Fachhochschule Köln

University of Applied Sciences Cologne

Fakultät Informatik und Ingenieurwissenschaften

Diplomarbeit

zur Erlangung des Diplomgrades

Diplom-Informatiker (FH)

in der Fachrichtung Wirtschaftsinformatik

Enterprise Application Integration (EAI) als Basis für Portale

Erstprüfer: Prof. Dr. Heide Faeskorn - Woyke

Zweitprüfer: Prof. Dr. Frank Victor

vorgelegt am: 14. November 2002

von: Thomas Fischer

Georgstr. 5a

50676 Köln

Matrikelnummer: 1102222916

Inhaltsverzeichnis

Abkürungsverzeichnis

API	Application Programming Interface
DB2	IBM Database 2
DOM	Document Object Model
EPK	Ereignisgesteuerte Prozesskette
EAI	Enterprise Applikation Integration
HTML	Hypertext Markup Language
IBM	International Business Machine
IMS	Information Management System
JRE	Java Runtime Environment
LDAP	Lightweight Directory Access Protocol
MQ	Message Qeuing
OLAP	Online Analytical Processing
PDF	Printable Document Format
SAX	Simple API for XML Processing
SCM	Supply Chain Management
UC	Use Case
UML	Unified Modeling Language
WAS	WebSphere Application Server
XML	Extensable Markup Language
WWW	World Wide Web

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Portale erfreuen sich in der Internet-Welt des World Wide Web einer großen Beliebtheit. Hervorgegangen aus Suchmaschinen werden Web-Portale von einer Vielzahl von Anbietern, wie etwa Yahoo, AOL, Altavista und Netscape, angeboten. Diese Web-Portale bieten neben Suchfunktionen und einem Angebot populärer Web-Seiten eine Vielzahl von weiteren Diensten, wie z.B. E-Mail, Shopping, Diskussionen und Chat.
Web-Portale, wie wir sie heute kennen, sind das Ergebnis einer Evolution im World Wide Web. Um sich in der Informationsvielfalt des WWW zu Recht zu finden, ist für die Benutzer eine Suchmaschine allein nicht ausreichend. Eine Vielzahl der Benutzer im Internet wünscht sich einen zentralen und einfachen Zugangspunkt, einen “Single Point of Access“, von dem aus Verbindungen zu den relevanten Informationen und Diensten hergestellt werden können.
Um diesen „Single Point of Access“ auch für Unternehmen ermöglichen, stehen die verschiedenen Softwarehersteller bereit, um den Inter- und Intranetauftritt mit ihrer Portalsoftware neu zu organisieren und strukturieren.
Somit können sich die Unternehmen wieder auf die Optimierung ihrer Prozesse und der korrespondierenden Steuerungssysteme - unter Beibehaltung ihrer heterogenen Systemlandschaften - konzentrieren.

Themenbereiche wie Supply Chain Management, Customer Relationship Management oder Business Intelligence treten wieder in den Vordergrund Ihres Interesses. Für solche Anwendungen werden „Enterprise Portals“ und „Enterprise Application Integration“ zunehmend zu zentralen Gestaltungselementen für die Prozesse und Systemintegration.

Das Augenmerk bei Unternehmensportalen liegt – neben personalisierten Informationen für die Anwender – auf unternehmensübergreifender Zusammenarbeit und Systemintegration. In folge dessen entwickeln sich Unternehmensportale als zentrale Schnittstelle für die Anwender immer stärker auch zu einem Werkzeug der Integrationsaufgaben. Einerseits ist es dem Anwender möglich, das vorhandene Informationsangebot nach seinen Wünschen zu personalisieren – und damit auf seine

Interessen zu fokussieren – andererseits lassen sich dem Anwender alle relevanten Informationen, Applikationen und Services als Arbeitsumgebung bereitstellen.

Das reine Informationsportal, das dem Management vorbehalten war, gehört der Vergangenheit an. Gefragt sind Portale mit integrierten betriebswirtschaftlichen Anwendungen. Aus Sicht der IT müssen Portale als offene, virtuelle Integrationsplattformen für Daten und Funktionen heterogener Backend-Systeme und deren Integration durch übergreifende Workflow-Szenarien (Integration) fungieren.

1.1 Problemstellung

Diesen Wünschen stehen die heutigen IT-Systeme entgegen. In einem Zeitraum von über 50 Jahre haben sich in Unternehmen verschiedenste Hard- und Softwaresysteme angesammelt, die nicht für eine Zusammenarbeit entworfen wurden und auch nur Teilaufgaben von Geschäftsprozessen abdecken. Beim Thema Enterprise Applikation Integration (EAI) geht es darum, heterogene Anwendungen eines Unternehmens so zu integrieren, dass sie sich möglichst so verhalten, als wären sie von Anfang an dazu entworfen worden, die aktuellen Geschäftsprozesse eines Unternehmens zu unterstützen.

Der Zugriff auf Standardsoftware (SAP, Groupware) kann bei den Portallösungen bereits heute erfolgen. Mit den so genannten Portlets lassen sich Portale um verschiedene Applikationen erweitern (frei verfügbar oder unter Lizenz). Sie bieten der bekannten Clientanwendung entsprechenden Funktionalität, ersparen jedoch den hohen Installationsaufwand der verteilten Softwarebestandteile.
Die fertigen Portalanwendungen bieten i.d.R. jedoch keine Geschäftsprozessorientierung über die verschiedenen IT-Systeme. Was Unternehmen jedoch besonders fehlt, ist die Integration der Eigenentwicklungen in den Unternehmen. Gerade die – in tausenden Zeilen Code versteckt und oft in Hostanwendungen implementierten – Eigenentwicklungen enthalten das betriebwirtschaftliche Wissen.

Zur Unterstützung der Geschäftsprozesse und zur Sicherung des langfristigen Portalerfolges gilt es, diese Eigenentwicklungen mit dem Internet zu verbinden. Dabei zählt für die Unternehmen nicht nur dass, sondern vor allem, wie sie ihre Anwendungen und Geschäftsprozesse „web-tauglich“ machen. Schließlich müssen Sie

die Anwendung über viele Jahre warten. Dementsprechend müssen sie bereits bei der Entwicklung eine entsprechende Strategie verfolgen.

Dem Ziel Web-Applikationen leichter zu entwickeln und wartbar zu halten widmen sich verschiedene Frameworks (vgl. [Java02]). Evolution ist ein Framework, das Geschäftsprozesse ins Web bringt, indem es - unter Verwendung der entsprechenden Tools - ermöglicht, den Workflow grafisch zu erzeugen und auf Knopfdruck Programmcode entwickelt. Dieser Programmcode wird dann um die Teile erweitert, die auf die verschiedenen Applikationen im Unternehmen zugreifen können. Dabei enthält Evolution eine Reihe vorgefertigter Klassen, die eine einfache Einbindung der heterogenen Systeme ermöglichen.

1.2 Aufgabenstellung und Vorgehensweise

Die Aufgabe dieser Diplomarbeit ist die Entwicklung einer generischen Lösung für die Integration des zuvor genannten Frameworks Evolution und dem IBM Portalserver. Zur Durchführung der Realisation ist ein Prototyp zu entwickeln. Dieser Prototyp soll zunächst als Webanwendung laufen, bevor diese Anwendung mit dem Portalserver integriert wird.
Im Nachfolgenden sei das Vorgehen wie folgt beschrieben:

Grundlagen

Im Kapitel Grundlagen werden die nötigen technischen und fachlichen Grundlagen die zum Verständnis dieser Arbeit notwendig sind, erläutert. Als Bestandteil der technischen Grundlagen wird im Besonderen auf die aktuellen Technologien von XML und XSL eingegangen. Sie sind ein wichtiger Bestandteil der zu implementierenden Prototypen.

Portale

Das Kapitel Portale gibt einen Überblick über die verschiedenen Portalbegriffe und -arten und konzentriert sich dabei auf die Einsatzgebiete in Unternehmen. Nach der Erläuterung des Konzeptes hinter einem Portal und einem kurzen Portalmarktüberblick wird an Hand des IBM WebSphere Portal Server wird eine Portalserverlösung vorgestellt. Als Bestandteil dieses Portal Server wird ein Einblick in dessen Portlet-API – derzeit existiert noch keine einheitliche Programmierschnittstelle – gegeben.

Enterprise Applikation Integration

Portalserver sind durch installierbare Anwendungen erweiterbar. Für Standardsoftware sind Portlets[1] bereits verfügbar - Analysten der Delphi Group bezeichnen diese Portlets als „Infrastructure Software“. Die Begründung, warum diese jedoch nicht ausreichen, erfolgt in diesem Kapitel. Dabei werden Technologien aufgezeigt, wie Altsysteme an einen „webtaugliche“ angebunden werden können.

Softwareentwicklung mit „Evolution“

Der Enterprise Application Integration Lösungsweg[2], der als Basis für die Anwendungsentwicklung des Prototypen dienen soll, wird in diesem Kapitel vorgestellt. Hierbei handelt es sich um „Evolution“ einem Softwareentwicklungsmodell, das den gesamten Softwareentwicklungsprozess begleitet. Evolution ist u.a. mit dem Ziel entwickelt worden ist, Webanwendungen geschäftsprozessorientiert erstellen zu können. Unter Anwendung von aktuellen Entwicklungsmethoden können dabei verschiedene Leagacy-Systeme integriert werden.

Prototyp

Anhand eines ausführlichen Prototyps - einer Webanwendung im Applicationserver - wird der geschäfsprozessorientierte Lösungsansatzes von Evolution dargestellt werden. Dazu werden Anwendungsfälle identifiziert, beschrieben und modelliert, Datenmodelle sowie Klassendiagramme entwickelt, bevor die fachliche Logik implementiert wird.

Im Folgenden wird der erstellte Prototyp auf eine Portallösung erweitert. Dazu werden die Anforderungen der zu integrierenden Lösung ermittelt, verschiedene Lösungskonzepte aufgezeigt und bewertet. Die Machbarkeit der Integration von Evolution und Portalserver wird anhand der am besten bewerteten Lösung vorgestellt.

[1] beispielsweise ist der Zugriff auf SAP die „iViews“ möglich.
[2] es handelt sich hierbei sich nicht um ein fertiges Softwareprodukt!

1.3 Rahmenbedingungen

Im Rahmen dieser Arbeit werden Grundkenntnisse der Objektorientierung, der Programmiersprache Java (besonders Java-Servlet, JDBC) sowie Datenbankkenntnisse vorausgesetzt.

Der Prototyp wurde unter Verwendung folgender Software erstellt:

- IBM WebSphere Portal Server in der Version 2.1.
- IBM DB2 in der V. 7.1, Fixpack 2a, Datenbank
- IBM WebSphere Application Server V. 3.5.4
- IBM SecureWay LDAP Server in der Version 3.2.1, eingesetzt.

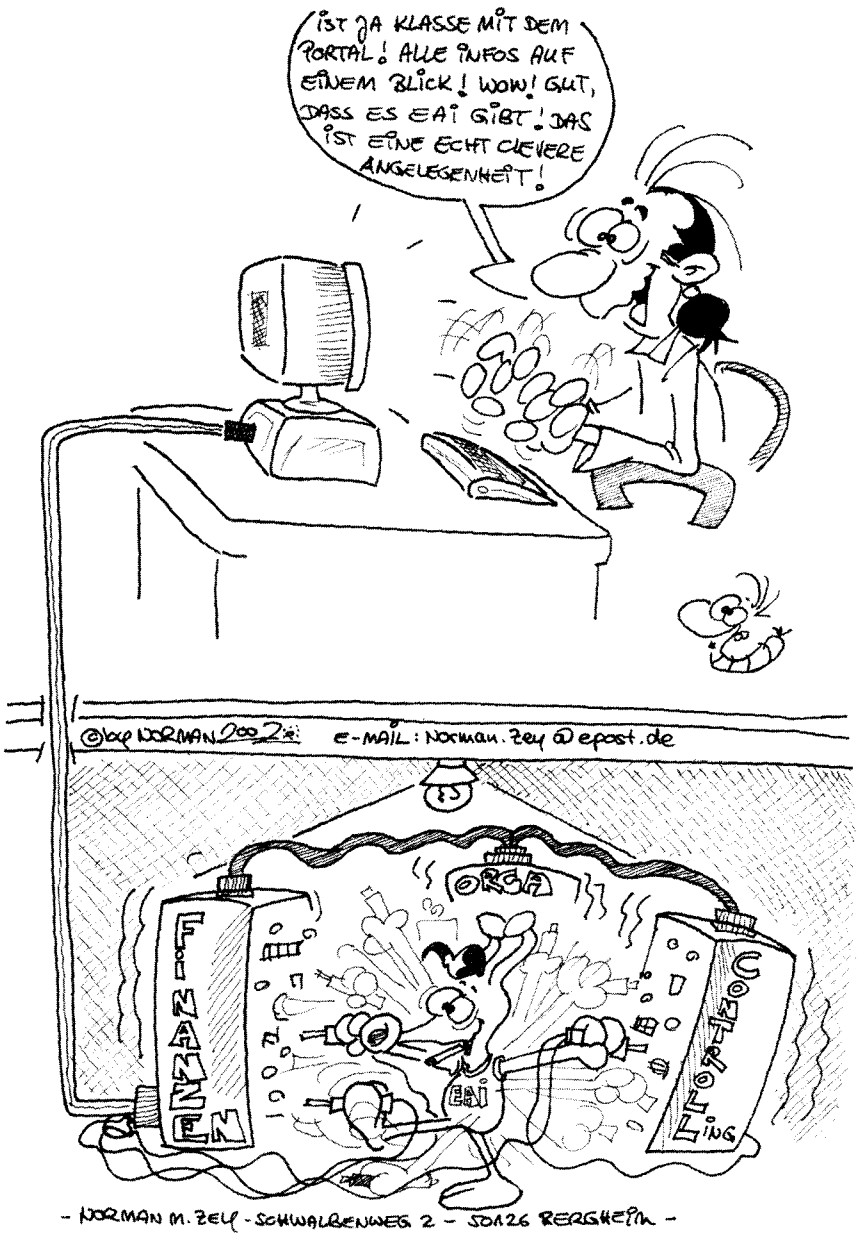
Als Entwicklungsumgebung werden folgende Produkte eingesetzt:

- Rational Rose Professional J edition 2000
  Rational Software Corporation, UML-Entwicklungsumgebung
- IBM Visual Age for Java, V 4.0, Java Entwicklungsumgebung
- XML Spy, V.3.5, Altova, XML, XSLT Entwicklungsumgebung
- ERwin, V. 3.5.2, Computer Associates International, Inc., Datenbankdesign

Tools

- XSplit, Percussion Software, [I.39]
  Überführt HTML nach XML, XSL
- Tablegen von Joe Carter, [I.40]
  generiert Java Klassen, die Datenbanktabellen repräsentieren

Mit der Bezeichnung der Portlet-API wird – sofern nicht anders darauf hingewiesen – die Portlet-API des IBM WebSphere Portal Servers in der Version 2.1 gemeint.


IST JA KLASSE MIT DEM PORTAL! ALLE INFOS AUF EINEM BLICK! WOW! GUT, DASS ES EAI GIBT! DAS IST EINE ECHT CLEVERE ANGELEGENHEIT!
©by NORMAN 2002 E-MAIL: Norman.Zey@epost.de
FINANZEN
ORGA
EAI
CONTROLLING
- NORMAN M. ZEY - SCHWALBENWEG 2 - 55126 BERGHEIM -

2 Grundlagen

2.1 Technische Grundlagen

2.1.1 XML

EXtensible Markup Language (XML) ist ein einfaches und sehr flexibles für Menschen lesbares Textformat.

Damit ist es im Gegensatz zu Formaten wie dem Word Dokumentformat (DOC) oder dem Rich Text Format (RTF) möglich den Inhalt einer Datei zu lesen, ohne ein spezielles Programm wie z.B. Microsoft Word zu verwenden. XML ist aber damit nicht nur für den Menschen lesbar. Dateien sind nicht nur davon abhängig, auf welchem Betriebssystem sie erzeugt wurden, auch durch Unterschiede zwischen den einzelnen Pogrammversionen (beispielsweise Microsoft Word) können Komplikationen auftreten. XML bietet durch Trennung der Darstellung vom Inhalt die Möglichkeit den Inhalt der Daten über die verschiedenen Systeme auszutauschen.

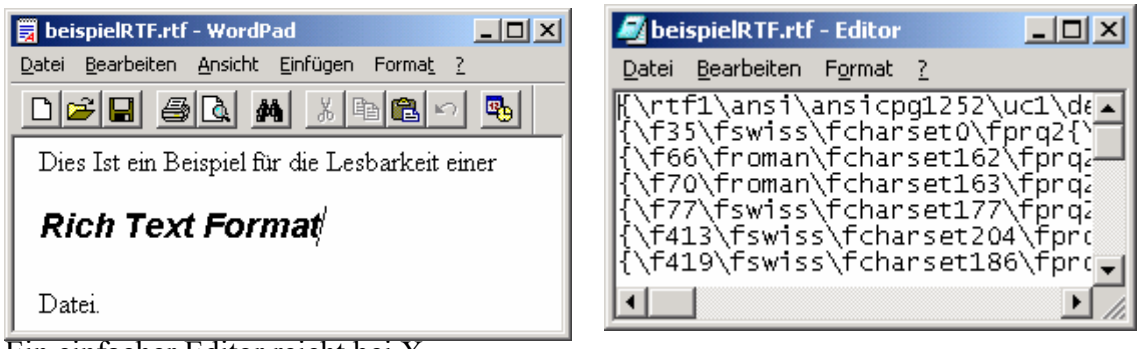


Abbildung 2-1 Beispiel Textformate

Die obige Abbildung zeigt das Dokument „beispielRTF.rtf“ einmal in dem Programm „Wordpad“ (links) und einmal in dem Programm „notepad“ (rechts). Während in der linken Darstellung alles normal erscheint, zeigt die rechte einen Ausschnitt daraus, wie das Beispiel-Dokument tatsächlich gespeichert wird. Hier ist die zu erkennen, wie die eigentlichen Daten (Text + Kennzeichnung der Überschrift) mit der grafischen Aufbereitung vermischt werden.

