeed-Sourcen>/lib/activation.jar ./lib
```

```
> cp <Pfad zu den Jetspeed-Sourcen>/lib/servlet_2_2.jar ./lib
```

In der `build.sh` muss der Eintrag für `PROJECTDIR` von „`..`“ auf „`.`“abgeändert werden. Aufgrund der in Kapitel 11 beschriebenen Interoperatibilitätsprobleme müssen vor der Übersetzung noch zwei Dateien ersetzt werden. Die Änderungen finden sich im Anhang B in Form eines Patches oder können unter

```
http://www.arends-online.de/DA/soap-2_2-patches.tar.gz
```

heruntergeladen werden.

Die abgeänderten Dateien extrahieren und an die richtige Stelle kopieren.

```
> tar xzf soap-2_2-patches.tar.gz
> cp RPCMessage.java src/org/apache/soap/rpc
> cp SoapEncUtils.java src/org/apache/soap/encoding/soapenc
```

Nun kann SOAP mit `> ./build.sh compile` übersetzt werden und das Java-Archiv an die entsprechende Stelle der Jetspeed-Installation und des Webservers kopiert werden.

```
> cp build/lib/soap.jar <Pfad zu den Jetspeed-Sourcen>/lib
> cp build/lib/soap.jar <Pfad zur Turbine-Installation>/common/
                lib/soap.jar
```

Danach Jetspeed neu übersetzen und wieder in Tomcat installieren. Nun sollte es allen Portles möglich sein SOAP-Dienste zu nutzen. Sie müssen dazu lediglich die entsprechenden Klassen importieren. In der Regel reichen die folgenden Klassen aus:

```
import org.apache.soap.*;
import org.apache.soap.rpc.*;
import org.apache.soap.util.xml.*;
import org.apache.soap.encoding.*;
import org.apache.soap.encoding.soapenc.*;
import org.apache.soap.transport.*;
import org.apache.soap.transport.http.SOAPHTTPConnection;
```

Soll Apache-SOAP auch als SOAP-Server eingesetzt weden, muss auch Xerces[1] innerhalb der Jetspeed-Installation auf den aktuellsten Stand gebracht werden. Die aktuelle Version ist derzeit 1.4.3.

1. http://xml.apache.org/xerces-j/index.html

Anhang B Weitere Informationen zu Jetspeed

B.1 Quellen

B.1.1 Veränderte Dateien

webapp/WEB-INF/conf/portlets.xreg

```
<?xml version="1.0" encoding="UTF-8"?>
<registry>
    <portlet-entry name="XSL" hidden="false" type="abstract" application="false">
        <classname>org.apache.jetspeed.portal.portlets.XSLPortlet</classname>
    </portlet-entry>

    <portlet-entry name="RSS" hidden="false" type="abstract" application="false">
        <classname>org.apache.jetspeed.portal.portlets.NewRSSPortlet</classname>
        <parameter name="stylesheet" value="/WEB-INF/xsl/rss.xsl" hidden="false">
            <security role="admin"/>
        </parameter>
        <parameter name="stylesheet.text/html"
            value="/WEB-INF/xsl/rss.xsl" hidden="false">
            <security role="admin"/>
        </parameter>
        <parameter name="stylesheet.text/vnd.wap.wml"
            value="/WEB-INF/xsl/rss-wml.xsl" hidden="false">
            <security role="admin"/>
        </parameter>
        <parameter name="itemdisplayed" value="15" hidden="false">
            <meta-info>
                <title>Number of items</title>
                <description>This parameter controls how many items may
                    be shown in this            portlet. If the
                    information channel has more items, only the first
                    will           be displayed.         </description>
            </meta-info>
        </parameter>
        <parameter name="showdescription" value="true" type="boolean" hidden="false">
            <meta-info>
                <title>Show item description ?</title>
                <description>          This parameter specify whether
                    the item descriptions and icons should be
                    displayed in this portlet, or only the headlines.         </description>
            </meta-info>
        </parameter>
```

```
   <parameter name="showtitle" value="true" type="boolean" hidden="false">
        <meta-info>
            <title>Show title description ?</title>
            <description>          This parameter specify whether
                the title description and icon should be
                displayed in this portlet, or only the title.        </description>
        </meta-info>
    </parameter>
    <parameter name="showtextinput" value="true" type="boolean" hidden="false">
        <meta-info>
            <title>Show Text Input?</title>
            <description>          This parameter specify whether
                the text input will be displayed.            Text
                Input is an optional element in the RSS feed.        </description>
        </meta-info>
    </parameter>
    <media-type ref="html"/>
    <media-type ref="wml"/>
</portlet-entry>

<portlet-entry name="WML" hidden="false" type="abstract" application="false">
    <classname>org.apache.jetspeed.portal.portlets.WMLFilePortlet</classname>
    <media-type ref="wml"/>
</portlet-entry>

<portlet-entry name="Velocity" hidden="false" type="abstract" application="false">
    <classname>org.apache.jetspeed.portal.portlets.VelocityPortlet</classname>
</portlet-entry>

<portlet-entry name="JSP" hidden="false" type="abstract" application="false">
    <classname>org.apache.jetspeed.portal.portlets.JspPortlet</classname>
</portlet-entry>

<portlet-entry name="CustomizerVelocity" hidden="false"
    type="abstract" application="false">
    <classname>org.apache.jetspeed.portal.portlets.CustomizerVelocityPortlet</classname>
</portlet-entry>

<portlet-entry name="HTML" hidden="false" type="abstract" application="false">
    <classname>org.apache.jetspeed.portal.portlets.FileServerPortlet</classname>
    <media-type ref="html"/>
</portlet-entry>

<portlet-entry name="WebPagePortlet" hidden="false" type="abstract" application="false">
    <classname>org.apache.jetspeed.portal.portlets.WebPagePortlet</classname>
    <media-type ref="html"/>
</portlet-entry>

<portlet-entry name="ClearPortlet" hidden="true" type="instance" application="false">
    <classname>org.apache.jetspeed.portal.portlets.ClearPortlet</classname>
    <media-type ref="html"/>
    <media-type ref="wml"/>
</portlet-entry>

<portlet-entry name="CreditCardCheck" hidden="false" type="ref"
    parent="CustomizerVelocity" application="false">
    <meta-info>
        <title>Zahlungs-Modul</title>
        <description>Modul zur Zahlung mit Kreditkarte</description>
    </meta-info>
    <parameter name="template" value="CreditCardCheck" hidden="false"/>
    <parameter name="action" value="portlets.CardCheckAction" hidden="false"/>
    <parameter name="text" value="STATUS: unbekannt" hidden="false"/>
   <media-type ref="html"/>
</portlet-entry>
```

```
<portlet-entry name="NTPortlet" hidden="false" type="ref" parent="RSS" application="false">
        <meta-info>
            <title>Newsticker</title>
    <description>Aktuelle Business-News</description>
        </meta-info>
        <url>http://headlines.internet.com/internetnews/bus-news/news.rss</url>
    </portlet-entry>
<portlet-entry name="Search" hidden="false" type="ref" parent="HTML" application="false">
        <meta-info>
            <title>Suche im Web</title>
            <description>Suche mit Yahoo</description>
        </meta-info>
        <url>/search/index.html</url>
    </portlet-entry>
</registry>
```

webapp/WEB-INF/psml/anon/html/?/default.psml

```
<?xml version="1.0" encoding="UTF-8"?>
<portlets xmlns="http://www.apache.org/2000/02/CVS">
    <controller name="RowController"/>
    <skin name="orange-grey"/>
    <entry parent="DAWelcome"/>
</portlets>
```

webapp/WEB-INF/templates/vm/layouts/html/default.vm

```
<html>
  <head>
    <base href="$clink.External" />
    <link href="$clink.setURI(„css/default.css").Absolute" type="text/css" rel="stylesheet" />
  </head>
  <body bgcolor="#ffffff">
    <table cellspacing="0" width="100%" border="0" cellpadding="0">
      <tr>
        <td><img src="$clink.setURI(„images/da-logo.jpg").Absolute" /></td>
        <td>$navigation.setTemplate(„html/top.vm")</td>
      </tr>
    </table>
    <table cellspacing="0" width="100%" cellpadding="0" border="0">
      <tr>
        <td valign="top" bgcolor="#ffffff">
          $navigation.setTemplate(„html/left.vm")
        </td>
        <td valign="top">
          $screen_placeholder
        </td>
      </tr>
    </table>
    <table cellspacing="0" width="100%" cellpadding="0" border="0">
      <tr>
        <td valign="bottom" bgcolor="#ffffff">
          $navigation.setTemplate(„html/bottom.vm")
        </td>
      </tr>
    </table>
  </body>
</html>
```

webapp/WEB-INF/templates/vm/navigations/html/left.vm

```
<table width="100%" border="0" cellspacing="0" cellpadding="2">
  <tr>
    <td>
      <!--<a href="$link">
        <small>Home</small>
      </a> -->
    </td>
  </tr>
  <tr>
    <td>
      <!-- <a href="$clink.setURI(„docs/")">
        <small>Documentation</small>
      </a> -->
    </td>
  </tr>
  <tr>
    <td>
      <!-- <a href="$clink.setURI(„apidocs/")">
        <small>API</small>
      </a> -->
    </td>
  </tr>
</table>
```

webapp/WEB-INF/templates/vm/navigations/html/bottom.vm

```
<p>
  <!-- <img align="right" src="$clink.setURI(„images/feather.gif")" /> -->
  <font size="-2">
    Apache Jetspeed 1.3a2-dev-DA-MA<br />
    &copy; Apache Software Foundation 2001<br />
    &copy; Meik Arends 2001
  </font>
</p>
```

webapp/WEB-INF/templates/vm/navigations/html/top.vm

```
<div>
#if ($data.User.hasLoggedIn())
<table cellspacing="2" border="0">
<tr>
  <td align="center">
  <h2>Handelssystem f&uuml;r Warentermingesch&auml;fte  </h2><a href="$link">Startseite
  </a>
     
  <!-- this is a temporary link while the profile customization is not in place -->
  <a href="$link.setAction(„controls.Customize")">Personalisieren</a>
     
  <a href="$link.setPage(„EditAccount")">Account bearbeiten ($data.User.UserName)</a>
     
  <a href="$link.setAction(„LogoutUser")">Logout</a>
     
  </td>
</tr>
</table>
#else
<table cellspacing="2" border="0">
<tr>
  <td valign="middle">
    <h2>Handelssystem f&uuml;r Warentermingesch&auml;fte  </h2><a href="$link">Startseite
   </a><br />
    <small><a href="$link.setPage(„NewAccount")">Neuen Benutzer anlegen</a></small>
  </td>
  <td valign="middle">
```

```
    <form method="POST" action="$link" enctype="application/x-www-form-urlencoded">
    <input name="$jlink.ActionKey" type="hidden" value="$config.getString(„action.login“)" />
    <table border="0" cellspacing="2" cellpadding="1">
      <tr>
        <td><small>Username:</small></td>
        <td><small><input size="10" value="$!data.Parameters.getString(„username“)" name="username"
          maxlength="25" type="text" /></small></td>
      </tr>
      <tr>
        <td><small>Password:</small></td>
        <td><small><input size="10" value="„ name="password" maxlength="25" type="password" />
          </small></td>
      </tr>
      <tr>
        <td colspan="2" align="center">
          <small><input name="submit" type="submit" value="Login" /></small>
        </td>
      </tr>
    </table>
    </form>
  </td>
</tr>
</table>
#end
</div>
```

webapp/WEB-INF/templates/vm/screens/html/NewAccount.vm