2.1.1.1 Definition

Das W3C definiert in ([I.1]) XML wie folgt:

„Die Extensible Markup Language, abgekürzt XML, beschreibt eine Klasse von Datenobjekten, genannt XML-Dokumente, und beschreibt teilweise das Verhalten von Computer-Programmen, die solche Dokumente verarbeiten. XML ist ein Anwendungsprofil (application profile) oder eine eingeschränkte Form von SGML, der Standard Generalized Markup Language [ISO 8879]. Durch ihre Konstruktion sind XML-Dokumente konforme SGML-Dokumente."

Grundsätzlich besteht ein XML-Dokument aus einem optionalen Prolog – der die XML-Deklaration und die Dokumenttypdefinition enthält - und einem Wurzelelement, das alle anderen Elemente oder Zeichendaten umfasst.

Das Activity Statement (vgl. [I.2]) liefert sechs Punkte, die Aufschluss darüber geben, was aus Sicht des W3C[3] durch XML ermöglicht werden soll:

1. Medienunabhängiges elektronisches Publizieren von Informationen in mehreren Sprachen
2. Definition von plattformunabhängigen Protokollen zum Datenaustausch (speziell für den Elektronischen Handel)
3. Automatische Verarbeitung übertragener Daten durch Software
4. Datenverarbeitung mit preisgünstiger Software
5. Benutzerdefinierte Präsentation von Informationen
6. Auffinden von Informationen durch Metadaten[4]

2.1.1.2 Elemente

Elemente sind das Markup des XML Dokumentes (d.h. sie beschreiben den Inhalt ihrer Daten). Sie werden in spitzen Klammern geschrieben.
Ein XML Element besteht aus einem Start- und aus einem End-Tag. Das End-Tag ist durch den „/" gekennzeichnet. Zwischen Start und Ende-Tag können entweder

[3] w3c = Konsortium das Internet Standards entwickelt
[4] Metadaten = Daten über Daten

Zeichendaten, weitere Elemente oder einer Mischung aus Zeichendaten und Elementen stehen.

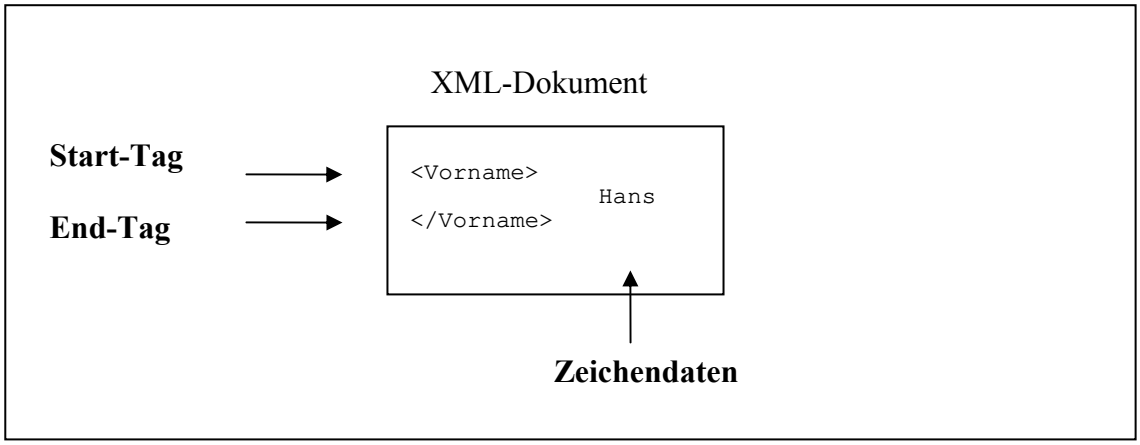


Abbildung 2-2 einfaches XML Beispiel

Elemente werden auch als Knoten bezeichnet. In der Regel enthält ein XML-Dokument verschachtelte Elemente (vgl. Abbildung 2-3 Beispiel Wohlgeformtheit). Durch diese Verschachtelung ergibt sich eine Baumstruktur.

2.1.1.3 Attribute

Elemente können Attribute besitzen. Sie haben die Form von `name="wert"`.

```
<Person geschlecht="m"></Person>
```

Enthält ein XML Element keine Zeichendaten, kann es als leeres Element ausgewiesen werden. Hierbei handelt es sich um eine verkürzte Schreibweise, in der das Start-Tag mit „/>“ beendet wird.

```
<Person geschlecht="w"/>
```

2.1.1.4 Wohlgeformtheit

Ein XML Dokument muss mindestens der Wohlgeformtheitsbeschränkung genügen (vgl. [I.1]).

- Ein Dokument hat ein oder mehrere Elemente
- Elemente bestehen aus einem Start und End-Tag
- die Elemente des Dokumentes korrekt sind korrekt geschachtelt
- es existiert ein Element (genannt Wurzelelement), das alle anderen Elemente umschließt

```
<Kontakte>
<Person>
      <Titel>     Dr. </Titel>
      <Name>
            <Vorname>   Max   </Vorname>
            <Nachname>  Mustermann  </Nachname>
      </Name>
</Person>
</Kontakte>
```

Abbildung 2-3 Beispiel Wohlgeformtheit

2.1.1.5 CDATA / Entitätsreferenzen

Wie zuvor beschrieben, erhalten bestimmte Zeichen (vgl. Tabelle 1) in XML eine besondere Bedeutung. Durch diese besondere Bedeutung dürfen diese Zeichen nicht mehr als Zeichendaten in einem XML-Dokument vorkommen. Der XML-Parser würde diese Zeichen falsch interpretieren. Um diese Zeichen trotzdem zu verwenden bietet XML folgende Möglichkeiten:

Entitätsreferenzen erlauben das Verwenden der Zeichen durch Maskierung d.h. die Zeichen sind durch die andere Zeichen ersetzt.

Zeichen	Maskierung
&	&amp;
<	<
>	>
"	&quote;
'	&pros;

Tabelle 1 Entitätsreferenzen

So genannte CDATA-Abschnitte `<!CDATA[ … ]]>` (vgl. [I.1]) können im Vergleich zu PCDATA jede Zeichenkombination enthalten (außer natürlich „>>]“, das das Ende des Abschnitts bedeutet). CDATA-Abschnitte können überall stehen, wo Zeichendaten stehen. Sie können jedoch nicht verschachtelt werden.

```
<!CDATA[
      <Nachname>  O'reilly </Nachname>
]]>
```

Abbildung 2-4 Beispiel CDATA

2.1.1.6 DTD

Dokumenttypdefinitionen (DTD) beinhalten Grammatiken, nach der ein XML-Dokument aufgebaut werden kann. In der Dokumenttypdefinition wird festgelegt, aus welchem Inhalt die XML-Datei bestehen darf. Genauer ausgedrückt: aus welchen Elementen ein Element besteht oder welchen Inhaltstyp es hat und welche Attribute mit welchen Werten vorkommen dürfen.

Eine DTD lässt sich mit einer Cobol Copy-Strecke bzw. mit einem Data-Dictionary.[5] vergleichen, dabei ist sie jedoch ungenauer[6] und wird nicht für die Ausgabeaufbereitung[7] verwendet.

Zeichen	Bedeutung
+	Ein oder mehrere Elemente
*	Kein, ein oder mehrere Elemente
?	Kein oder ein Element

Tabelle 2 Kardinalitäten in XML

```
<!ELEMENT Kontakte (Person*|Firma*)>
<!ELEMENT Person (Titel?, Name)>
<!ATTLIST Person
      geschlecht (m | w) #IMPLIED "m"
>
<!ELEMENT Name (Vorname+, Nachname)>
<!ELEMENT Titel (#PCDATA)>
<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Nachname (#PCDATA)>
<!ELEMENT Firma (#PCDATA)>
```

In diesem Beispiel werden Kontakte wir folgt definiert: Kontakte bestehen aus Personen oder Firmen. Eine Person kann über einen oder keinen Titel verfügen und muss einen Namen[8] - bestehend aus (mindestens) einem Vor- und (genau) einem Nachnamen - besitzen. Das Geschlecht einer Person kann als Attribut angegeben werden, erlaubt sind „m“ oder „w“ wobei „m“ als Vorgabewert vorgesehen ist. Als Titel, Vorname, Nachname, Firma können Zeichendaten (PCDATA[9])verwendet werden.

2.1.1.7 Gültigkeit

Ein XML Dokument ist gültig, wenn es den Regeln der DTD (vgl. S. 6) entspricht. Ein XML-Dokument „Kontakte“ ist beispielsweise ungültig, wenn eine Person keinen Namen besitzt.

---

[5] Eine Cobol-Copy Strecke bzw. Data-Dictionary beschreiben den Aufbau einer Datenstruktur

[6] beispielsweise lassen sich bei einer Cobol-Copy Strecke festlegen, wie viele Vor- und Nachkommastellen ein Zahlenwert hat oder nach welchem Format die Felder einer Datei gepackt sind.

[7] z.B. links- oder rechtsbündiges Ausrichten von Feldern für die Ausgabe auf einem Drucker.

[8] ohne Angabe der Kardinalität gilt eins als Vorgabewert. – Der Name ist also einmal vorgesehen.

[9] PCDATA = Parsed Character Data – Zeichendaten, sie dürfen bestimmte Zeichen nicht enhalten (vgl. Tabelle 1)

2.1.1.8 Parsen

Grundsätzlich werden zwei verschiedene Programmierschnittstellen für die Bearbeitung (parsen) d.h. den Aufbau und die Änderung von XML-Dokumenten im Hauptspeicher angeboten. Oft werden in der Presse diese Schnittstellen mit den einzelnen Parsern verwechselt. Die Schnittstelle eines Parsers beschreibt lediglich, welche Methoden zur Verfügung stehen. Der jeweilige Parser ist dann für die Umsetzung der Methoden zuständig. Er kann beispielsweise nach Performance- oder Kostengründen ausgewählt werden. Bei den verschiedenen Programmierschnittstellen handelt es sich um SAX und DOM.

- Das Simple API for XML (SAX) ist eine ereignisorientierte API. d.h. bestimmte Methoden werden aufgerufen, wenn Ereignisse stattfinden. (Ein Ereignis ist beispielsweise das Erreichen eines Start-Element).

  Das Durchlaufen des XML-Dokumentes bei SAX erfolgt sequentiell, somit ist lediglich der Zugriff auf das aktuelle Element möglich. Möchte der Entwickler auf weitere Elemente – als das aktuelle - zugreifen, muss er selbst für die interne Repräsentation sorgen.
- Beim Document Object Model (DOM) wird das XML Dokument vor der Bearbeitung komplett in den Hauptspeicher geladen und in einer Baumstruktur festgehalten. Diese Baumstruktur kann dann über die Methoden der DOM Schnittstelle rekursiv durchsucht werden.

Neben den unterschiedlichen Schnittstellen werden Parser auch noch danach unterschieden, ob sie ein XML-Dokument anhand einer DTD überprüfen, oder nicht. Dementsprechend werden sie validierende bzw. nicht-validierende Parser genannt.

XSL (XPath + XSLT)

Um ein XML Quelldokument in ein anderes - wie beispielsweise HTML (vgl. [I.17]) – zu überführen, bedarf es einer Transformation. Die Transformationsregeln werden in der XML Stylesheet Language Transformation (XSLT, vgl.[I.19]) festgelegt.
Um auf ein einzelnes Element bzw. Attribut zugreifen zu können, bedarf es der Angabe eines XML Pfades. Der XML Pfad wird in XPath (vgl. [I.16]) - der Pfadsprache von XML - festgelegt. XSLT und XPath zusammen bilden die XML Stylesheet Language (XSL, vgl. [I.18]).

Die XSLT-Transformationen werden dabei nicht in einer Programmiersprache wie z.B. Java durchgeführt sondern anhand so genannter Stylesheets. Diese beinhalten eine Menge von Regeln[10].

Prozessor	Implementierungssprache	Autor
MSXML	C++	Microsoft
Oracle XSLT-Prozessor	C++ und Java	Oracle
Xalan	C++ und Java	IBM / Apache Projekt (vgl. [I.20])
XML::XSLT	Perl	SourceForge
XT	Java	James Clark

Tabelle 3 Übersicht XSLT-Prozessoren[11]

Der grundlegende Ablauf ist einfach: Der XSLT-Prozessor (vgl. Tabelle 3) liest das Quelldokument Koten für Knoten ein und prüft, ob er eine anwendbare Regel findet. Wie aus Abbildung 2-5 zu entnehmen zeigt sich hier der Vorteil der Verwendung der XML-Technologie.
Es ist auf einfache Art und Weise möglich aus einem Quelldokument die gewünschten Zieldokumente zu erzeugen. Neben HTML und PDF können auf neue XML-Dokumente zum Datenaustausch zu anderen Systemen oder SVG-Grafiken dynamisch erzeugt werden. Ziel des Apache POI-Projektes (vgl. [I.15]) ist es, aus XML Microsoft Office Dateien – hauptsächlich Word und Excel – erzeugen zu können.

[10] Regeln werden auch als Templates bezeichnet
[11] (vgl. [Arci01], S. 256)

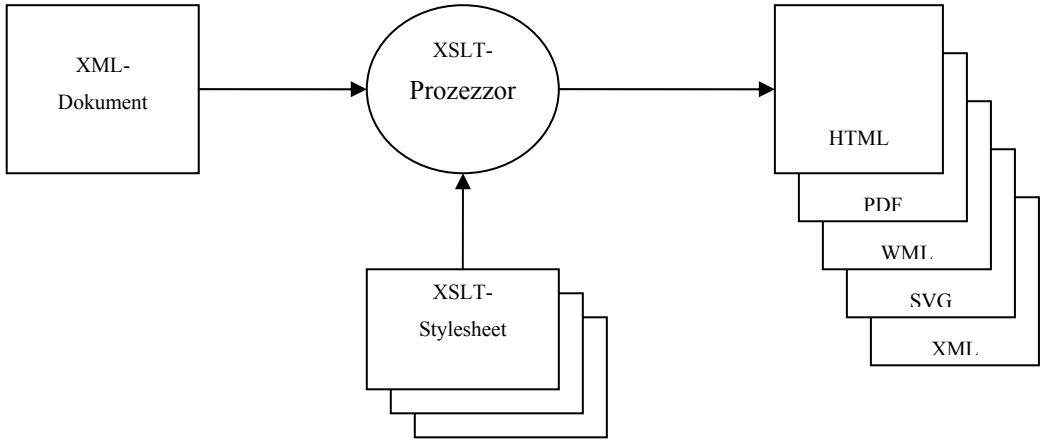


Abbildung 2-5 XSLT-Verarbeitung[12]

Ein XSLT-Dokument beginnt mit einem optionalen Prolog. Das folgende Beispiel verwendet die zu verwendende Stylesheet Version und den dazugehörigen Namensraum (xmlns = XML-namespace). Dieser Namensraum ermöglicht es, das die Regeln mit <xsl: eingeleitet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Da es sich bei einem XSLT-Dokument auch um ein XML-Dokument handelt, muss es wohlgeformt sein, d.h. es muss mit

```
</xsl:stylesheet>
```

enden. Zwischen diesen Elementen sind die Regeln einzufügen. Eine Regel wird wie folgt beschrieben:

```
<xsl:template match="/">
       <html> <head>
                     <title>Demo</title>
              </head>
              <body>
                     Ein einfaches Beispiel mit nur einer Regel
              </body>
       </html>
</xsl:template>
```

[12] in Anlehnung an [Arci01], S250

Match="/" ist ein XPath-Ausdruck[13] (vgl, [I.16]). Er sagt aus, dass diese Regel beim Wurzelknoten des Dokumentes angewendet wird. Im obigen Beispiel wird nur ein einfaches HTML-Dokument erzeugt, ohne das weitere Regeln verwendet werden.

Es existieren zwei unterschiedliche Arten, wie die Transformationsregeln angewendet werden können. Beim prozeduralen Ansatz werden die Knoten durch Verwendung von Schleifen durchlaufen, während beim rekursiven Ansatz die Transformation durch das Aufrufen der entsprechenden Regel ausgeführt wird.

```
<table>
<xsl:for-each select="Kontakte/Person">
<tr>
      <td>   <xsl:value-of select="Name/Nachname"/>        </td>
      <td>   <xsl:value-of select="Name/Vorname"/>         </td>
      <td>   <xsl:value-of select="Beruf"/>                </td>

</tr>
</xsl:for-each>
</table>
```

Dieses Beispiel verwendet einen prozeduralen Ansatz. Für jedes Person wird in einer Schleife (`<xsl: for-each ..>`) der Nachname, Vorname und der Beruf ausgegeben (`<xsl:value-of select="")`. Dabei werden die Elemente bzw. Knoten auf die sich die Regeln beziehen in Anführungszeichen gesetzt. Enthält ein Knoten Attribute – z.B. `<Person geschlecht="m">`, ist ein @ vor den Attributnamen zu setzen:

```
<xsl: value-of select="@geschlecht">
```

Wie auf die entsprechenden Knoten zugegriffen wird, ist in XPath - der Pfadsprache von XML- festgelegt.