```
<div align="left">
  <form accept-charset="UTF-8, ISO-8859-1"
        method="POST"
        action="$link.setPage(„ConfirmRegistration“)"
        enctype="application/x-www-form-urlencoded"
       name = „newuser“>
    <input name="$jlink.ActionKey" type="hidden" value="CreateNewUserAndConfirm" />
    <table cellpadding="4" cellspacing="0" border="0">
    <tr>
      <td>
        <h2>Neuen Benutzer anlegen</h2>
        <p>
          Um einen neuen Benutzer anzulegen bitte alle Felder ausfüllen <br />
          und die Allgemeinen Gesch&auml;ftsbedingungen akzeptieren. <br />
         Nach dem Anlegen des Accounts erhalten Sie eine Email mit einem Sicherheits-Key. <br />
          Diesen Key m&uuml;ssen Sie im n&auml;chsten Schritt zur Best&auml;tigung Ihrer Anmeldung
         eingeben. <br />
          <b>Daher wird eine gültige Email-Adresse ben&ouml;tigt.</b>
        </p>
#if ($data.Message)
        <b>$data.Message</b>
        <p />
#end
        <table cellpadding="0" cellspacing="0" border="0">
          <tr>
            <td> Benutzername: </td>
            <td> <input name="username" type="TEXT" value="$!data.Parameters.getString(„username““/>
          </td>
          </tr>
          <tr>
            <td> Passwort: </td>
            <td> <input name="password" type="PASSWORD" value="„ /> </td>
          </tr>
          <tr>
            <td> Passwort (Wiederholung): </td>
            <td> <input name="password_confirm" type="PASSWORD" value="„ /> </td>
          </tr>
          <tr>
            <td colspan="2"> <hr size="1" noshade /> </td>
          </tr>
```

```
          <tr>
            <td> Vorname: </td>
           <td><input name="firstname" type="TEXT"
                value="$!data.Parameters.getString(„firstname")"/> </td>
          </tr>
          <tr>
            <td> Nachname: </td>
            <td> <input name="lastname" type="TEXT"
                value="$!data.Parameters.getString(„lastname")" /> </td>
          </tr>
          <tr>
            <td> Email: </td>
           <td> <input name="email" type="TEXT" value="$!data.Parameters.getString(„email")" /> </td>
          </tr>
  <tr>
     <td> </td>
     <td><input type="checkbox" name="agb" value="BLA">Ich habe die <a href="#">AGB</a> gelesen und
         akzeptiert. </td>
  </tr>
  <tr>
    <td heigth="30"> </td>
  </tr>
          <tr>
            <td colspan="2" align="right">
              <input name="submit1" type="reset"  value="Zur&uuml;cksetzen" />
               
              <input name="submit2" type="submit" value="Benutzer anlegen"
                onClick="javascript: if (!document.forms.newuser.agb.checked) { alert(,Sie haben die
                AGB\nnicht akzeptiert !'); return false; }" />
            </td>
          </tr>
        </table>
      </td>
    </table>
  </form>
</div>
```

webapp/ocs/local.ocs

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
  xmlns:rdf = „http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ocs = „http://alchemy.openjava.org/ocs/ocs-syntax#"
  xmlns:dc  = „http://purl.org/dc/elements/1.0/">
  <rdf:description about="„>
    <dc:title>My Jetspeed OCS Feed</dc:title>
    <dc:creator>Raphaël Luta (mailto:raphael@apache.org)</dc:creator>
    <dc:description>This is a sample OCS channel to be extended by your own channel
        descriptions</dc:description>
   </rdf:description>
</rdf:RDF>
```

webapp/rss/Jetspeed.rss sowie webapp/rss/Jetspeed.rss

```
<?xml version="1.0"?>
<rss version="0.91">
</rss>
```

B.1.2 Hinzugefügte Dateien

src/java/org/apache/jetspeed/modules/actions/portlets/CardCheckAction.java

```
package org.apache.jetspeed.modules.actions.portlets;

import org.apache.jetspeed.portal.portlets.VelocityPortlet;
import org.apache.jetspeed.portal.portlets.SOAPDemoPortlet;

// Turbine stuff
import org.apache.turbine.util.Log;
import org.apache.turbine.util.RunData;

// Velocity Stuff
import org.apache.velocity.context.Context;

import org.apache.soap.util.net.*;

/**
 * An abstract action class to build VelocityPortlet actions.
 *
 * <p>Don‘t call it from the URL, the Portlet and the Action are automatically
 * associated through the registry PortletName
 *
 * @author <a href=“mailto:raphael@apache.org“>Raphaël Luta</a>
 */
public class CardCheckAction extends VelocityPortletAction
{

    /**
     * Subclasses should override this method if they wish to
     * build specific content when maximized. Default behavior is
     * to do the same as normal content.
     */
    protected void buildMaximizedContext( VelocityPortlet portlet,
                                          Context context,
                                          RunData rundata )
    {
        buildNormalContext( portlet, context, rundata);

        String text = (String)context.get(„text“);
       context.put(„text“, text);
    }

    /**
     * Subclasses should override this method if they wish to
     * provide their own customization behavior.
     * Default is to use Portal base customizer action
     */
    protected void buildConfigureContext( VelocityPortlet portlet,
                                          Context context,
                                          RunData rundata )
    {

        buildNormalContext( portlet, context, rundata);

        setTemplate(rundata, „CreditCardCheck“);

    }

    /**
     * Subclasses must override this method to provide default behavior
     * for the portlet action
     */
```

```
   protected void buildNormalContext( VelocityPortlet portlet,
                                      Context context,
                                      RunData rundata )
    {
        context.put(„text“, (String)context.get(„text“) );
    }

    public void doUpdate(RunData data, Context context)
    {
        String number = data.getParameters().getString(„number“);
        String result = null;

        if (!number.equals(„“))
        {

            VelocityPortlet portlet = (VelocityPortlet)context.get(„portlet“);
            portlet.setAttribute(„number“,number,data);
           result = SOAPDemoPortlet.getCardStatus(number);
            context.put(„text“,result);

    if (result.indexOf(„Valid“) >= 0)
    {
          context.put(„status“, „OK“);
          context.put(„balance“, „0,00“);
    }
    else
    {
          context.put(„status“, „NOK“);
    }


            context.put(„message“, „Anfrage erfolgreich“);

        }
        else
        {
            context.put(„text“, „Keine Kartennummer angegeben !“);
           context.put(„status“, „NOK“);
        }
    }
}
```

src/java/org/apache/jetspeed/portal/portlets/DAWelcomePortlet.java

```
package org.apache.jetspeed.portal.portlets;

import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.turbine.util.RunData;
import org.apache.ecs.*;
import org.apache.ecs.html.*;

public class DAWelcomePortlet extends AbstractPortlet
{
    public ConcreteElement getContent (RunData aRunData)
    {
       Table table = new Table()
      .setBorder(0)
      .addElement(new TR()
        .addElement(new TD()
        .setHeight(150)
        .setVAlign("bottom")
        .setAlign("center")
        .addElement(new H1()
        .addElement("Diplomarbeit Meik Arends"))))
```

```
.addElement(new TR()
   .addElement(new TD()
   .setAlign("center")
   .addElement(new H2()
  .addElement("M&ouml;glichkeiten der Anwendungsintegration und Personalisierung Web-basierter
              Informationssysteme durch Portale am Beispiel eines Handelssystems f&uuml;r
              Warentermin gesch&auml;fte"))
    .addElement(new H3()
   .addElement("-Konzepte und Realisierung-"))))
 .addElement(new TR()
    .addElement(new TD()
    .setHeight(150)
   .addElement(" ")));

  return table;
  }
}
```

src/java/org/apache/jetspeed/portal/portlets/MarketInfoPortlet.java

```
package org.apache.jetspeed.portal.portlets;

import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.turbine.util.RunData;
import org.apache.ecs.*;
import org.apache.ecs.html.*;

public class MarketInfoPortlet extends AbstractPortlet
{
    public ConcreteElement getContent (RunData aRunData)
    {
Center table = new Center()
   .addElement(new Table()
   .setBorder(0)
   .addElement(new TR()
       .addElement(new TD()
      .setAlign(„center“)
      .addElement(new IMG()
      .setSrc(„http://gfx.finanztreff.de/charts/cc_gatrixx.gfx?typ=1&
             string=DAX&boerse=9&uvon=0900&ubis=2015&land=276&h=100&b=130&realtime=1“)
      .setAlign(„center“))))
   .addElement(new TR()
       .addElement(new TD()
      .setAlign(„center“)
      .addElement(new IMG()
      .setSrc(„http://gfx.finanztreff.de/charts/cc_gatrixx.gfx?typ=1&
             string=NMKX&boerse=9&uvon=0900&ubis=2015&land=276&h=100&b=130&realtime=1“)
      .setAlign(„center“))))
       .addElement(new TR()
       .addElement(new TD()
   .setHeight(20)
   .addElement(„&copy; Gatrixx (www.gatrixx-finanztreff.de) “))));
    return table;
   }
}
```

src/java/org/apache/jetspeed/portal/portlets/SOAPDemoPortlet.java

package org.apache.jetspeed.portal.portlets;

```
import org.apache.ecs.*;
import org.apache.ecs.html.*;

import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.turbine.util.RunData;

import org.w3c.dom.*;
import org.xml.sax.*;


import org.apache.soap.util.xml.*;
import org.apache.soap.*;
import org.apache.soap.Body.*;
import org.apache.soap.encoding.*;
import org.apache.soap.encoding.soapenc.*;
import org.apache.soap.rpc.*;
import org.apache.soap.transport.*;
import org.apache.soap.transport.http.SOAPHTTPConnection;

import java.io.*;
import java.io.IOException;
import java.net.*;
import java.util.Vector;
import javax.xml.parsers.*;



public class SOAPDemoPortlet extends AbstractPortlet {

   public static String getCardStatus(String CreditCardNumber) {

      URL url = null;

      // Definition der SOAP Endpoint URL. An diese wird die Nachricht letztendlich ¸bertragen
      // (Beispielsweise via HTTP POST)
      try {
             url=new URL("http://www.iemfamily.com/iemfamily/CreditCard.asmx");

      } catch (MalformedURLException mue) {

         return mue.getMessage();
                  }

      // Hier wird das Haupt-Objekt definiert, der SOAP-Call
      Call soapCall = new Call();

     // Festlegung des zu nutzenden Encoding-Schemas
      soapCall.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);

      // Hier findet der Aufruft der neuen Methode des von mir erweiterten SOAP-API statt und
      // verhindert die Generierung von Namespaces im <SOAP-ENV:-Body>. Dies reduziert die
      // Interoperatibilitätsprobleme bei Anfragen an Microsoft .NET SOAP-Servern.
     soapCall.setConnectToDotNet("true");

      // Aufgrund der obigen Interoperatibilit‰tsproblemen bei der
     // Komminikation mit .NET SOAP-Servern muss auch die
      // SOAP-Response einem entsprechenden Namespace zugeordnet werden.
      // Ansonsten kann die Response vom .NET Server durch den Client nicht ausgewertet werden.
      // Dies geschieht durch Erweiterung des genutzen Encodings Constants.NS_URI_SOAP_ENC
      // welches http://schemas.xmlsoap.org/soap/envelope/ entspricht.
     SOAPMappingRegistry smr = new SOAPMappingRegistry ();
     StringDeserializer sd = new StringDeserializer ();
     smr.mapTypes (Constants.NS_URI_SOAP_ENC, new QName ("http://tempuri.org/",
          "VerifyCreditCardResult"), String.class, null, sd);
```

```
        soapCall.setSOAPMappingRegistry (smr);

        // Wurde von aktuellen Entwicklungen ¸berholt, da der Server-Betreiber den Support
        // f¸r SOAPRPC eingerichtet hat
        soapCall.setConnectToDotNet("true");

        // Hier wird das aufzurufende Objekt angegebben
        soapCall.setTargetObjectURI("http://tempuri.org/");

        // Die aufzurufende Methode wird hier festgelegt
        soapCall.setMethodName("VerifyCreditCard");

        // F¸r die ‹bermittlung der Parameter an die Methode wird eine Vector als Datentyp benˆtigt
        Vector soapParams = new Vector();

        // Parameter mit (name, type, value, encoding style)
        Parameter CreditCardNumberParam = new Parameter("CardNumber", String.class,
                    CreditCardNumber, null );

        // Hinzuf¸gen des Parameters zum Vector
        soapParams.addElement(CreditCardNumberParam);

        // Setzen des Vectros zur Parameter¸bergabe an die Methode
        soapCall.setParams(soapParams);

        try {

           // Ausf¸hren des SOAP-calls mit der entsprechenden SOAPAction
           Response soapResponse = soapCall.invoke(url,"http://tempuri.org/VerifyCreditCard");

           // Tritt ein Fehler auf wird eine Fehlermeldung zur¸ckgeliefert
           if (soapResponse.generatedFault()) {
              Fault ft = soapResponse.getFault();
             return ft.toString();

           } else {

              // Auslesen des R¸ckgabewertes in der SOAP Response
             Parameter ret = soapResponse.getReturnValue();
             Object value = ret.getValue();

             return (value.toString());
          }

        } catch (SOAPException se) {

           return se.getMessage();
        }
     }

      public ConcreteElement getContent(RunData runData) {

     StringElement price = new StringElement();

     // Aufruf und ‹bergabe der Kreditkartennummer
     price.addElement("Java Servlet Programming: " + getCardStatus("4111111111111111"));

         return price;
     }
}
```