Wie das folgende Beispiel zeigt, verwendet der rekursive Ansatz keine Schleifen. Durch `<xsl: apply-templates >` werden die weiteren Regeln angewendet.

```
<xsl:template match="/">
<html>       <head><title>links</title></head><body>
      <table>
            <xsl:apply-templates select="*"></xsl:apply-templates>
      </table></body></html>
</template>

<xsl:template match="Person">
<tr>
      <xsl:apply-templates select="Name"/>
      <xsl:apply-templates select="Beruf"/>
</tr>
</xsl:template>
```

[13] XPath Ausdrücke sind an den doppelten Anführungszeichen zu erkennen

```
<xsl:template match="Name">
       <td>   <xsl:value-of select="Nachname"/>        </td>
       <td>   <xsl:value-of select="Vorname"/>        </td>
</xsl:template>

<xsl:template match="Beruf">
       <td>          <xsl:value-of select="."/></td>
</xsl:template>
```

Sollen zu einem Knoten zwei verschiedene Regeln angewendet werden, kann der Parameter mode verwendet werden. Der muss dazu sowohl beim Regelaufruf (`<xsl:apply-templates select="Beruf" mode="blau"/>`) als auch in der Regel (wie folgt) verwendet werden.

```
<xsl:template match="Beruf" mode="blau">
       <td bgcolor="blue"> <xsl:value-of select="."/></td>
</xsl:template>
```

Zur Wiederverwendung und / oder um bei einer großen Anzahl von Regeln den Überblick nicht zu verlieren, lassen sich Regeln auf verschiedene Dateien aufteilen. Um diese Regeln dann zu verwenden, können Sie mittels

```
<xsl:import href="http://localhost/import.xsl"></xsl:import>
```

in das aktuelle Dokument einbezogen werden[14].

[14] alternativ lässt sich auch ein Verweis auf das lokale Dateisystem mit z.B `href="file:///c://import.xsl"` erstellen

2.1.2 Muster[15]

Am Anfang einer Entwicklung steht immer ein Problem, das gelöst werden muss. Wenn Menschen ein Problem haben, wenden sie sich an einen Experten um von seinen Erfahrungen, seinem Wissen und seinen Problemlösungen zu profitieren. Doch was passiert, wenn der Experte dieses Problem nicht lösen kann? Er schaut sich nach Lösungen seiner Kollegen um, um von ihren Ideen und Lösungen zu lernen und Fehler die gemacht worden sind zu vermeiden.

Dieses Prinzip steckt auch hinter Muster.

Ein Muster ist eine abstrakte Problemlösung eines Problems, die auf gesammelten Erfahrungen von Experten beruht. Als Begründer von Mustern gilt der Architekt C. Alexander. Er definiert Muster wie folgt:

„Jedes Muster beschreibt ein in unserer Umwelt beständig wiederkehrendes Problem und erläutert den Kern der Lösung für dieses Problem, so daß Sie diese Lösung beliebig oft anwenden können, ohne sie jemals ein zweites Mal gleich auszuführen."

Ralf Johnson in [Fowl99] über die Verwendung von Muster:

„Sie sind nicht unbedingt die offensichtlichsten Lösungen der Modellierungsprobleme, dennoch scheinen sie mir glaubhaft. … und sie haben einfach funktioniert."

Es lassen sich folgende Muster unterscheiden:

- Architektur Muster
- Entwurfsmuster
- Idiome

[15] engl. Pattern

2.1.2.1 Architektur Muster

Ein Architektur Muster[16] drückt ein fundamentales Organisationsschema für Software-Systeme aus. Es stellt ein Set von vordefinierten Subsystemen bereit, legt deren Verantwortung fest, und fügt Regeln und Richtlinien für die Organisation der Beziehungen zwischen ihnen hinzu. Das bekannteste Architektur Muster ist das MVC-Muster (siehe S. 14)

2.1.2.2 Entwurfsmuster

Ein Entwurfsmuster[17] stellt ein Schema für die Verfeinerung der Subsysteme oder Komponenten eines Software Systems oder Beziehungen unter ihnen zur Verfügung. Es beschreibt eine wiederkehrende Struktur von kommunizierenden Komponenten, dass ein generelles Design Problem in einem bestimmten Kontext löst.
„Auf den Punkt gebracht: Entwurfsmuster helfen Entwicklern, einen Entwurf schneller richtig zu machen“ (vgl.[Gamm96], S. 2)

Gamma definiert in seiner „Entwurfsmusterbibel“ einen Musterkatalog von 23 Designmustern (vgl. Tabelle 4) im Wesentlichen unterteilt er drei Kategorien:

- Erzeugermuster
  Erzeugermuster helfen dabei ein System unabhängig davon zu machen, wie seine Objekte erzeugt und repräsentiert werden.
- Strukturmuster
  Strukturmuster befassen sich damit, wie Objekte und Klassen zu größeren Strukturen zusammengefasst werden können.
- Verhaltensmuster
  Verhaltensmuster befassen sich mit der Interaktion zwischen Objekten und Klassen. Sie beschreiben komplexe Kontrollflüsse, die zur Laufzeit schwer nachvollziehbar sind.

[16] engl. Architectural Pattern
[17] engl. Design Pattern

		Aufgabe		
		Erzeugermuster	**Strukturmuster**	**Verhaltensmuster**
Gültigkeits-bereich	**Klassenbasiert**	Fabrikmethode	Adapter	Interpreter Schablonenmethode
	Objektbasiert	Abstrakte Fabrik Erbbauer Prototyp Singleton	Adapter (objektbasiert) Brücke Dekorierer Fassade Fliegengewicht Komposition Proxy	Befehl Beobachter Besucher Iterator Memento Strategie Vermittler Zustand Zuständigkeitseffekte

Tabelle 4 Dimension der Ausprägung von Entwurfsmustern[18]

2.1.2.3 Idiome

Ein Idiom ist ein „Low-Level“ Muster, spezifisch für *eine Programmiersprache*. Es beschreibt, wie einzelne Aspekte von Komponenten oder die Beziehungen zwischen diesen, die die Features einer gegebenen Programmiersprache nutzen, zu implementieren sind.

2.1.3 verwendete Muster in Evolution

Als Teil der Grundlagen werden die in Evolution (vgl. Kapitel 5) verwendeten Muster bereits an dieser Stelle erläutert.

2.1.3.1 Model-View-Controller Muster (MVC)

Das Model-View-Controller Muster ist ein aus Smalltalk bekanntes Architektur Muster. Die verschiedenen Schichten des Musters haben dabei folgende Bedeutung:

- Model

  Das Model beinhaltet die eigentlichen Daten des Musters. Innerhalb einer Evolutionanwendung kapseln die Geschäftsobjekte (siehe S. 59), die Daten aus den verschiedenen Backendsystemen. Ein Teil dieser Daten wird in Anzeigeobjekte(siehe S. 59) überführt. Hierbei handelt es sich um die Daten die zum Ende einer Anfrage im Browser angezeigt werden z.B. der Name eines Kunden.

[18] vgl. [Gamm96]

Evolution verwendet das Model also auf zwei Arten – einmal innerhalb der Geschäftsobjekte und einmal in Form der Anzeigeobjekte und dem daraus resultierenden XML-Baum im Hauptspeicher.

- View (Sicht)

  Die View beinhaltet die Repräsentation der Daten, die aus dem Model stammen. Wie oben bereits beschrieben erstellt Evolution ein XML-Dokument. Dieses XML-Dokument wird über XSL-Dateien[19] in das entsprechende Ausgabeformat (die View) überführt. Derzeit können HTML und PDF Dokumente erstellt werden. Ebenfalls ist es anhand einer XSL-Datei möglich, das XML-Dokument umzustrukturieren, damit evtl. GUI-Clients (z.B. Visual Basic) bedient werden können, die ihrerseits für die Transformation der Daten sorgen.

- Controller (Steuerung)

  Der Controller hat die Aufgabe auf die Benutzereingaben zu reagieren und für den Programmablauf zu sorgen. In Evolution existieren genau genommen zwei Controller.

  Ein Java-Servlet steuert die generelle Abarbeitung einer Anfrage (beispielsweise durch Eingabe einer URL im Browser). Dazu werden die vom Benutzer eingegebenen Parameter ausgelesen und der zu Startende Prozess ermittelt.

  Der Prozess wird durch die vorgegebene Workflowelemente (Prozess, Vorgang, Aktivität - vgl. Abbildung 5-6) gesteuert. Die Reihenfolge der Prozessabarbeitung ist vorgegeben. Zur Steuerung des Workflows werden Zustände verwendet.

Die Trennung der Schichten in Model, View und Controller bietet folgende Vorteile:

- Wiederverwendung

  Für verschiedene Views (PDF, HTML weiter denkbar wären Anzeige auf einem Handy oder einem PDA[20] wie beispielsweise ein PALM) kann ein Model verwendet werden.

- Geringerer Wartungsaufwand / Austauschbarkeit

[19] Sie enthalten die XSLT-Regeln sowie die XPath-Ausdrücke
[20] Personal Digital Assistant

Wird ein Fehler im Model beseitigt, profitieren automatisch die verschiedenen Views von der Änderung. Im Idealfall kann man ein Model austauschen, ohne Änderungen in der View durchzuführen.

- parallele Bearbeitung

  Model und View können – nach Beschreibung der Schnittstellen – getrennt von einander entwickelt bzw. erweitert werden und somit auf verschiedene Projektmitarbeiter verteilt werden.

2.1.3.2 Fabrikmethode (Factory)

Wie aus Tabelle 4 zu entnehmen ist, handelt es sich bei der Fabrikmethode (alternativer Begriff auch „Fabrik“) - innerhalb der Entwurfsmuster- um ein Erzeugermuster.
Die Fabrikmethode bietet eine Schnittstelle mit Operationen zum Erzeugen eines Objektes an, wobei der eigentliche Erzeugerprozess an die Unterklasse delegiert wird.
Die Fabrikmethode lässt sich parametrisieren. Evolution nutzt diesen Mechanismus um mandantenabhängig Objekte zur Laufzeit zu erzeugen.

Vorteile:

Fabrikmethoden ermöglichen es, dass anwendungsspezifische Klassen nicht in den Frameworkcode eingebunden werden müssen.

Die Erzeugung von Unterklassen ist flexibeler – es können weitere, spezialisiertere Unterklassen - ohne großen Aufwand – eingeführt werden

```
// Auszug aus dem Steuerungsservlet "EvolutionServlet"
Session evolutionSession = SessionFactory.create(„evolutionSession");

//SessionFactory wird anhand des Mandanten ermittelt, welche Session
public class SessionFactory{ …
      /* Klassenname zur Laufzeit aus Konfiguration lesen und somit
               dynamisch bestimmen
      */
      String className =
            Configuration.readAttribute(serviceId,Session.SESSION);
      Class clazz = Class.forName(className);
        java.lang.reflect.Constructor constructor =
                        clazz.getConstructor(new Class[]  });
        return (Session) constructor.newInstance(new Object[]{});
..}
}
```

Abbildung 2-6 Beispiel Fabrikmethode

2.1.3.3 Befehlsmuster (command)

Das Befehlmuster (ein Verhaltensmuster) kapselt, einen Befehl als Objekt. Dies ermöglicht es, Klienten mit verschiedenen Anfragen zu parametrisieren, Operationen in eine Queue zu stellen, ein Logbuch zu führen und Operationen rückgängig zu machen (vgl. [Gamm96], S. 273).
Das Befehlsmuster ermöglicht es Steuerelementen (Controls) – in Evolution Prozess sowie Vorgang - Anfragen an unbekannte Anwendungsobjekte zu richten (z.B. Aktivitäten), indem es die Anfrage selbst zu einem Objekt macht. Dreh- und Angelpunkt dieses Musters ist die abstrakte Klasse *Command*. (vgl. beiliegende CD)

Die Java-API bietet neben Servlets und Applets zahlreiche weiter Beispiele für das Befehlsmuster, in dem es den Aufbau der abzuleitenden Klassen vorgibt (bei Applets werden die Methoden init(), start(), stop() und destroy() vorgegeben).

2.1.4 Framework

Gamma in ([Gamm96]) über Framework:
„Ein Framework bestimmt die Architektur einer Anwendung. Es definiert die Struktur im Großen und seine Unterteilung in Klassen und Objekte, die jeweiligen zentralen Zuständigkeiten, die Zusammenarbeit der Klassen und Objekte sowie den Kontrollfluss. Ein Framework legt die Entwurfsparameter im voraus fest, so dass der Anwendungsentwickler und Programmierer sich auf die spezifischen Details kümmern kann. Frameworks betonen somit die Entwurfswiederverwendung gegenüber der Codewiedergewinnung. „

H. Balzert (vgl. [Balz99]) erweitert im den Frameworkbegriff und gibt Auskunft darüber, was von einem Entwickler – dem Anwender des Frameworks erwartet wird:
„Es besteht aus konkreten und – insbesondere – aus abstrakten Klassen, die Schnittstellen definieren…Im allgemeinen wird vom Anwender des Frameworks erwartet, dass er Unterklassen definiert um das Framework zu verwenden und anzupassen.“

Im Unterschied zur prozeduralen Entwicklung übernimmt ein Framework also die Steuerung des Kontrollflusses:

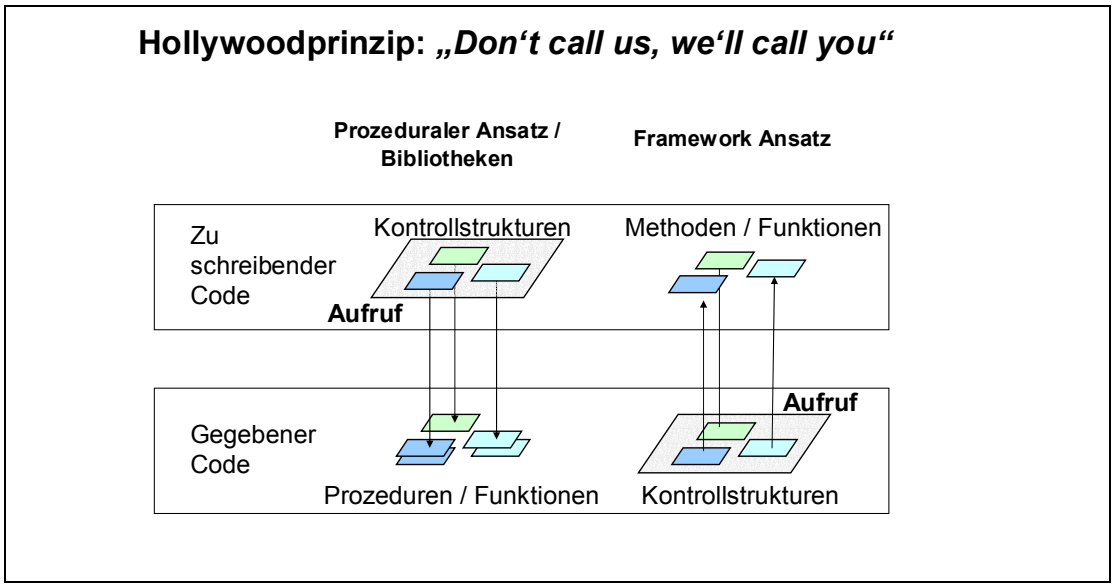


Abbildung 2-7 Framework Ansatz

2.1.4.1 Abgrenzung zwischen Framework und Muster

Frameworks und Muster basieren auf einem ähnlichen Prinzip – ihnen liegen jeweils Erfahrungen ihrer Entwickler zu Grunde und werden genutzt um den Entwurf einer Anwendung zu bestimmen. Deshalb weißt Gamma in [Gamm96], S 38 auf die Unterschiede zwischen Frameworks und Entwurfsmuster hin:

1.) Muster sind abstrakter als Frameworks.
   Frameworks werden als Code vorgegeben. Muster hingegen können nur beispielhaft als Code repräsentiert werden.
2.) Muster sind kleiner als Frameworks.
   Ein Framework verwendet typischerweise mehrere Muster, die Umkehrung gilt nie.
3.) Muster sind weniger spezialisiert als Frameworks.
   Frameworks haben immer einen bestimmten Anwendungsbereich – Evolution ist z.B. für Internetapplikationen bestimmt. Muster hingegen können für nahezu alle Anwendungsbereiche verwendet werden.

2.2 Fachliche Grundlagen

2.2.1 Geschäftsprozess

Nach dem Erscheinen des vielzitierten, richtungweisenden Buch der Amerikaner Hammer und Champy zur Reorganisation der Unternehmen ist eine breite Diskussion um den Begriff Geschäftsprozess (Business Process) mit einer Vielzahl unterschiedlicher Definitionen entstanden.
Mehrheitlich versteht man unter einem Geschäftsprozess eine Folge von logisch zusammenhängenden Vorgängen, Aktivitäten oder Arbeitsschritten, die für das Unternehmen einen Beitrag zur Wertschöpfung leistet.

2.2.2 Anwendungsfall

Ein Anwendungsfall oder der auch in der deutschen Literatur verwendete Begriff Use Case besteht aus mehreren, zusammenhängenden Aufgaben, die von einem Akteur durchgeführt werden. Ein Akteur ist eine Rolle, die ein Benutzer des Systems spielt. (vgl. [Balz99])

Jacobson (vgl. [JBR99]) unterscheidet zwischen einem Use Case in einem Unternehmen und Use Case in einem Informationssystem. Beim Use Case im Unternehmen soll die ausgeführte Aufgabe für das Unternehmen von einem messbaren Wert sein und kann weitere organisatorische Schritte enthalten. Der Use Case in einem Interaktionssystem spezifiziert die Interaktionen zwischen einem Akteur und dem System, d.h. er beschreibt die Benutzung des Systems.