webapp/WEB-INF/templates/vm/portlets/html/CreditCardCheck.vm

```
#if (!$text)
#set ( $text = „STATUS: unbekannt“ )
#end
#if (!$number)
#set ( $number = „“ )
#end
#if (!$status)
#set ( $status = „“ )
#end
#if (!$balance)
#set ( $balance = „-354,20“ )
#end
<form action=“$jlink“ method=“post“>
<font color=“$!{skin.Color}“>
<br />
#if ($status != „OK“)
#if ($status == „NOK“)
<h2>Karte ung&uuml;ltig ($text)</h2>
#else
<h2>Zahlen mit einer Kreditkarte</h2>
#end
<br />
<h3>Kontostand ($data.User.UserName): $balance EUR</h3>
<p>Bitte geben Sie Ihre Kreditkartennummer ein:
<br /><input name=“number“ value=“„ size=“30“ />
</p>
<input type=“submit“ name=“eventSubmit_doUpdate“ value=“Absenden“ />
#else
<h2>Karte g&uuml;ltig ($text)</h2>
<h3>Zahlung erfolgt !</h3>
<h3>Kontostand: $balance EUR</h3>
<p>Vielen Dank !
</p>
#end
</font>
</form>
```

webapp/WEB-INF/conf/local-portlets.xreg

```
<?xml version=“1.0“ encoding=“UTF-8“?>
<registry>
   <portlet-entry name=“DAWelcome“ hidden=“false“ type=“instance“ application=“false“>
        <meta-info>
            <title>WELCOME-Portlet</title>
            <description>Startbildschirm der DA</description>
        </meta-info>
        <classname>org.apache.jetspeed.portal.portlets.DAWelcomePortlet</classname>
       <media-type ref=“html“/>
    </portlet-entry>
   <portlet-entry name=“MarketInfo“ hidden=“false“ type=“instance“ application=“false“>
        <meta-info>
            <title>Marktionformationen</title>
            <description>DAX und NEMAX Verlauf</description>
        </meta-info>
        <classname>org.apache.jetspeed.portal.portlets.MarketInfoPortlet</classname>
       <media-type ref=“html“/>
    </portlet-entry>
   <portlet-entry name=“SOAPDemo“ hidden=“false“ type=“instance“ application=“false“>
        <meta-info>
            <title>SOAP-Demo</title>
            <description>Beispiel fuer Anwendungsintegration</description>
        </meta-info>
        <classname>org.apache.jetspeed.portal.portlets.SOAPDemoPortlet</classname>
       <media-type ref=“html“/>
    </portlet-entry>
</registry>
```

B.2 SOAP-Patches

RPCMessage.java

```
--- soap-2_2-orig/src/org/apache/soap/rpc/RPCMessage.javaTue May 29 21:58:26 2001
+++ soap-2_2/src/org/apache/soap/rpc/RPCMessage.javaWed Sep 12 17:39:00 2001
@@ -1,6 +1,12 @@
 /*
- * The Apache Software License, Version 1.1
+ * The Apache Software License, Version 1.1-MA
+ * ------------------------------------------------------------------------------
+ * Modified by Meik Arends 12.09.01 (meik@arends-online.de)
+ * Added setConnectToDotNet() method for disabling namespace and
+ * encoding stuff in <SOAP-ENV:Body> to connect to .NET services
   *
+ * Reference it from your Java client as <yourCallDescriptor>.setConnectToDotNet(„true“)
+ *------------------------------------------------------------------------------
   *
   * Copyright (c) 2000 The Apache Software Foundation.  All rights
   * reserved.
@@ -65,6 +71,7 @@
 import org.apache.soap.*;
 import org.apache.soap.encoding.*;
 import org.apache.soap.server.*;
+import org.apache.soap.encoding.soapenc.*;

 /**
  * An <code>RPCMessage</code> is the base class that <code>Call</code> and
@@ -77,6 +84,7 @@
 public class RPCMessage implements Serializer {
   protected String targetObjectURI;
   protected String fullTargetObjectURI;
+  protected String ConnectToDotNet;
   protected String methodName;
   protected Vector params;
   protected Header header;
@@ -104,6 +112,11 @@
                               targetObjectURI);
   }

+  public void setConnectToDotNet(String ConValue) {
+      this.ConnectToDotNet = ConValue;
+      SoapEncUtils.setConnectToDotNet(ConValue);
+    }
+
   public String getTargetObjectURI() {
     return targetObjectURI;
   }
@@ -232,12 +245,24 @@

       if (!resp.generatedFault()) {
         // Get the prefix for the targetObjectURI.
+  String targetObjectNSPrefix = null;
+
+  if (ConnectToDotNet != „true“ ) {
         StringWriter nsDeclSW = new StringWriter();
-        String targetObjectNSPrefix = nsStack.getPrefixFromURI(
+      targetObjectNSPrefix = nsStack.getPrefixFromURI(
           targetObjectURI, nsDeclSW);
-
         sink.write(,<, + targetObjectNSPrefix + ,:‘ +
                    methodName + suffix + nsDeclSW);
+  }
```

```
+  else
+      {
+  StringWriter nsTrash = new StringWriter();
+  targetObjectNSPrefix = nsStack.getPrefixFromURI(
+ targetObjectURI, nsTrash);
+  sink.write(,<, +
+     methodName + suffix + „ xmlns=\“„ + targetObjectURI + „\“„);
+
+      }

         // Determine the prefix associated with the NS_URI_SOAP_ENV
         // namespace URI.
@@ -246,10 +271,20 @@

         if (declMsgEncStyle != null
             && !declMsgEncStyle.equals(inScopeEncStyle)) {
+
+     if (ConnectToDotNet != „true“ ) {
           sink.write(, , + soapEnvNSPrefix + ,:‘ +
                      Constants.ATTR_ENCODING_STYLE + „=\“„ +
                      declMsgEncStyle + ,\“,);
         }
+     else
+ {
+     sink.write(, , +
+Constants.ATTR_ENCODING_STYLE + „=\“„ +
+declMsgEncStyle + ,\“,);
+ }
+
+        }

         sink.write(,>‘ + StringUtils.lineSeparator);

@@ -272,9 +307,19 @@

         serializeParams(params, actualMsgEncStyle, sink, nsStack, xjmr, ctx);

+ if (ConnectToDotNet != „true“ ) {
         sink.write(„</“ + targetObjectNSPrefix + ,:‘ +
                    methodName + suffix + ,>‘ +
                    StringUtils.lineSeparator);
+ }
+ else
+     {
+
+ sink.write(„</“ +
+    methodName + suffix + ,>‘ +
+    StringUtils.lineSeparator);
+     }
+
       } else {
         // Get the fault information.
         Fault fault = resp.getFault();
@@ -283,12 +328,25 @@
       }
     } else {
       // Get the prefix for the targetObjectURI.
+String targetObjectNSPrefix = null;
+if (ConnectToDotNet != „true“ ) {
       StringWriter nsDeclSW = new StringWriter();
-      String targetObjectNSPrefix = nsStack.getPrefixFromURI(targetObjectURI,
+
+    targetObjectNSPrefix = nsStack.getPrefixFromURI(targetObjectURI,
                                                              nsDeclSW);

       sink.write(,<, + targetObjectNSPrefix + ,:‘ +
                  methodName + suffix + nsDeclSW);
+}
```

```
+else
+    {
+
+StringWriter nsTrash = new StringWriter();
+targetObjectNSPrefix = nsStack.getPrefixFromURI(targetObjectURI,
+      nsTrash);
+sink.write(,<, +
+   methodName + suffix + „ xmlns=\“„ + targetObjectURI + „\“„);
+    }

       // Determine the prefix associated with the NS_URI_SOAP_ENV
       // namespace URI.
@@ -297,17 +355,34 @@

       if (declMsgEncStyle != null
           && !declMsgEncStyle.equals(inScopeEncStyle)) {
+
+  if (ConnectToDotNet != „true“ ) {
         sink.write(, , + soapEnvNSPrefix + ,:‘ +
                    Constants.ATTR_ENCODING_STYLE + „=\“„ +
                    declMsgEncStyle + ,\“,);
       }
+  else
+      {
+
+  sink.write(, , +
+     Constants.ATTR_ENCODING_STYLE + „=\“„ +
+     declMsgEncStyle + ,\“,);
+      }
+      }

       sink.write(,>‘ + StringUtils.lineSeparator);

       serializeParams(params, actualMsgEncStyle, sink, nsStack, xjmr, ctx);

+      if (ConnectToDotNet != „true“ ) {
       sink.write(„</“ + targetObjectNSPrefix + ,:‘ +
                  methodName + suffix + ,>‘);
+      }
+      else
+  {
+      sink.write(„</“ +
+ methodName + suffix + ,>‘);
+  }
     }

     nsStack.popScope();
```