Ostereich (vgl. [Oest98], S. 124) beschreibt einen Anwendungsfall wie folgt:
„…die Interaktionen zwischen dem Anwender und dem Anwendungssystem, die notwendig sind, um einen Arbeitsgang durchzuführen. … Ein Anwendungsfall sollte beschreiben, was ein Benutzer zu einem Zeitpunkt an einem Anwendungssystem macht, um einen Geschäftsvorfall in einem Geschäftsprozess abschließend zu bearbeiten.“

2.2.3 Mandantenfähigkeit

Unter Mandantenfähigkeit versteht man die Nutzung einer Plattform/Architektur für mehrere Benutzergruppen (Mandanten). Ein Mandant könnte z.B. eine Firma sein, die Funktionalitäten von einer anderen Firma verwenden möchte und dafür die gleiche Anwendung nutzen möchte. Beispielsweise nutzt die Firma Consors für die Aktienanalysen und Detailansichten die Plattform der Firma onVista. Mandanten können aber auch innerhalb eines Unternehmens existieren. So kann der Innendienst eines Unternehmens im Vergleich zum Außendienst eine erweiterte Funktionalität einer Anwendung nutzen.
Eine wichtige Anforderung an eine mandantenfähige Architektur ist, dass sie so flexibel ist, dass sie verschiedene Mandanten mit unterschiedlichen Funktionalitäten, ohne großen Aufwand und Veränderungen im Workflow, integrieren kann. Dabei sollte eine größtmögliche Wiederverwendung von vorhandener Funktionalität gewährleistet sein.
Die Mandantenfähigkeit wird in Evolution der hier vorgestellten Architektur durch zwei unterschiedliche Vorgehens- und Implementierungsmuster umgesetzt.
Zum einen wird das Model-View-Control (vgl. S. 14) Muster dafür verwendet, die Ansicht von der Steuerung und der Logik zu trennen. Zum anderen unterstützt das Fabrikmuster die Mandantenfähigkeit (siehe S.16).

An einem Beispiel soll die Mandantenfähigkeit der Architektur erläutert werden:

Ein Versicherungsunternehmen verkauft Lebensversicherungen. Es möchte ein System installieren, das den Innendienstmitarbeitern erlaubt, Lebensversicherungen an Kunden zu verkaufen und diese Abschlüsse direkt in die Systeme der Versicherung zu übertragen. Außerdem soll ein Kunde im Internet die Möglichkeit haben, aus dem Internet selbst eine Lebensversicherung bei dem Unternehmen abschließen zu können. Beide Anwendungen sollen auf Basis der hier vorgestellten Architektur realisiert werden. Hierbei sind der Innendienst und die Internetkunden die Mandanten.
Der Prozess eine Lebensversicherung zu verkaufen ist bei beiden Mandanten gleich. Der Vorgang einen Antrag zu erstellen unterscheidet sich nicht. Erst bei den Aktivitäten ist ein Unterschied erkennbar. Am Beispiel der Provisionsdatenerfassung wird gezeigt, wie das Fabrikmuster eingesetzt wird. Im Vorgang „Antrag erstellen“ wird die Aktivität Provisionsdaten erfassen aufgerufen. Aus Sicht des Workflows ist egal, um welche Provisionsdatenerfassung es sich handelt. Die Fabrik entscheidet, welches Produkt

instanziert wird. Für den Innendienst wird eine Aktivität erzeugt, bei dem die Provisionen einem Mitarbeiter zugeordnet und berechnet werden, während beim Internetkunden erzeugte Aktivität automatisch die Provisionsdaten festlegt. Andere Aktivitäten, wie z.B. das Erfassen der Adresse der Kunden, werden bei den beiden Mandanten nicht unterschieden und können von beiden verwendet werden.

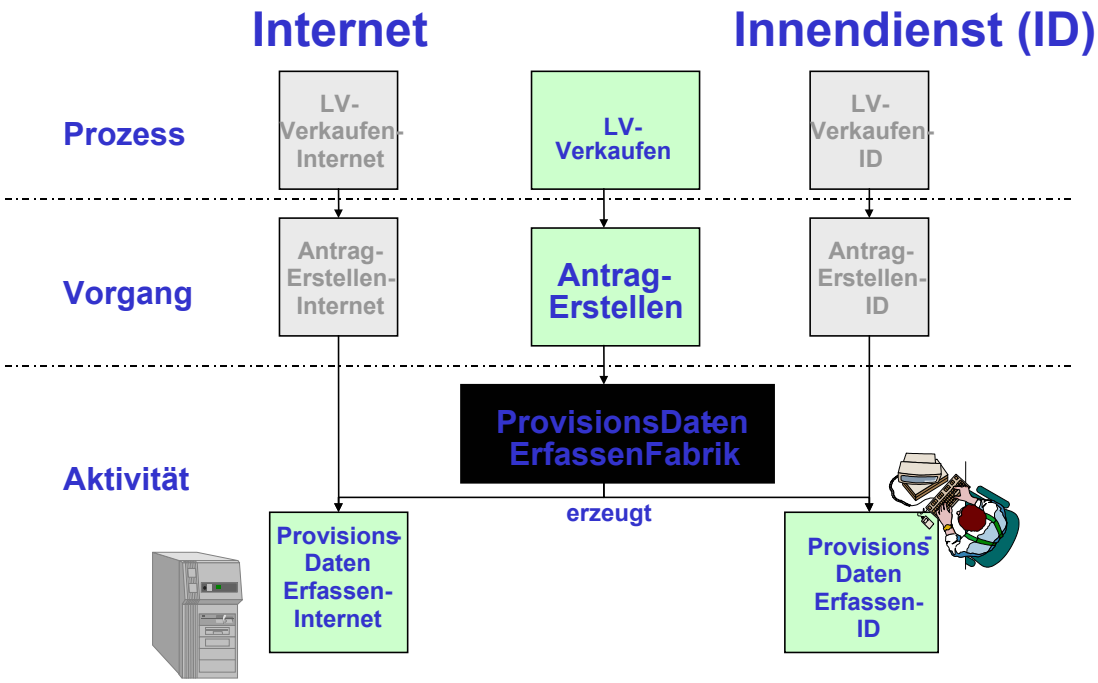


Abbildung 2-8 Beispiel Mandantenfähigkeit

Das Schaubild zeigt, wie die hier vorgestellte Architektur funktioniert. Die grau hinterlegten Objekte entsprechen dabei einer herkömmlichen Entwicklung, bei der für jeden Mandanten ein eigener Workflow implementiert wird und somit jeder Mandant eine eigene Anwendung darstellt.

3 Portale

3.1 Definition

Es gibt zahlreiche Erklärungen zu dem Begriff „Portal“. Während Meyers Taschenlexikon von einem „monumentalen Eingang“ ausgeht und der Duden von einem „prunkvollen Tor“ spricht, handelt es sich im Bereich der Informationstechnologie um eine Webseite, die als Einstieg ins Internet dient.

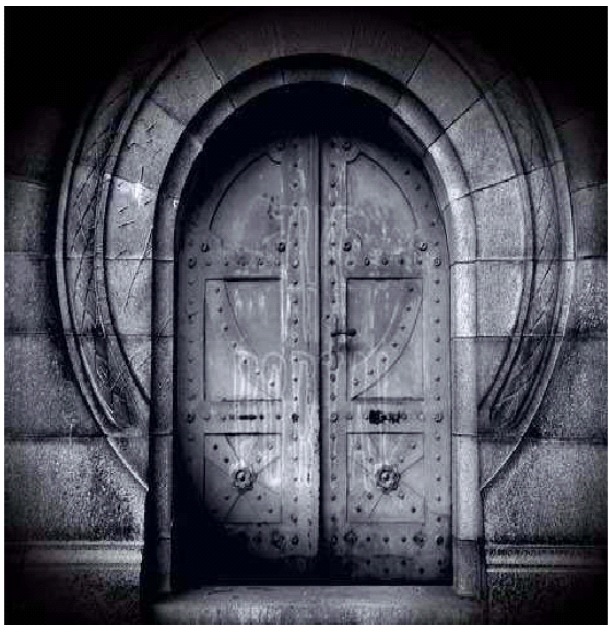


Abbildung 3-1 Monumentaler Eingang

Doch worin unterscheidet sich Portal von einer normalen Webseite?

In einer Analyse [21] der Meta Group wird der Begriff „Personalisierung“ aufgenommen:

„Ein Portal dient als zentraler Einstiegspunkt für die Benutzer eines Inter-/Intranets oder Extranets und aggregiert und/oder personalisiert sowohl strukturierte als auch unstrukturierte Daten aus unterschiedlichsten Quellen innerhalb eines Unternehmens.“

[21] Der Markt für Portale, Marktplätze und Mobile Commerce in Deutschland

Während es bei den bisherigen Definitionen „nur“ um die Anzeige der Daten geht, geht IBM einen Schritt weiter, in dem es die Applikationen einbezieht:

“A portal provides a single point of access to diverse Information and applications, a customizable interface, personalized content, and much more.”

Chuck Kao nimmt diesen Gedanken in seinem Artikel „Enterprise Application Portals“[22] auf und unterscheidet diesbezüglich zwischen Enterprise Application Portals(EAP) und Enterprise Information Portals (EIP). Während EIP’s „lediglich“ Informationen aus Datenbanken darstellen, um zur Entscheidungsfindung beizutragen, erläutert er ferner folgende kritische Merkmale:

- Workflow enabled – sie unterstützen komplexe Geschäftsprozesse
- Flexible – sie können einfach konfiguriert und so den sich ändernden Anforderungen angepasst werden
- Extensible – neue Funktionen können einfach integriert werden
- Portable – sie sind unabhängig vom darunterliegenden Betriebssystem lauffähig
- Scalable – sie unterstützen viele Benutzer und Anwendungen
- Secure – sie ermöglichen einen sicheren Zugang

3.2 Generationen

Die Gartner Group stellt in ihrem Artikel „Information Portals: Opportunity for BI Vendors in Europe?“ „Generationen“ von Portalen vor:

3.2.1 Portale der 1. Generation: Internet Entry Point

Portale der ersten Generation bieten einen zentralen Einstiegspunkt zum Unternehmen und zu Ressourcen aus dem Internet.

3.2.2 Portale der 2. Generation: Content Integration

Portale der zweiten Generation ermöglichen den Zugriff auf interne und externe Daten, die zur Entscheidungsfindung im Unternehmen beitragen. Erweiterte Suchmöglichkeiten und Personalisierung kommen hinzu.

[22] Enterprise Application Portals, eAI Journal, Februar 2001

3.2.3 Portale der 3. Generation: Application Integration

Portale der dritten Generation ermöglichen Zugriff und Support für den virtuellen Arbeitsplatz im Internet. Gartner bezieht hier neben E-Mail, Groupware, ERP, CRM Anwendungen auch Datawarehouse und Transaktionssysteme ein.

3.2.4 Portale der 4. Generation: Outward Integration

Portale der vierten Generation ermöglichen business-to-business, Zusammenarbeit in Produkt-Design oder Herstellung, Supply-Chain-Management, e-commerce Integration und weitere organisationsübergreifende Geschäftsprozesse.

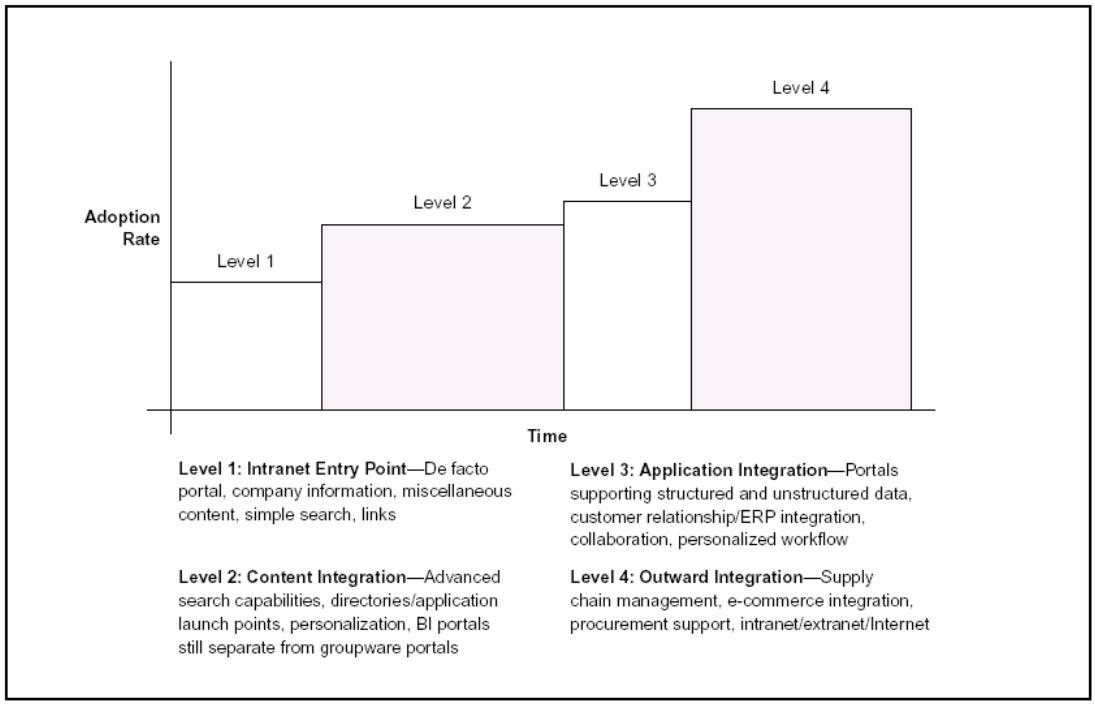


Abbildung 3-2 Portalgenerationen[23]

[23] Quelle: Dataquest, December 1999

3.3 Portalarten

3.3.1 Webportale

Webportale – oder auch Allgemeine bzw. Mega-Portale genannt vgl. - sind aus den verschiedenen Suchmaschinen hervorgegangene Websites (z.B. yahoo, comundo). Sie sind am ehesten den Portalen der ersten Generation zuzuordnen, da sie das Informationsangebot des Internets zusammenfassen. Webportale zeichnen sich dadurch aus, dass sie zahlreiche Benutzerzahlen bewältigen können und zu den verschiedenen Themenbereichen externe Links verwalten.

3.3.2 Fach- und Themenportale

Aufgrund der Informationsflut wird es für Webportale immer schwieriger Informationen strukturiert zusammenzufassen. Aus dieser Problematik sind Fach- und Themenportale – hervorgegangen. Sie werden auch als vertikale Portale oder Vortale (vgl. [I.14]) bezeichnet.
Sie konzentrieren sich auf ein oder mehrere Interessensgebiete und bieten bestimmten einer bestimmten Nutzergruppe Daten und Anwendungen.

Dabei veröffentlichen Sie beispielsweise

- Artikel, Studien, Trends, Leitfaden, Fallbeispiele und ermöglichen
- Dialoge mit Top-Experten und hochkarätigen Nutzern oder
- Networking und Anbahnungen von Geschäftskontakten

Die „competence-site“ (vgl. [I.3]) bietet etwa „Unterstützung bei der täglichen Arbeit“ und Weiterbildung in den Bereichen Management, Recht, IT-Systeme und Spartenwissen und zeigt, dass Fach- bzw. Themenwissen nicht kostenpflichtig sein muss.

3.3.3 Wissensportale

Wissensmanagement[24] ganz allgemein befasst sich mit dem Einsatz von Wissen, um Geschäftsziele zu erreichen, die Innovationsfähigkeit eines Unternehmens zu fördern und die Produktivität der Mitarbeiter zu verbessern.
Leistungsfähige Wissensportale nutzen die Infrastruktur von Web-Anwendungsservern und Technologien aus den Bereichen Datenbank- und Informationsmanagement.
Kernbestandteile eines Wissensportal (vgl. [I.21]) sind:

- Funktionen zur Erstellung und Verwaltung von Profilen über Benutzer und Interessengemeinschaften, strukturiert nach Kenntnissen, Fähigkeiten und Tätigkeitsfeldern. Das Portal sollte sich individuell konfigurieren und erweitern lassen, um beispielsweise Anschluss an ERP-Systeme oder auch weitere virtuelle Interessengemeinschaften herzustellen. Eigene virtuelle Treffpunkte lassen sich erstellen.
- Weitere Funktionen umfassen die Bereiche Lokalisieren von Expertise und einen so genannten „Content Catalog“. Damit lassen sich Antworten auf Fragen finden wie: Wer in der Organisation verfügt über welches Know-how? An welchem Ort finden sich die gesuchten Unterlagen? Welche Informationen sind relevant? Der Content Catalog produziert und aktualisiert einen Lageplan mit Themen, die für eine Wissensmanagement-Anwendung relevant sind. Zum Einsatz kommen hier Techniken, die eine Inhaltsanalyse vornehmen, die ermitteln, wer wie oft welche Informationen abfragt und die ein Beziehungsgeflecht zwischen Informationen aufbauen.

3.3.4 Kollaborationsportale[25]

Bei Kollaborationsportalen (vgl. [I.33]) steht die Unterstützung der verteilten Gruppenarbeit im Mittelpunkt. Es soll die Kommunikation, Kooperation und Koordination zwischen den Teammitgliedern verbessert werden. Typisch sind gemeinsame Arbeitsbereiche, auf die Teammitglieder von verschiedenen Standorten zugreifen und in denen sie Dokumente ablegen und bearbeiten können. Des Weiteren finden sich in Kollaborationsportalen zumeist Aufgabenlisten, Diskussionsforen, Dokumentenmanagement- Funktionen und weitere Funktionen zur Publikation und Verbreitung von Informationen.

[24] engl. Knowledge Management (KM)
[25] engl. Collaborative Portals

Anbieter kommen aus den Bereichen Groupware und Dokumentenmanagement. Als solche sind z.B. Documentum, Lotus und Open Text zu nennen.

3.3.5 Unternehmensportale

Unternehmensportale fungieren als Einstiegspunkt, der verschiedene Informationen, Anwendungen, firmeninterne sowie firmenexterne Dienstleistungen verbindet. Sie lassen sich anhand ihres Nutzerkreises in folgende Gruppen unterteilen:

- B2B
- B2C
- B2E

3.3.5.1 B2B

Ein „Business-to-Business“- Portal (oder auch Transaktionsportal) ist ein auf eine Branche spezialisiertes vertikales Portal, das ausschließlich geschäftlichen Zwecken dient.

T. Kirchmaier stellt in([MP02], S.53ff.) das Händlerportal „Volkswagen (VW) PartnerNet“ vor, das die vorherige Kommunikation via Fax, Telefon, Brief und E-Mail zwischen VW und den Händlern ablöst bzw. ergänzt.

Volkswagen sammelt in diesem Portal[26] Produkt-, Marketing- und Vertriebsinformationen, Erfahrungen und Fragen der Händler. Die Händler können Informationen bewerten, online Fragen stellen, Probleme melden und erhalten Zugriff auf die Volkswagen „Wissensbasis“, aktuelle Meldungen als „Must Read“ sowie den Status Ihrer Anfragen.

Das „PartnetNet“ ermöglicht Volkswagen im Gegenzug eine einfache Bearbeitungsmöglichkeit der Händleranfragen sowie erweiterte Analysemöglichkeit der Aufgetretenen Probleme.

3.3.5.2 B2C

B2C-Portal „Business-to-Consumer“-Portale sind Portale die sich vom Unternehmen an den Konsumenten richten. Diese Portale wollen Konsumgüter verkaufen, ihre Zielgruppe sind daher Käufer oder potenzielle Käufer dieser Güter. Die laufenden Betreuung des Kunden durch EEC („Elelctronic Customer Care“) und CRM („Customer Relationship Management“) steht im Vordergrund dieser Angebote.

[26] ca. 30.000 Nutzer

Einen ähnlichen Weg wie Volkswagen im B2B, geht die SKW Schoellerbank im B2C Bereich. (vgl. [I.41]) Sie bietet Ihren Kunden die Möglichkeit ihre „myBank“ zu personalisieren d.h. besonders Informationen aus verschiedenen Bereichen auszuwählen, Anwendungen auszuwählen sowie verschiedene Depots zu verwalten, Termine mit den Kundenberatern zu vereinbaren etc.

3.3.5.3 B2E

B2E-Portal, ein „Business-to-Employee“- Portal (vom Unternehmen zum Angestellten) nutzt das Intranet zur Optimierung der unternehmensinternen Geschäftsprozesse. B2E-Portale stellen den Einstieg der Mitarbeiter für die Web-Anwendungen ihres Unternehmens. Der Kontakt zwischen und zu den Mitarbeitern soll verbessert und den Angestellten die Arbeit durch den Zugriff auf aufgabenspezifische Informationen (dazu gehört das interne Telefonbuch genauso wie ein firmenweites Lernportal) erleichtert werden. Das B2E-Portal wird oft im Zusammenhang mit einem Wissensportal (Welcher Mitarbeiter hat welche Qualifikation bzw. welchen Tätigkeitsbereich) oder mit einem Kollaborations-Portal genannt

3.3.6 Gründe für den Einsatz von Unternehmensportalen

Zusammenfassend lassen sich folgende Gründe für den Einsatz von Portalen(vgl. [Meta00]) und besonders Unternehmensportalen darstellen:

- Verringerung der Bearbeitungszeiten und –kosten
  durch die Orientierung an Geschäftsprozessen und integrierten Zugriffen auf verschiedene Backendsysteme über nur ein Portal. Die Prozesseffizenz kann verringert werden (vlg.[ÖFA02])
- Flexibilitätszuwachs
  Die Flexibilität wird erhöht, da über ein Portal schnell wechselnde Anforderungen an elektronische Produktkataloge bedient werden können (vlg.[ÖFA02])
- Verbesserung der Teamarbeit
  durch eine geschäftsprozessorientierte Aufgabenverteilung und Funktionen zur Kommunikationsuntersützung
- Geringerer Schulungsaufwand
  für die Mitarbeiter. Sie greifen über das ihnen bekannte Portal auf die Backendsysteme zu, ohne sich mit deren Benutzungskonzepten direkt auseinandersetzen zu müssen.

- Externalisierung
  Die Unternehmen versuchen, den Kunden und Geschäftspartnern Mehrwertdienste zur Verfügung zu stellen um ihn in Pre- und Aftersales Aktivitäten zu unterstützen.
- Umgang mit der Informationsflut
  Neben dem Internet existieren auch im Intranet eine große Anzahl von Webseiten. Durch Kategorisierung und Kanalisierung können die wichtigen Informationen den Interessenten gezielt zugeordnet werden.
- Wissensmanagement
  Portale sind eine Möglichkeit um Wissensmanagementstrategien zu implementieren.

3.3.7 Zusammenfassung

Anhand der Erklärungen und die einleitenden Definitionen scheint eine genaue Abgrenzung von Portalen schwierig, wenn nicht sogar unmöglich. Ferner soll in dieser Arbeit kein Versuch unternommen werden, weitere Portalbegriffe einzuführen. Ein Portal zeichnet sich letztendlich durch den Aufgabenbereich aus, für den das Portal entwickelt wurde. Diese Aufgabenbereiche lassen sich durch Portlets – also Anwendungen im Portal - einfach erweitern. Ebenso wie eine Webseite, verrät ein „monumentaler“ Eingang nichts darüber, was hinter dem Eingang verborgen ist – dies wird immer erst ersichtlich, wenn man den Eingang durchschreitet – sich authentifiziert - und für weitere Räume zugelassen „autorisiert“ wird.

Deshalb wird im folgenden Kapitel der Blick auf die wichtigsten Portalfunktionen gerichtet. Als Portalfunktionen werden die erweiterten Funktionalitäten eines Portals im Gegensatz zu einer „normalen“ Website erläutert.

3.4 Die wichtigsten Portalfunktionen

3.4.1 Grundlagen

Nachdem zuvor die wichtigsten Portalarten vorgestellt wurden, sollen an dieser Stelle die wichtigsten Portalfunktionen erklärt werden. Bei den Portalfunktionen handelt es sich um Erweiterungen der üblichen Internet-Architektur (mindestens ein HTTP-, ein Applicationserver und im Regelfall eine Datenbank). Während der Applicationserver „nur“ das Ausführen von Programmen ermöglicht, erweitert eine Portallösung das bestehende System um folgende Funktionen:

- Authentisierung und Single Sign On
- Autorisierung und Benutzerverwaltung
- Personalisierung
- Applikationszugriff
- Aggregation
- Portalverwaltung

Teilweise erwähnen die verschiedenen Portalserverhersteller weitere Funktionen – klassischer Weise die Suche. Hierbei handelt es sich jedoch um zusätzliche Software, die nicht wesentlicher Bestandteil einer Portallösung ist. Sie muss i.d.R hinzugekauft werden.

Die unterschiedliche Portalsoftware unterscheidet sich dabei nach der Anzahl der angebundenen Systeme (relationale Datenbank, LDAP), der Güte der Benutzerverwaltung (Benutzeroberfläche, Flat-file) und nach den verschiedenen Applikationen, die für eine Portalsoftware verfügbar sind.

3.4.2 Authentifizierung und Single Sign On

Zugriffskontrolle ist ein Weg wie Daten und Programme vor Missbrauch geschützt werden können (vgl. [SH02], S490ff). Dabei wird zwischen Authentifizierung und Autorisierung (siehe unten) unterschieden. Unabhängig von der letztendlichen Art der Implementierung verwendet ein Authentifizierungsmechanismus meist eine Zugangskennung und ein Zugriffsschlüssel in Form eines Passwortes. Sie dienen der Identifikation eines Benutzers am System.

Als Beispiel dient der Prozess: Geldabheben an einem Geldautomaten. Die Zugangskennung wird über die Konto bzw. Kreditkarte an das System übermittelt. Den

Zugriffschlüssel übermittelt der Bankkunde durch das Eingeben seiner PIN. Entsprechen die eingegebenen Daten den hinterlegten, wird dem Kunden der Zugriff auf weitere Funktionen gewährt.

Betrachtet man verschiedenen Anwendungen und Systeme (Kontokarte, Handy, Zugangsdaten für den PC, verschiedene Host-Systeme, verschiedene Standardsoftware wie ERP, OLAP, Groupware) eines Menschen, wird die hohe Anzahl der zu merkenden Zugangsdaten ersichtlich. Unter Single Sign On versteht man einen Mechanismus, der es unter einmaliger Authentisierung ermöglicht, auf verschiedene Systeme zuzugreifen. Im Rahmen des Portals reicht also eine einmalige Anmeldung aus, um für die verschiedenen Anwendungen zugelassen zu werden.

3.4.3 Autorisierung und Benutzerverwaltung

Zweiter Bestandteil der Zugriffkontrolle (vgl. [SH02], S. 490ff) ist die Autorisierung[27]. Nach erfolgreicher Identifizierung werden einem Anwender verschiedene Rechte zugewiesen. Die erforderlichen Maßnahmen werden als Rechtverwaltung und Rechteprüfung (vgl. [SH02], S. 490) bezeichnet.

Rechte können explizit einem Anwender oder einer Anwendergruppe (einer Rolle) zugewiesen werden. Im Rahmen des Portals kann vom Portaladministrator beispielsweise festgelegt werden, welcher Anwender bzw. welche Anwendergruppe ein Portlet sehen (anwenden), editieren, konfigurieren oder Rechte delegieren darf (vgl. Abbildung 3-11 Zugriffsverwaltung).

Durch Authentifizierung und Autorisierung werden einem Portalanwender seine bereits personalisierten Anwendungen angezeigt.

3.4.4 Personalisierung

Findet ein Webbesucher die Möglichkeit vor, Inhalte einer Seite nach seinen Wünschen und Vorlieben anzupassen, kann man von Personalisierung sprechen.

Durch diese Möglichkeit grenzt sich eine Portalserversoftware deutlich von einem Content-Management-System (CMS) ab, das selbst „nur“ für die Aktualisierung von Webseiteninhalten zuständig ist und keine Personalisierung durch den Webbesucher vorsieht.

[27] engl. authorisation

Grundlegend werden zwei Arten von Personalisierung unterschieden:

- die rollenbasierte und
- die profilbasierte Personalisierung.

3.4.4.1 Rollenbasierte Personalisierung

Bei rollenbasierter Personalisierung wird dem Portalanwender eine Struktur bzw. die Anwendungen entsprechend seinen Vorlieben oder Merkmalen vorgegeben. Dazu können bei der Anmeldung an einem Portal Interessensgebiete des Anwenders abgefragt werden oder bereits vorhandene Informationen genutzt werden. In einem Unternehmen ist es so beispielsweise denkbar, dass abhängig von der Organistationseinheit besondere Anwendungen vorgegeben werden, z.B. erhalten Marketingmitarbieter automatisch Zugriff auf die aktuellen Daten aus der Marktanalyse und Markttrends. Meldet sich hingegen ein Kunde am Portal an, kann ihm durch Zugehörigkeit zu einer speziellen Kundengruppe eine besondere Offerte unterbreitet werden.

3.4.4.2 Profilbasierte Personalisierung

Durch die profilbasierte Personalisierung stellt der Portalanwender die gewünschten Informationen und Anwendungen selbst zusammen, d.h. er hat „seine“ Seite im Inter bzw. Intranet. Er kann nicht benötigte Anwendungen aus der rollenbasierten Personalisierung entfernen und nicht vorhandene hinzufügen. Durch diese flexible Art, seinen Arbeitsplatz zusammenzustellen, ist er im Vergleich zum starren System - mit der fest vorgegebenen Seitenstruktur – in der Lage produktiver zu arbeiten. Lange Suchzeiten über die Suchmaschine bzw. Navigationsleisten entfallen.

Ein Beispiel für eine Personalisierungsfunktion eines Portals ist „my yahoo“ (siehe unten). Nach der Anmeldung, bei der neben der Angabe der persönlichen Daten und Zuteilung einer persönlichen Identifikationsnummer noch ein Passwort anzugeben ist, erhält man hier die Möglichkeit, Themenbereiche und Layout für „sein yahoo“ auswählen.

Die folgende Abbildung zeigt, die personalisierte Seite „NeueSeite“ mit den Anwendungen Büro, Sport-Tageskalender, Bookmarks, Yahoo! Mail sowie Suche. Die darunterliegende Zeile deutet an, dass weitere Anwendungen der Seite hinzugefügt werden können.

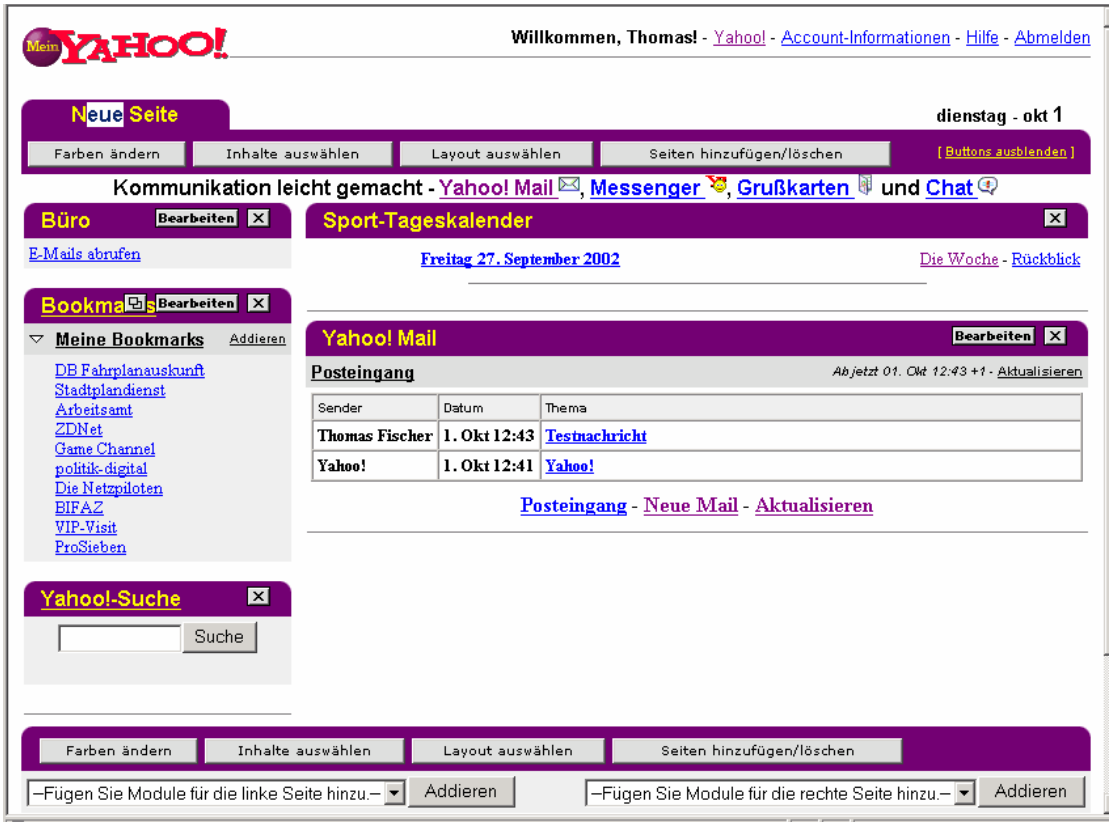


Abbildung 3-3 Personalisierung am Beispiel von "my yahoo"

3.4.5 Applikationszugriff

Primäres Ziel eines Portals ist es, die gewünschten Informationen und Anwendungen im Inter- oder Intranet bereitzustellen. Die Möglichkeit, Applikationen über das Internet auszuführen bietet bereits ein Applicationserver. Ein Portalserver erweitert dabei einen Applicationserver um Portalfunktionen: Authentisierung, Autorisierung, Personalisierung sowie eine einheitlichen Applikations- bzw. Portalverwaltung. Ferner wird in einem Portalserver die Möglichkeit geboten, verschiedene Anwendungen auf einer Seite zusammenzustellen, die Anwendungen bis auf eine Titelzeile auszublenden (minimieren) und auf der ganzen Seite anzuzeigen (maximieren).

Die hohe Anzahl der verschiedenen Portalarten verdeutlicht die vielfältigen Anwendungen innerhalb eines Portals wie z.B. E-Mail, Groupware, ERP-Anwendungen, OLAP etc.

Die folgenden Abbildungen zeigen den Applikationszugriff am Beispiel des IBM WebSphere Portal Server. Ohne dass eine entsprechende Softwareinstallation auf dem Client vorhanden ist, wird der Zugriff auf das Hostsystem sowie den Zugriff auf Lotus Notes – E-Mail, Kalender, Kontaktliste und ToDo-Liste möglich. Weitere Beispiele sind im Anhang enthalten (Kapitel 0, S.154).