SoapEncUtils.java

```
--- soap-2_2-orig/src/org/apache/soap/encoding/soapenc/SoapEncUtils.javaTue May 29 21:58:25 2001
+++ soap-2_2/src/org/apache/soap/encoding/soapenc/SoapEncUtils.javaWed Sep 12 17:39:01 2001
@@ -1,7 +1,13 @@
 /*
- * The Apache Software License, Version 1.1
- *
- *
+ * The Apache Software License, Version 1.1-MA
+ * ------------------------------------------------------------------
+ * Modified by Meik Arends 12.09.01 (meik@arends-online.de)
+ * Added setConnectToDotNet() method for disabling namespace and
+ * encoding stuff in <SOAP-ENV:Body> to connect to .NET services
+ *
+ * It will be referenced from RPCMessage.java if you set
+ * <yourCallDescriptor>.setConnectToDotNet(„true“) in your Java client
+ * ------------------------------------------------------------------
  * Copyright (c) 2000 The Apache Software Foundation.  All rights
  * reserved.
  *
@@ -71,6 +77,12 @@
  */
 public class SoapEncUtils
 {
+  protected static String ConnectToDotNet;
+
+  public static void setConnectToDotNet(String ConValue) {
+      ConnectToDotNet = ConValue;
+  }
+
   public static void generateNullStructure(String inScopeEncStyle,
                                            Class javaType, Object context,
                                            Writer sink, NSStack nsStack,
@@ -137,9 +149,11 @@
     String elementTypeNSPrefix = nsStack.getPrefixFromURI(
       elementType.getNamespaceURI(), sink);

+     if (ConnectToDotNet != „true“ ) {
     sink.write(, , + xsiNSPrefix + ,:‘ + Constants.ATTR_TYPE + „=\“„ +
               elementTypeNSPrefix + ,:‘ +
               elementType.getLocalPart() + ,\“,);
+     }

     if (inScopeEncStyle == null
         || !inScopeEncStyle.equals(Constants.NS_URI_SOAP_ENC))
@@ -149,10 +163,17 @@
       String soapEnvNSPrefix = nsStack.getPrefixFromURI(
         Constants.NS_URI_SOAP_ENV, sink);

+      if (ConnectToDotNet != „true“ ) {
       sink.write(, , + soapEnvNSPrefix + ,:‘ +
                  Constants.ATTR_ENCODING_STYLE + „=\“„ +
                  Constants.NS_URI_SOAP_ENC + ,\“,);
     }
+      else {
+  sink.write(, , +
+     Constants.ATTR_ENCODING_STYLE + „=\“„ +
+     Constants.NS_URI_SOAP_ENC + ,\“,);
+      }
+    }

     if (arrayElementType != null)
     {
@@ -164,15 +185,27 @@
       String soapEncNSPrefix = nsStack.getPrefixFromURI(
         Constants.NS_URI_SOAP_ENC, sink);
```

```
+      if (ConnectToDotNet != „true“ ) {
       sink.write(, , + soapEncNSPrefix + ,:‘ +
                  Constants.ATTR_ARRAY_TYPE + „=\“„ + arrayTypeValue + ,\“,);
     }
+      else {
+  sink.write(, , +
+             Constants.ATTR_ARRAY_TYPE + „=\“„ + arrayTypeValue + ,\“,);
+      }
+    }

     if (isNull)
     {
+if (ConnectToDotNet != „true“ ) {
       sink.write(, , + xsiNSPrefix + ,:‘ + Constants.ATTR_NULL + „=\“„ +
                  Constants.ATTRVAL_TRUE + „\“/“);
     }
+else {
+    sink.write(, , + Constants.ATTR_NULL + „=\“„ +
+       Constants.ATTRVAL_TRUE + „\“/“);
+}
+    }

     sink.write(,>‘);
   }
```