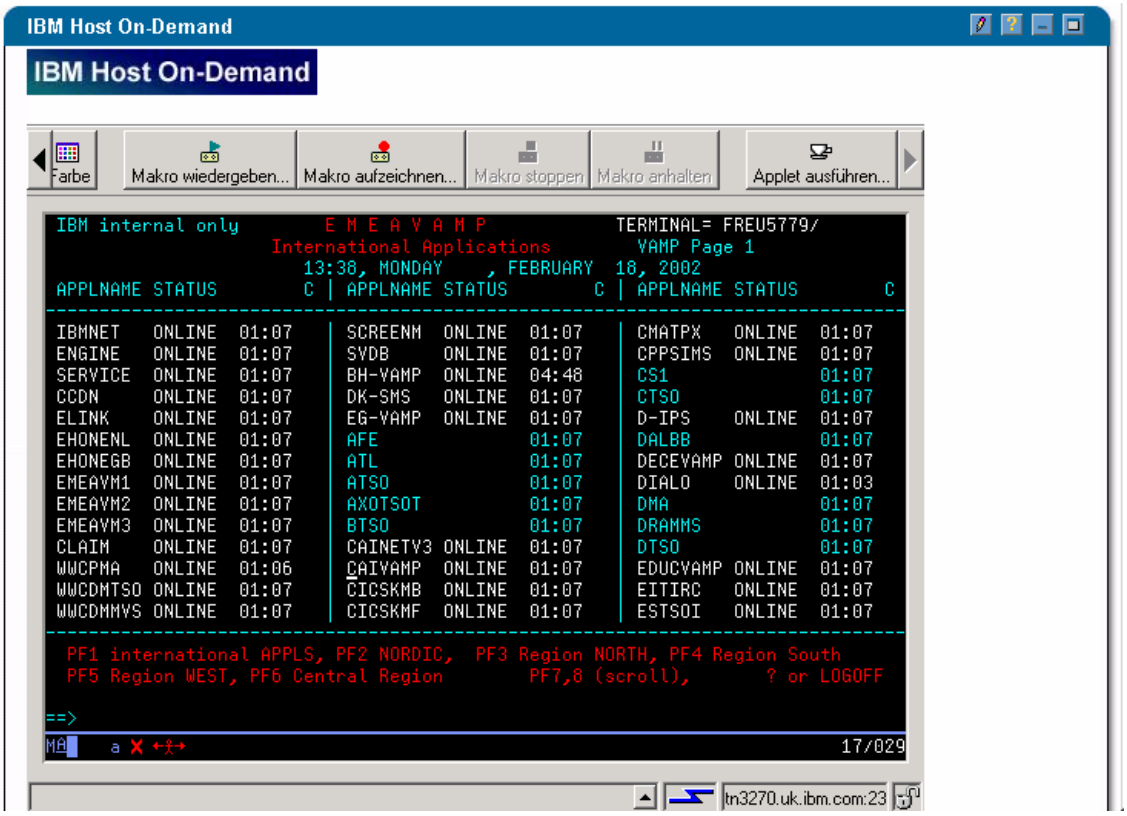


Abbildung 3-4 Applikationszugriff : Host

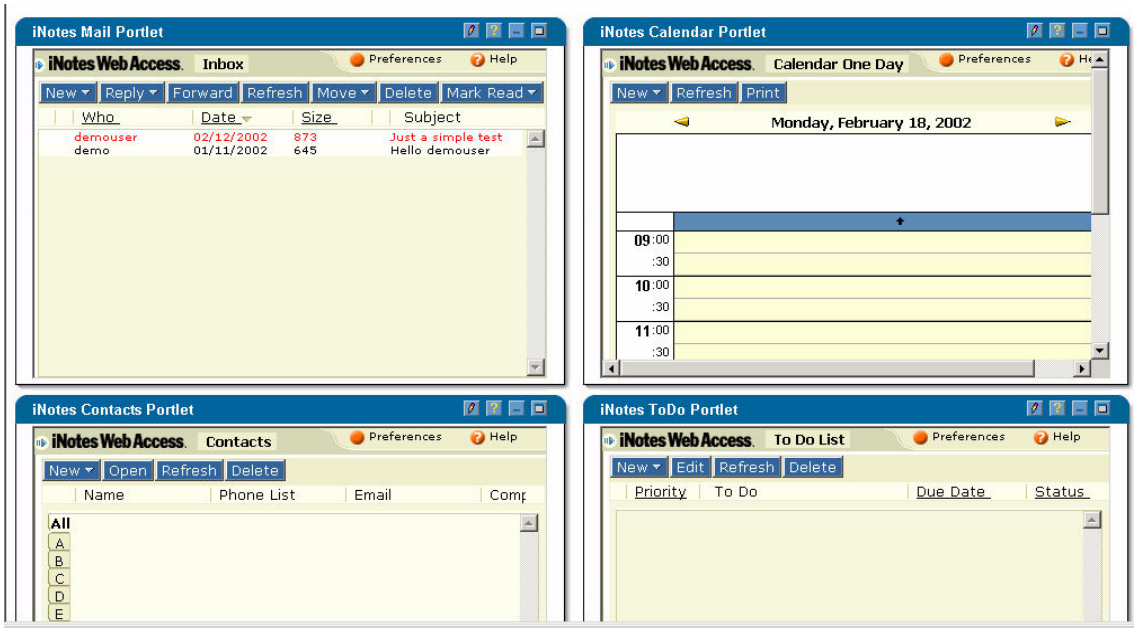


Abbildung 3-5 Applikationszugriff: Lotus Notes

3.4.6 Aggregation

In einem Portal kann ein Anwender sich mehrere Seiten zusammenstellen. Innerhalb einer Seite können wiederum verschiedene Anwendungen (siehe oben) zusammengefasst (aggregiert) werden. Die Aggregation unterscheidet ein Portal wesentlich von „üblichen“ Web-Schnittstellen[28] einzelner Programme – die jeweils nur eine Applikation auf einer Internetseite anzeigen.

3.4.7 Portalverwaltung

Die Portalverwaltung umfasst die zuvor genannten Portalfunktionen d.h. sie enthält die Möglichkeiten, wie ein Anwender oder Portaladministrator Einstellungen innerhalb seiner Seiten bzw. des gesamten Portals verändert.

Während der Anwender „sein“ Portal über die Personalisierungsmöglichkeit verwaltet[29], sorgt der Portaladministrator mit der Auswahl der Applikationen für den maximal zur Verfügung stehenden Funktionsumfang und mittels Benutzerverwaltung dafür, das nur die vorgesehenen Anwender bzw. Gruppen die Applikationen nutzen können (vgl. Abbildung 3-11).

Der Portaladministrator kann noch weiteren Einfluss auf das Portal ausüben. Er bestimmt die Struktur (den generellen Seitenaufbau) und das Layout des gesamten Portals[30].

Beispielsweise bestimmte er

- die Portalkopfzeile,
- die Navigationsleiste
- Aufbau der Anmeldemaske,
- Standardgrafiken der Titelleiste eines Portlets sowie
- die Seite zur Anmeldung eines neuen Benutzer
- die Seiten, die ein Anwender ohne Anmeldung am Portal sieht

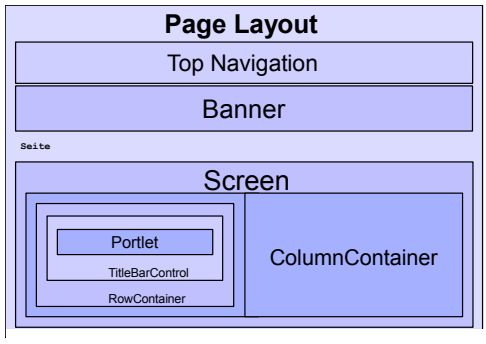


Abbildung 3-6 PageLayout des WPS

[28] gemeint ist hier, das die Programme wie Lotus Notes auch ohne Portal über das Internet bedienbar sind
[29] vgl. Abbildung 3-3 Personalisierung am Beispiel von "my yahoo" (Seitenfuß)
[30] sofern der Portalanwender kein anderes Layout wählt – wie beipsielsweise bei „my yahoo“ möglich.

3.5 Der Portalmarkt

Zdnet vermeldet(vgl. [I.31]), das nach Angaben von Gartner die Softwareausgaben der Unternehmen im Jahr 2001 gesunken sind. Gegen diesen Trend verzeichnen die Lizenzausgaben für Portalsoftware einen Zuwachs im Vergleich zum Vorjahr um 59 % auf 709 Mio. US-Dollar.
Trotz des erwarteten Einbruchs bei den Technologie-Ausgaben von 14 Prozent in diesem Jahr planen einer Umfrage von Forrester Research zufolge mehr als ein Drittel der 3.500 größten Unternehmen, noch 2002 Portal-Serversoftware zu erwerben. (vgl. [I.31])

Gartner beobachtet den Portalmarkt nicht erst seit diesem Jahr und hat zur Bewertung der verschiedenen Hersteller den „Portal Product Magic Quadrant“ entwickelt.
Die verschiedenen Portalserver-Hersteller werden anhand ihrer „Completeness of Vision“ sowie „Ability to Execute“ den Quadranten „Nice-Player“, „Visionär“, „Challenger“ oder „Leader“ zugeordnet.

Während Gartner am 23. Mai 2001 (vgl. [Gart01]) als Leader SAP, IBM, Sybase und Iplanet im Portalserver Markt vorsah, sind der der aktualisierten Fassung in „Big Change Evident in 2H02 Horizontal Portal Product MQ“ vom 1. März 2002 (vgl.[I.13]) folgende Mitglieder im „LeaderQuadranten“ vorhanden: IBM (jetzt bedeutender als SAP), SAP, Plumptee Software, Sun Microsystems und BEA Systems.

Gartner untersagt die Reproduktion des Magic Quadranten, deshalb sei auf die Internetadresse [I.13] verwiesen.

Im Magic Quadranten vom 1. März 2002 werden insgesamt 25 Hersteller benannt. Nicht erwähnt wird u.a. die „jetspeed“ Lösung des jakarta-Projektes der Apache Group [I.15]. Hierbei handelt es sich um die zukünftige Referenzimplementierung (RI) des in die Wege geleiteten Java-Standards zur Entwicklung von Portalanwendungen (Portlets).

3.6 WebSphere Portal Server (WPS)

Der Websphere Portal Server ist eine Software, die den Websphere Application Server um die vorher bereits erläuterten Portalfunktionen erweitert. Dabei verwendet der WPS verschiedene Projekte[31] der Apache Group. Nach der Vorstellung des Produktes erläutert dieses Kapitel, wie der WPS die Portalfunktion umsetzt.

3.6.1 Das Produkt

Um den verschiedenen Anforderungen von Unternehmen in unterschiedlichen Märkten gerecht zu werden, besteht die IBM WebSphere Portal Familie aus drei Angeboten:

- IBM WebSphere Portal Enable
  stellt die Basisfunktionen eines Portals dar, bestehend aus dem WebSphere Applikation Server, WebSphere Portal Server und WebSphere Personalisation und IBM Secure Way (LDAP)
- IBM WebSphere Portal Extend
  enthält zusätzlich zum Enable-Angebot noch ausführliche Möglichkeiten der Kollaboration, Suche und Analysefunktionen.
- IBM WebSphere Portal Experience
  ist das umfassendste Angebot der WebSphere Portal Familie und ist auf den Einsatz von öffentlichen Portalen oder Portalen in sehr großen Unternehmen ausgerichtet. Ergänzend zu den Funktionen des Extend-Angebots enthält es mit dem Tivoli Access Manager erweiterte Möglichkeit der Systemsicherheits-infrastruktur. Die Funktionen für Kollaboration sind durch die Komponenten Lotus Sametime, Lotus Quickplace erweitert. Mit dem IBM Content-Manager sind Möglichkeiten umfassender Web-Inhalte und Dokumentenverwaltung gegeben.

3.6.2 Portlets

Portlets bzw. eine Portletapplikation (eine Portletapplikation besteht aus einem oder mehreren Portlets) sind Anwendungen, die im Portalserver laufen. Der Portal-Administrator kann während des Betriebes Portlets verwalten.
Standardmäßig wird der IBM WebSphere Portalserver bereits mit verschiedenen Portlets ausgestattet. Zahlreiche freie Portlets für den WPS sind unter ([I.23]) zu finden

[31] aus dem jakarta-Projekt (jetspeed, turbine, log4j) sowie aus den XML-Projekten (Xalan, Xerces) (vgl. [I.15],[I.20])

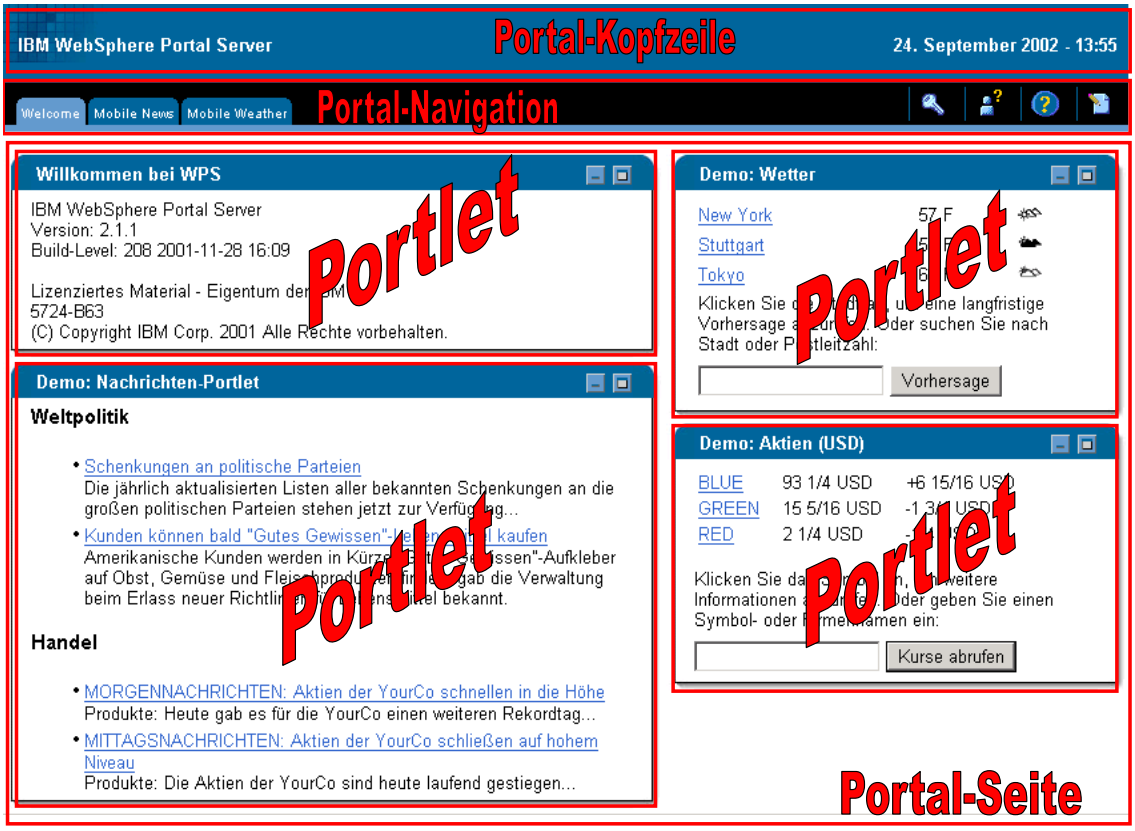


Abbildung 3-7 Portlets

Die obige Abbildung zeigt die Startseite des WebSphere Portal Server. Diese Seite wird standardmäßig allen Internetnutzern – ohne Anmeldung – gezeigt. Sie enthält neben dem Portalkopf und der Navigationsleite die „Willkommen“-Seite mit den vier verschiedenen Portlets. Der WPS ist nicht nur auf die HTML-Ausgabe beschränkt.

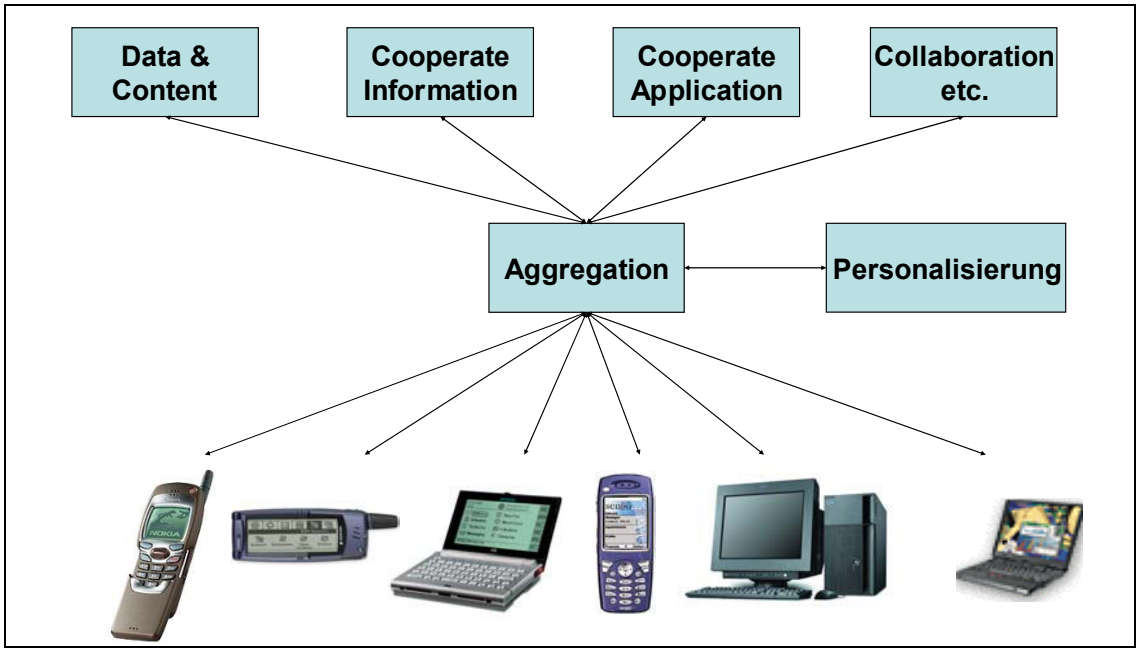


Abbildung 3-8 multi-device support

Durch die wachsenden Mobilitätsanforderungen gewinnen mobile Zugangsmöglichkeiten immer mehr an Bedeutung. Der WPS unterstützt bereits heute WML und cHTML (i-mode) für verschiedene Endgeräte wie Handy oder PDA.

3.6.2.1 Portlet-Modi

Im WebSphere Portal Server verfügt ein Portlet über drei Anzeigemodi, die über Symbole auf der Titelleiste des Portlets aufgerufen werden können.

- **Anzeigemodus (VIEW)**

  Wenn ein Portlet zum ersten Mal auf der Portalseite eines Benutzers aufgebaut wird, wird es im Anzeigemodus angezeigt. Dies ist der normale Modus.

- **Hilfemodus (HELP)** 

  Wird dieser Modus von einem Portlet unterstützt, wird vom Portlet eine Hilfetextseite für Benutzer bereitgestellt, die weitere Informationen über das Portlet enthält.

- **Bearbeitungsmodus (EDIT)** 

  Wird dieser Modus von einem Portlet unterstützt, wird vom Portlet eine Seite für die Benutzer bereitgestellt, auf der sie das Portlet entsprechend ihren Bedürfnissen anpassen können. Zum Beispiel kann ein Portlet eine Seite bereitstellen, auf der die Benutzer ihren Standort angeben können, um den lokalen Wetterbericht sowie Veranstaltungen und Ereignisse abzurufen. Zugriff auf den Bearbeitungsmodus haben nur Benutzer, die beim Portal angemeldet sind.

Durch Klicken auf eines der Steuerelemente kann der Modus oder Status des Portlets geändert werden. Die folgende Abbildung zeigt die leere Titelleiste eines Portlets im Anzeigemodus.

Abbildung 3-9 Standardtitelleiste eins Portlets

Die Titelleiste enthält die Symbole für Bearbeiten, Hilfe, Minimieren und Maximieren.

3.6.2.2 Portlet-Status

Über den Portlet-Status können die Benutzer die Darstellung des Portlet-Fensters im Portal ändern. In einem Browser rufen die Benutzer diese Status über Symbole in der Titelleiste genau in der Weise auf, in der Windows-Anwendungen geändert werden. Es werden folgende Stati unterschieden:

- **Normal**
  Wenn ein Portlet zum ersten Mal auf der Portalseite aufgebaut wird, wird es in seinem normalen Modus angezeigt und zusammen mit anderen Portlets auf der Seite angeordnet.
- **Maximiert** 
  Wenn ein Portlet in seiner maximalen Größe angezeigt wird, nimmt es den gesamten Hauptteil des Portals ein und ersetzt die anderen Portlets.
- **Minimiert** 
  Wenn ein Portlet minimiert bzw. in Symbolgröße angezeigt wird, wird nur die Titelleiste des Portlets auf der Portalseite angezeigt.

3.6.3 Benutzerverwaltung

Der IBM WebSphere Portal Server verwaltet die Einstellungn der Portalbenutzer innerhalb eines LDAP-Verzeichnisdienst (siehe unten) und in einer Datenbank[32]. Zusammen mit dem WPS wird SecureWay Directory ausgeliefert. Hierbei handelt es sich um ein LDAP-Verzeichnis von IBM. Neben SecureWay lassen sich aber auch problemlos andere LDAP-Server verwenden um beispielsweise existierende Verzeichnisdienste zu nutzen.

3.6.3.1 Was ist ein Verzeichnis(dienst) ?

Ein Verzeichnis ist eine Sammlung von (sortierten) Objekten bzw. Objektklassen und Informationen über diese Objekte – ihren Attributen.
Typisches Beispiel: Ein Telefonverzeichnis. Die Objekte sind hier die Personen. Sie besitzen bestimmte Eigenschaften (Telefonnummern, Faxnummern, Anschrift etc.). Innerhalb des Telefonverzeichnisses sind die Personen alphabetisch sortiert und in Orten, Kreisen, Ländern etc. zusammengefasst.

[32] Die wesentlichen Benutzerdaten werden im Verzeichnisdienst und erweiternde Einstellungen - wie die einzelnen Portletberechtigungen - werden in der Datenbank „WPS“ gespeichert.

Im Bereich der IT ist ein Verzeichnis eine spezielle Datenbank, die Informationen über Objekte speichert. Ein Verzeichnis über das Objekt Drucker speichert beispielsweise folgende Informationen: Anzahl der Seiten pro Minute (numerisch), Druckformate (beispielsweise Postscript oder ASCII), Lokation in der sich der Drucker befindet (alphanumerisch) etc.

Das Verzeichnis bzw. der Verzeichnisdienst ermöglichet es Anwendern und Anwendungen die gewünschten Informationen zu finden. Beispielsweise durch das Navigieren innerhalb des Verzeichnis bzw. Verzeichnisbaums, wie bei einem Telefonbuch oder aber durch das Suchen nach bestimmten Eigenschaften (Welcher Drucker in der aktuellen Lokation kann Postscript Dateien ausdrucken).
Aufgrund dieser Suchmöglichkeit wird ein Verzeichnis oft mit den Gelben Seiten verglichen.

Doch worin liegt der Unterschied zu „normalen“ Datenbanken und Verzeichnissen?
Verzeichnisse sind besonders auf Such- und Leseoperationen optimierte Datenbanken. Verzeichnisse liefen grundsätzlich keinen Transaktions- oder Rollback Mechanismus wie beispielsweise Datenbank Management Systeme (DBMS).
In einem Verzeichnis wird - im Vergleich zu normalen Datenbanken - verhältnismäßig wenig geschrieben und viel mehr gelesen.
Ein weiterer Unterschied ist das Fehlen der Abfragesprache Structured Query Language (SQL), die viele Datenbanken unterstützen, um komplexe Informationen aus normalisierten Tabellen zu erhalten bzw. einzufügen und zu aktualisieren. LDAP beispielsweise bietet hingegen einen einfachen Abfragemechanismus (vgl. [I.24]).

3.6.3.2 LDAP

Das Lightweight Directory Access Protocol (LDAP[33]) verwaltet verschiedene Einträge. Die Einträge haben verschiedene Attribute. Ein Eintrag wird über den Distinguished Name (DN) global eindeutig identifiziert. Jedes Attribut besitzt einen Typ und ein oder mehrere Werte. Werte eines Attributes können neben numerischen und alphanumerischen Werten auch binäre Daten enthalten. Dies ermöglicht beispielsweise,

[33] LDAP ist eine abgespeckte Variante des Directory Access Protocol (DAP), welches wiederum Bestandteil des X.500 Standards, in einem globalen Verzeichnis(dienst) ist. Details über LDAP können der RFC2251 "The Lightweight Directory Access Protocol (v3) (vgl.[I.22]) entnommen werden.

eine Grafik im jpg-Format in das Verzeichnis aufzunehmen, um ein Bild des Mitarbeiters in der entsprechenden Liste aufzunehmen.

Der Typ wird als mnemonischer String angegeben.

Häufig genutzte Attribute sind (unter anderen):

Bezeichnung	Mnemonischer String
Common Name	CN
Distinguished Name	DN
Organization Name	O
Organizational Unit Name	OU
Locality Name	L
Street Address	SA
State or Province Name	S
Country	C
Userid	UID

Anhand dieses mnemonischen Strings werden die Verzeichnisse zusammengestellt, die üblicherweise in einer Baumstruktur dargestellt werden.

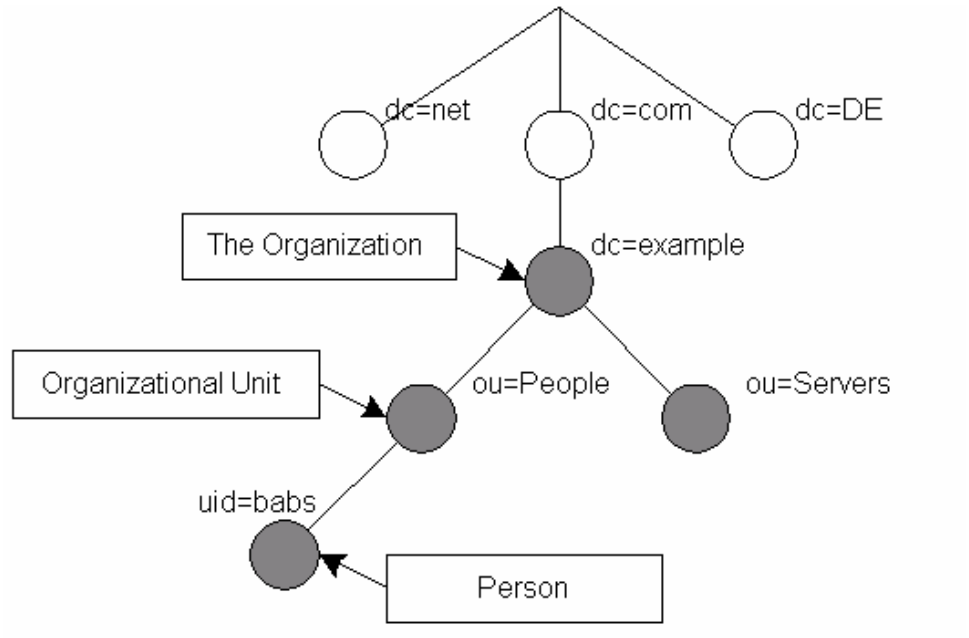


Abbildung 3-10 Beispiel LDAP-Baumstruktur[34]

Der Distinguished Name (DN) von „babs“ ist wie folgt:

```
dn=dc=com,dc=example, ou=People, uid=babs
```

Der Portalserver verwendet den Organizational Unit Name, sowie den Distinguished zur Rechteverwaltung. Der Portaladministrator kann Benutzern bzw. Benutzergruppen

[34] Quelle: [I.24]

Rechte auf Portlets und Seiten zuordnen. Die Zuordnung eines Benutzers in eine Gruppe wird im LDAP, die einzelnen Rechte in der Portalserver Datenbank abgelegt.

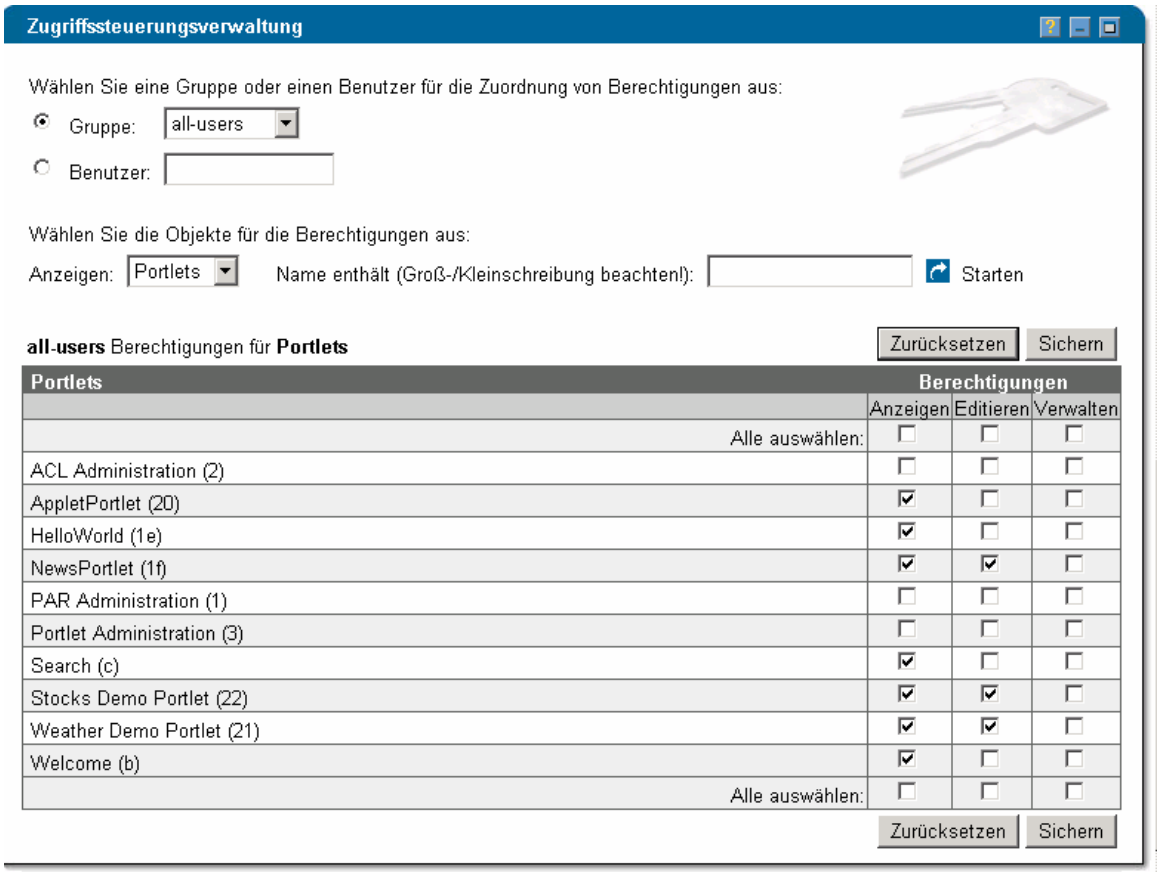


Abbildung 3-11 Zugriffsverwaltung

Die obige Abbildung zeigt die aktuellen Portletberechtigungen der Gruppe „all-users“. Hierbei handelt es sich um die eine spezielle Gruppe. Diese Einstellungen gelten für alle Portalbenutzer – also auch diejenigen, die nicht angemeldet sind (vgl Abbildung 3-7).

Der Portaladministrator kann Berechtigungen nicht nur für Portlets sondern auch den verschiedenen Seiten, Gruppen zuordnen. So können ganze Seiten – mit einer Auswahl an Portlets – vorgegeben werden.

Während in der obigen Abbildung alle Benutzer das „Welcome“ Portlet sehen („Anzeigen“-Spalte) dürfen, sind sie nicht für die Portlet-Administration zugelassen.

Das Editierrecht („Editieren“-Spalte) ermöglicht es einem Anwender ein Portlet einzustellen, beispielsweise seine Bookmarks zu verwalten. Das Verwaltungsrecht („Verwalten“-Spalte) ermöglicht es Rechte zu delegieren um beispielsweise Gruppenseiten zu personalisieren[35].

[35] Der aktuell angemeldete Benutzer verfügt also über das Verwaltungsrecht der Gruppe „all-users“ und

3.6.4 Portlet-Entwicklung

In diesem Kapitel folgt eine Beschreibung, wie Portlets selbst entwickelt werden können. Es sei auf weitere Möglichkeiten – auf Java Server Pages und Generetic Portlets - hingewiesen (vgl. [I.35]). Sie erlauben es, einfache Portlets auch ohne Kenntnis der Portlet-API zu erstellen.

3.6.4.1 Portlet- Standardisierung

Im Gegensatz zur Servlet-API ist die Portlet-API (vgl. [I.5]) noch nicht standardisiert. Das bedeutet, dass jeder Portalserverhersteller seine eigene „Portlet-API“[36] implementiert.

Mit dem Java Specification Requests (JSR) 168 liegt derzeit der dritte Versuch dem JCP – dem Java Community Process – einer Standardisierung vor.
IBM und SUN hatten zuvor zwei unabhängige JSR (162 und 167) eingereicht und zugunsten des 168 zurückgezogen. Bei der JCP handelt es sich um ein Gremium von derzeit mehr als 500 Firmen (SUN, IBM, Oracle etc) und Privatpersonen, die sich zusammengeschlossen haben, um Java-Standards zu beschließen.

Mit einer verabschiedeten „Standard Portlet-API“ lassen sich Portlets so entwickeln, dass sie unter den verschiedenen Portalservern austauschbar – und somit herstellerunabhängig - sind. Gerade diese Austauschbarkeit (wie beispielsweise von Servlets) ist ein wesentlicher Erfolgsfaktor der Programmiersprache Java.

Der IBM Portalserver verwendet die „jetspeed“ - Klassen. Jetspeed ist ein jakarta Projekt der Apache Group (vgl. [I.15]) und wird zugleich die wichtige Referenzimplementierung (RI) des in die Wege geleiteten Portletstandards. Eine standardnahe Entwicklung ist also sichergestellt, zumal der Vorsitz des JSR 168 ebenfalls in der Hand von IBM ist.

[36] oder wie auch immer die Hersteller ihre Portal-Programmierschnittstelle nennen

3.6.4.2 PAR-File

Als erweiterbare Anwendungen „Pluggable Modules“ werden Portlets in einer Datei zusammengefasst – in einem so genannten Portlet Archive File (PAR-File). Ein PAR-File wird - wie ein übliches Java Archive File(jar-File) – in komprimierter Form ausgeliefert.