Literaturverzeichnis

[Abo01] About.com, Inc., *Computer Networking*, http://compnetworking.about.com/compute/compnetworking/library/glossary/bldef-portal.htm?IAM=mcrawl-de&terms=+portal++definition, 2001

[BEA01] BEA, *Portlet Developers´s Guide*, http://edocs.bea.com/wlac/portals/docs/portlet.html#what, 2001

[BuGo99] Burris P., Gotta M., *Enterprise Portals: The Killer Application for the Web User Interface*, http://www.metagroup.com/metaview/mv0122/mv0122.html, 1999

[Brio01] Brio Technology, *Brio.Portal 7.0*, http://www.brio.com/products_solutions/brio_portal/index.html, 2001

[Brio01b] Brio Technology, *Brio.Report (Industrial-Strength Enterprise Reporting)*, http://www.brio.com/pdfs/report.pdf, 2001

[Cis01] Associate Professor, Computer and Information Science Department, *CIS307 : Introduction to Distributed Systems and Networks*, *http*://www.cis.temple.edu/~ingargio/cis307/readings/corba.html, Temple University, Philadelphia, Spring 2001

[DOM01] W3C, *Document Object Model (DOM)*, http://www.w3c.org/DOM, August 2001

[ECS01] The Jakarta Project, *Element Construction Set (ECS)*, http://jakarta.apache.org/ecs, 2001

[Epi01] Epicentric, *Epicentric Foundation Server 3.5*, http://www.epicentric.com/solutions/efs.jsp, 2001

[Exo01] Exolab, *Castor, an open source data binding framework for Java*, http://castor.exolab.org, 2001

[Foch97] K. Fochler, P. Perc und Jörg Ungermann, *Electronic Commerce mit Lotus Domino*, Kapitel 1, 1. Auflage 1997

[Gam96] Erich Gamma et al., *Entwurfsmuster*, Addison Wesley, 1. Auflage 1996

[Geb01] Christian Gebauer, *SOAP*, Seminar: DB-Aspekte des E-Commerce; Schwerpunkt: Techniken, Universität Kaiserslautern, SS 2001

[GrTh00] V. Gruhn, A. Thiel, *Komponentenmodelle, DCOM, JavaBeans, Enterprise JavaBeans, CORBA*, Addison-Wesley, 2000

[Han98] H.R. Hansen. *Internet-Glossar zum Buch "Wirtschaftsinformatik I"*, http://wicky.wu-wien.ac.at/glossar/1-2_17.htm, Stuttgart, 1998

[Has01] W. Hasselbring, A. Koschel, A. Mester, *Basistechnologien für die Entwicklung von Internet-Portalen*, 2001

[HeRo01] U. Heck, A. Rogger, P. Eichenberger, *XML und Java als komplementärer Integrationsansatz*, Java Spektrum, Ausgabe 4/2001

[Hr83] T. Härder. A. Reuter, *Principles of Transaction Oriented Database Recovery*, In: ACM Computing Surveys, Vol. 15, No. 4, 1983

[HSQL01] Source Forge, *HSQL Database Engine*, http://sourceforge.net/projects/hsqldb, 2001

[HST97] T. Härder, G. Sauter, J. Thomas, *The Intrinsic Problems of Heterogenity and an Approach to their Solution*, 1997

[HTML99] W3C, *HTML 4.01 Specification*, http://www.w3.org/TR/html4, 1999

[HTTP99] Fielding, et al, *RFC2616 - Hypertext Transfer Protocol -- HTTP/1.1*, The Internet Society, 1999

[Hyp01] Hyperwave AG, *Hyperwave Information Server 5.5*, http://www.hyperwave.de/d/products/his.html, 2001

[IBM01] IBM, *WebSphere Portal Server*, http://www-4.ibm.com/software/webservers/portal, 2001

[IBM01b] IBM, *WebSphere Transcoding Server*, http://www-4.ibm.com/software/webservers/transcoding, 2001

[Ipl01] iPlanet e-commerce solutions, *iPlanet Portal Server*, http://www.iplanet.com/products/iplanet_portal/ds_portalsrv.pdf, 2001

[Ipl01b] iPlanet e-commerce solutions, *Four Steps to Web Content Management Success*, http://www.iplanet.com/about_us/features/interwoven_iplanet_5_2_1aa.html, 2001

[Jet00] XML.com, *XML at Jetspeed*, http://www.xml.com/pub/a/2000/05/15/jetspeed/index.html, 2000

[Jet01] The Jakarta Project, *Jetspeed*, http://jakarta.apache.org/jetspeed/site/index.html, 2001

[Jet01b] The Jakarta Project, *Jetspeed - Standard Portlets*, http://jakarta.apache.org/jetspeed/site/catalog.html, 2001

[JS01] Netscape, *JavaScript Resources*, http://developer.netscape.com/tech/javascript/resources.html, 1999

[Kno01] knowinc.com, *Know Inc. Portal Model*, http://knowinc.com/partnership/portal.htm, 2001

[Loe98] Henrik Loeser, *Techniken für Web-basierte Datenbankanwendungen: Anforderungen, Ansätze, Architekturen*, In: Informatik - Forschung und Entwicklung 13(4), Springer, 1998

[MVC01] Sun, *Model-View-Controller Architecture*, http://java.sun.com/j2ee/blueprints/design_patterns/model_view_controller/index.html, 2001

[NLM00] Nielsen H., Leach P., Microsoft et al, *RFC2774 - An HTTP Extension Framework*, Network Working Group, http://www.faqs.org/rfcs/rfc2774.html, Februar 2000

[ODBC01] Microsoft, *ODBC Technical Materials*, http://www.microsoft.com/data/techmat.htm#odbc, 2001

[Omg01] Object Management Group, *CORBA 2.4.2 Spezification*, http://www.omg.org/technology/documents/formal/corba_2.htm, 2001

[Ora00] Oracle, *Oracle Portal 3.0 - Production Release, Product Overview, Product Features*, August 2000

[Ora00b] Todd D. Vender Orcale Corporation, *Enhancing Oracle Portal with Data-Driven Portlets*, http://otn.oracle.com/products/iportal/pdf/enhace_oportal_data_portlets.pdf, Oktober 2000

[Ora01] Oracle, *Heterogeneous Data Access*, http://www.oracle.com/gateways/, 2001

[Ora01b] Oracle, *Oracle9iAS InterConnect*, http://www.oracle.com/ip/deploy/ias/docs/oai_fov.html, 2001

[PWC00] PricewaterhouseCoopers, *Leitfaden E-Business: EAI - Enterprise Application Integration*, PwC Deutsche Revision, http://www.pwcglobal.com/de/ger/ins-sol/publ/20_pwc_e-bus.pdf, Frankfurt Juni 2000

[PDF01] Adobe, *Adobe PDF*, http://www.adobe.com/products/acrobat/adobepdf.html, 2001

[PS01] Adobe, *Adobe Postscript 3*, http://www.adobe.com/products/postscript/resources.html#white, 2001

[RPC92] W3C, *Introduction to RPC*, http://www.w3.org/History/1992/nfs_dxcern_mirror/rpc/doc/Introduction/Abstract.html, 1992

[RSS01] RSS-DEV Working Group, *RDF Site Summary (RSS) 1.0*, http://groups.yahoo.com/group/rss-dev/files/specification.html, 2001

[Sah00] Ü. Sahin, Salih Korkmaz, Baris Kildi, Miriman Saiti, *Wissensmanagement und Wettbewerb*, http://user.cs.tu-berlin.de/~tron/oss/seminar/knowledge/img18.htm, TU Berlin 2000

[SAX00] Megginson Technologies, *SAX 2.0: The Simple API for XML*, http://www.megginson.com/SAX, 2000

[SLH01] Stetson Library Home Page, *Internet Portals*, http://www.sldirectory.com/searchf/portals.html, 2001

[SSL96] Transport Layer Security Working Group, *The SSL Protocol Version 3.0*, http://home.netscape.com/eng/ssl3/draft302.txt, 1996

[Stan98] Katarina Stanoevska-Slabeva et al., *Ein Glossar für die NetAcademy on Business Media*, mcminstitute- Working Report- 1998-13, University of St. Gallen 03/1998

[Stei98] Hans-Peter Steiert, *Towards a Component-based n-Tier C/S-Architecture*, In: 3rd Int. Software Architecture Workshop (ISA W3), Orlando, ACM Press, 1998

[SOAP01] W3C, *Simple Object Access Protocol (SOAP) 1.1*, http://www.w3c.org/TR/SOAP, Mai 2000

[SOAP01b] XML Apache Project, *SOAP Documentation: Interoperability with Other SOAP Implementations*, http://xml.apache.org/soap/docs/guide/interop.html, 2001

[SZ98] Steiert, H.-P., Zimmermann, J.: *JPMQ - An Advanced Persistent Message Queuing Service*, In: Advances in Databases, Proc. 16th British Nat. Conf. on Data Management (BNCOD16), LNCS 1405, Springer, 1998

[Tar01] Tarak Modi, *Clean up your wire protocol with SOAP (Part 1-4)*, http://www.javaworld.com/javaworld/jw-03-2001/jw-0330-soap_p.html , JavaWorld, 2001

[TMA01] The Mail Archive, *An easy-to-use archiving service for electronic mailing lists (hier Jetspeed)*, http://www.mail-archive.com/index.php3?hunt=jetspeed

[Tom01] The Jakarta Project, *Jakarta Tomcat*, http://jakarta.apache.org/tomcat/index.html, 2001

[Turb01] The Jakarta Project, *What is Turbine*, http://jakarta.apache.org/turbine/, 2001

[Vel01] The Jakarta Project, *Velocity*, http://jakarta.apache.org/velocity/index.html, 2001

[Vil01] Whichever Group, *Village*, http://share.whichever.com/index.php?SCREEN=village, 2001

[W3C01] W3C, *The World Wide Web Consortium*, http://www.w3.org, 2001

[WAP00] WAP Forum, *The June 2000 (WAP 1.2.1) conformance release*, http://www.wapforum.org/what/technical.htm, 2000

[Weba01a] Webagency, *Electronic Commerce Informationspool: E-Commerce-Know-how*, http://www.webagency.de/infopool/e-commerce-knowhow/ak981021.htm, 2001

[Weba01b] Webagency, *Electronic Commerce Informationspool: E-Commerce-Know-how*, http://www.webagency.de/infopool/e-commerce-knowhow/ak981030.htm, 2001

[Webo01] Webopedia, *Online Computer Dictionary for Internet Terms and Technical Support*, http://webopedia.internet.com/TERM/W/Web_portal.html, 2001

[Webo01b] Webopedia, *Online Computer Dictionary for Internet Terms and Technical Support*, http://webopedia.internet.com/TERM/L/LDAP.html, 2001

[Weid01] M.Weidner, *Diplomarbeit: Kapazitätsorientierte Produktionsplanung und -steuerung auf der Basis terminmarktlichen Product-Pricings*, Lehrstuhl für Fertigungstechnik und Betriebsorganisation, Universität Kaiserslautern, 2001

[Wiel01] T. Wieland, *Beschreibung von WebServices*, Java Spektrum, Ausgabe 4/2001

[Wol01] Folkhard Wolf, *Portal total*, http://www.software.de.ibm.com/infocenter/IC_ENEWS.NSF/eNewsPublic/2001-02-02-ess-1, Februar 2001

[WSDL01] W3C, *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/wsdl, 2001

[XHTML01] W3C, *XHTML 1.0: Extensible HyperText Markup Language*, http://www.w3.org/TR/xhtml1, 2000

[XML01] W3C, *Extensible Markup Language (XML)*, http://www.w3.org/XML, 2001

[XML01b] The Apache XML Project, *Welcome to the Apache XML Project*, http://xml.apache.org, 2001

[XPATH99] W3C, *XML Path Language (XPath)*, http://www.w3.org/TR/xpath.html, 1999

[XSL01] W3C, *Extensible Stylesheet Language (XSL)*, http://www.w3.org/Style/XSL, 2001

[XSLT01] W3C, *XSL Transformations (XSLT)*, http://www.w3.org/TR/xslt.html, 1999