Im Einzelnen setzt sich ein PAR-File wie folgt zusammen:

/

Wurzelverzeichnis

/images

in diesem Verzeichnis sind die Grafiken eines Portlets zu hinterlegen

/PORTLET-INF/

Das PORLTET-INF Verzeichnis bewerkstelligt, dass die darin enthaltenen Dateien eines Portlets nicht direkt über die URL abrufbar sind, sondern nur vom Portlet verwendet werden können.

/PORTLET-INF/classes/

Unterhalb dieses Verzeichnisses sind einzelne Java-Klassen zu hinterlegen

/PORTLET-INF/lib/

Unterhalb des PORTLET-INF/lib/ Verzeichnis sind Dateien im Java Archive Format zu hinterlegen.

Zentrale Einstellungen eines Portlets sind der Datei /PORTLET-INF/portlet.xml vorzunehmen. Diese Datei (der so genannte Portletdeskriptor) enthält neben den Konfigurationsparametern einer Portletapplikation generelle Informationen wie den Markuptype (HTML, WML, cHTML) und die unterstützten Portlet-Modi (Edit, Help).
Die Überführung in ein PAR-File kann über das zum Lieferumfang von j2sdk gehörende Java Programm jar erfolgen.

3.6.4.3 Portlet-API

Die Klassen der Portlet-API werden nicht von der Servlet-API abgeleitet, d.h. es handelt sich grundsätzlich um unabhängige Klassen. Die Klassen der Portlet-API orientieren sich jedoch an den Servlet-Klassen. Die Abweichungen sind an der Verwendung zusätzlicher Klassen (siehe unten) sowie abweichender Methoden innerhalb der Servlet-API „ähnelnden“ Klassen[37] ersichtlich. Die folgende Abbildung zeigt einen Überblick der wesentlichen Klassen[38] (innerhalb der Rechtecke) und Zugriffsmethoden der Portlet-API.

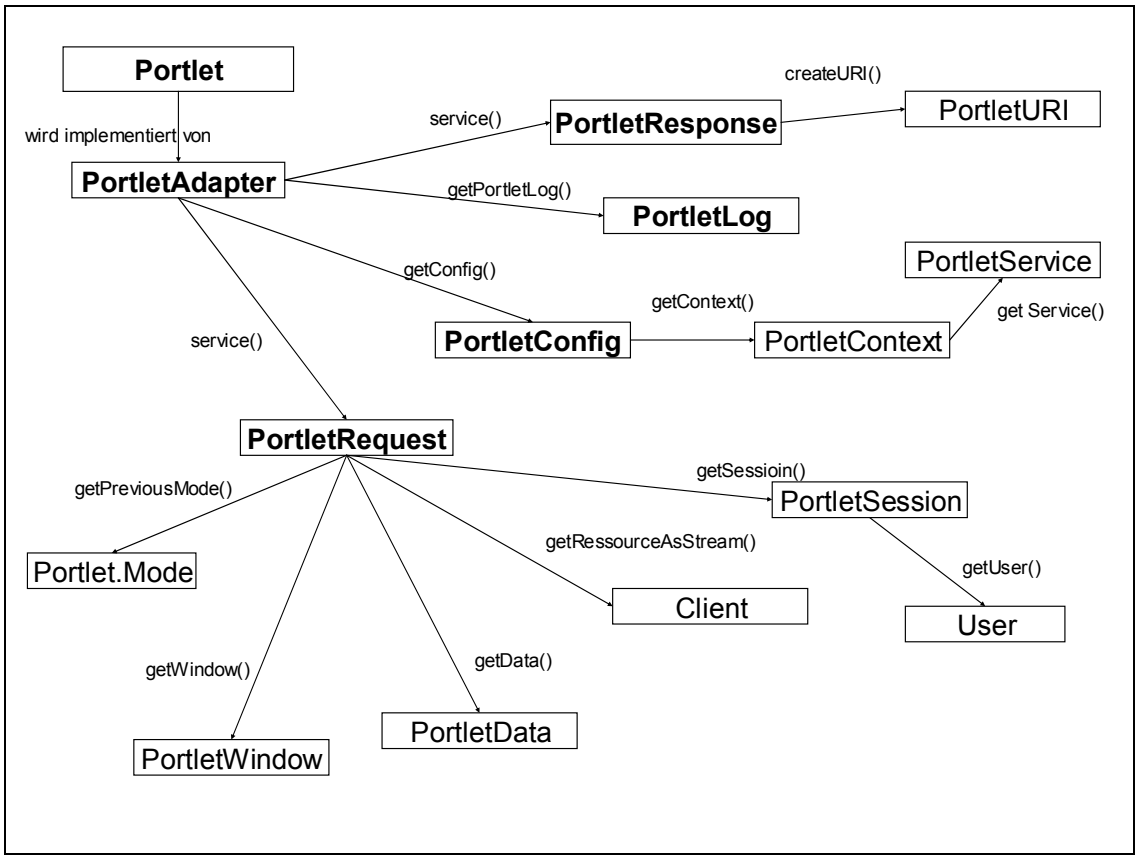


Abbildung 3-12 Überblick Portlet-API

Neben den bereits erwähnten Möglichkeiten Portlet-Modi[39] und Portlet-Status[40] zu verwenden bietet die Portlet-API wesentlich folgende Erweiterungen:

[37] beispielsweise bietet die PortletSession im Vergleich zur HttpSession keine isNew() Methode
[38] bzw. die zur Laufzeit zu verwendenden Objekte
[39] Die Klasse PortletWindow besitzt die Methode getWindowState(), die Werte für minimized, maximized, normal etc. liefert
[40] Die Klasse PortletMode besizt die Methode getId() sie liefert Werte entsprechend des Edit, View, Configure und Help Modus zurück

Logging

Das jakarta Projekt „log4J“ (vgl.[I.15]) wird zusammen mit dem Portalserver ausgeliefert und bietet die Möglichkeiten Laufzeitinformationen eines Portlets aufzuzeichnen[41]. Grundsätzlich bietet log4J (vgl.[I.20]) die Möglichkeiten Fehler, Debuginformationen und Warnmeldungen zu loggen. Dementsprechend können die Methoden `getPortletLog.error()`, `getPortletLog.debug()`, `getPortletLog.info()` und / oder `getPortletLog.warn()` mit den gewünschten Meldungen aufgerufen werden. Im Portalserver können Einstellungen vorgenommen werden ob und in welche Datei Loggingmeldungen geschrieben werden.

Aus Gründen der Performancesteigerung bietet log4j jeweils die Möglichkeit abzufragen, ob der entsprechende Logging-Modus aktiv ist, z.B.

`getPortletLog.isDebugEnabled()`.

Ereignisverarbeitung

Portlets können Ereignisse verarbeiten. Denkbar ist eine Portletapplikation mit 3 Portlets (Kopf, Links und Hauptteil). Durch Betätigen eines Buttons des Linken Portlets wird ein Ereignis „geworfen“. Die beiden anderen Portlets können dieses Ereignis „auffangen“ und anhand der Daten aus dem linken Portlet die Verarbeitung des Ereignisses durchführen.

Im Portalserver 2.1 können 3 verschiedene Ereignisse (events) realisiert werden:

- Ein window-event wird aufgerufen, wenn der Anwender den Status eines Portlets ändert z.B. Maximierung eines Portlets (Beispiel siehe)
- Ein action-event wird aufgerufen, wenn der Anwender einen Button betätigt. Dieses Action event kann dann entsprechend verarbeitet werden.
- Ein message-event ermöglicht den Datentransport zwischen Portlets. Die Portlets müssen dazu der gleichen Portletapplikation angehören und sich auf der gleichen Seite innerhalb des Portals befinden.

Die verschiedenen Events werden durch die entsprechenden Listener (window-, action- und message-Listener) aufgefangen. Die verschiedenen Listener müssen samt implementierender Klasse in der Portletkonfigurationsdatei (portlet.xml) eingetragen werden.

[41] loggen

Persistenz

Die Portlet-API bietet eine einfache Möglichkeit Daten zu speichern. Dazu wird das Objekt „PortletData“ (vgl. Abbildung 3-12) verwendet. Ein Objekt dieser Klasse wird seitenabhängig – und nicht userabhängig - gespeichert. Wenn also ein Portlet auf einer Gruppenseite (vorgegeben durch den Administrator) diesen Mechanismus benutzt, hat es zur Folge, das alle User – Zugriff auf diese Daten besitzen. In einem PortletData-Objekt können Daten in Form von Schlüssel-Wert Paaren Daten gespeichert werden.

3.6.4.4 Portlet-Schnittstelle

Die Portlet-Schnittstelle ist die zentrale Abstraktion der Portlet-API. Alle Portlets implementieren diese Schnittstelle, in den meisten Fällen durch Erweitern einer Klasse, die die Schnittstelle implementiert, wie z. B. PortletAdapter (vgl. [I.5]).

Die folgenden wesentlichen Methoden der Portlet-Schnittstelle werden für ein Portlet im Laufe seines Lebenszyklus aufgerufen:

init()

Das Portlet wird nach der Initialisierung des Portals aufgebaut und anschließend mit der Methode init() initialisiert. Das Portal erstellt immer nur eine einzige Portlet-Instanz, die dann von allen Benutzern gemeinsam benutzt wird, vergleichbar mit einem Servlet, das von allen Benutzern eines Anwendungsservers gemeinsam benutzt wird.

login()

Nachdem sich ein Benutzer an einem Portal angemeldet hat, erstellt jedes Portlet eine Session[42] für den Benutzer. Die Kombination aus physischer Portlet-Instanz und Benutzersession erstellt die virtuelle Portlet-Instanz. Mit dieser Methode kann das Portlet die Sessioninstanz des Benutzers initialisieren, um beispielsweise Attribute in der Session zu speichern.

service()

Das Portal ruft die Methode service() auf, wenn das Portlet aufgefordert wird, seinen Inhalt wiederzugeben. Während des Portlet-Zyklus wird die Methode service() normalerweise viele Male aufgerufen.

[42] Sie ermöglicht den Transfair von Daten innerhalb des Portalserver.

logout()

Wenn der Benutzer die Sitzung mit dem Portal beendet, ruft der Portalserver die Methode logout() auf, um das Portlet darüber zu informieren, dass die Sessioninstanz des Benutzers beendet wird. Das Portlet muss daraufhin alle für die virtuelle Instanz verwendeten Ressourcen bereinigen.

Während die Methoden init(), service() auch in der Servlet-API vorhanden sind, kommen in der Portlet-API die Methoden login() und logout() hinzu.

3.6.4.5 Portlet-Beispiel

Das folgende Beispiel (vgl. [I.23]) zeigt den Java-Code des üblichen HelloWorld Programms unter Verwendung der Klassen des jetspeed-Projektes:

```
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlets.*;
import java.io.*;
package com.ibm.wps.samples.helloworld;

public class HelloWorld extends AbstractPortlet{

      public void init(PortletConfig portletConfig) throws
UnavailableException{
            super.init( portletConfig );
      }
      public void service( PortletRequest portletRequest,  PortletResponse
                                    portletResponse )
                  throws PortletException, IOException {

             PrintWriter writer = portletResponse.getWriter();

             writer.println("<p>Hello Portal World!</p>");
      }
}
```

Abbildung 3-13 Java-Code eines Portlets

Der Java-Code des Portlets ist in diesem Fall (da weder Grafiken noch Eingabefelder ausgelesen werden) einem Servlet sehr ähnlich.

```
<?xml version="1.0"?>
<!DOCTYPE portlet-app PUBLIC "-//IBM//DTD Portlet Application 1.0//EN"
     "portlet.dtd">
<portlet-app>
    <portlet-app-name>HelloWorld Portlet App #1</portlet-app-name>
    <context-param>
        <param-name>Portlet Master</param-name>
        <param-value>yourid@yourdomnain.com</param-value>
    </context-param>
    <portlet>
        <portlet-name>HelloWorld</portlet-name>
        <portlet-class>com.ibm.wps.samples.helloworld.HelloWorld
             </portlet-class>
        <portlet-type>instance</portlet-type>
        <allows>
            <maximized/>
            <minimized/>
        </allows>
        <language locale="en_US">
          <title>HelloWorld - Example Portlet #1</title>
          <title-short>Hello-World</title-short>
```

```
          <description>HelloWorld - Example Portlet #1</description>
          <keywords>portlet hello world</keywords>
        </language>
        <supports>
            <markup name="html">
                <view/>
            </markup>
        </supports>
    </portlet>
</portlet-app>
```

Abbildung 3-14 Beispiel Portlet.xml

Der zuvor gezeigte Portletdeskriptor (Datei portlet.xml) zeigt die Konfiguration einer Portletapplikation.

Der Inhalt des <portlet-app-name> Element wird während der Installation (siehe Kapitel 8.14, Anhang S. 155) im Administationsportlet angezeigt. Konfigurationsparameter können vom Administrator in innerhalb des <context-param>-Elementes mitgeteilt werden.

Für jedes Portlet einer Portletapplikation ist ein <portlet> Element anzulegen. Der Portletname bezeichnet dabei den Java-Klassennamen des Portlets samt Paketnamen. Ferner kann für jedes Portlet eine Beschreibung angegeben werden. – Diese Beschreibung wir dem Portalanwender während der Personalisierung seiner Seiten angezeigt (siehe Kapitel 8.13, S154).

Neben weiteren Ausgabeformaten[43] werden in der Datei portlet.xml auch noch die verschiedenen Listener der zuvor vorgestellten Ereignisverarbeitung eingetragen (vgl. [I.10]).

PAR-File Verzeichnisstruktur, Inhalt und Erzeugung.

```
d:/HelloWorld
d:/HelloWorld/PORTLET-INF/portlet.xml
d:/HelloWorld/PORTLET-
INF/classes/com/ibm/wps/samples/helloworld/HelloWorld.java
```

Unter der Annahme, dass die benötigten Klassen (des jetspeed Projektes) im Classpath[44] vorhanden sind, kann ein PAR-File mit folgendem bat-file[45] erzeugt werden:

```
rem *** Inhalt von makeHW.bat

set bd=D:/HelloWorld
javac -d %bd%/PORTLET-INF/classes
   %bd%/PORTLET-INF/classes/com/ibm/wps/samples/helloworld/HelloWorld.java

jar cvf0 HelloWorld.par PORTLET-INF
```

[43] weiter vorgesehen sind WML sowie cHTML

[44] Classpath ist eine Variable, die auf Verzeichnisse oder Dateien verweist. Der Classpath kann beipsielsweise in der DOS-Eingabeaufforderung mit Set classpath=C:\test.jar festgelegt werden.

[45] Ein bat-file enthält verschiedene Anweisungen (z.B. Löschen von Dateien). Diese Anweisungen können durch Aufruf des Dateinamens unter DOS ausgeführt werden.

4 Enterprise Application Integration

4.1 Definition

Ovum definiert (unter [I.25]) den Begriff „Enterprise Application Integration“ (EAI) als:

„…the combination of technologies and processes that enable custom-built and/or packaged business applications to exchange business-level information in formats and contexts that each understands.”

Das heißt, dass EAI unterschiedliche Anwendungen in die Lage versetzt geschäftsrelevante Daten so miteinander auszutauschen, dass alle am Austausch beteiligten Anwendungen ein gleiches Verständnis der Daten haben.

Die Gartner Group sagt zum Thema EAI schlicht und einfach:

„Application integration means making independently designed systems work together.“

Fasst man die obigen Definitionen zusammen geht es bei EAI darum, System übergreifende Geschäftsprozesse mit Hilfe einer geeigneten Technik so abzubilden, dass die am Geschäftsprozess beteiligten unterschiedlichen Systeme in der Lage sind, prozessrelevante Daten in geeigneter Form auszutauschen.

4.2 EAI-Entwicklungsgeschichte[46]

EAI ist nicht erst seit diesen Tagen von Bedeutung. Die ersten Rechner erhielten in den 50er Jahren Einzug in die Unternehmen. Die Daten, die zuvor in Form von Akten über meterlange Regale gehalten wurden, fanden Platz auf einem Großrechner. Aufgrund der hohen Kosten für die IT – oder eher für die alte Datenverarbeitung – existierte oft nur ein Rechner mit vielen Bedienstationen – Terminals genannt.

[46] (vgl. [I.9], [I.42])

Auf diesem einen Rechner konnten die Daten vergleichsweise einfach ausgetauscht (ein Betriebssystem, ein Datenformat etc.) und die verschiedenen Anwendungen integriert werden.
Die Technologie entwickelte sich weiter. Unix wurde ab 1969 entwickelt parallel dazu entstand das ARPA-Net - die Mutter des heutigen Internets. Ab 1982 hielten die ersten PC`s mit anderen Betriebssystemen Einzug in die Unternehmen.
Das hatte zur Folge, dass die Daten von einem Zentralsystem auf kleinere Unix und NT Systeme verteilt wurden. Man hatte es nun nicht mehr mit nur einem zentralen Mainframe-System zu tun, das als Rückrad der gesamten EDV diente, sondern man befand sich in einer heterogenen Umgebung mit dezentraler, redundanter Datenhaltung auf unterschiedlichen Betriebssystemen und Geschäftsprozessen, die sich über mehrere Systeme erstreckte. Hieraus entstand die Notwendigkeit, Daten mit anderen Systemen in elektronischer Form austauschen zu können.

Ein Versuch den Wildwuchs wieder in den Griff zu bekommen, war die Einführung von Enterprise Resource Planing (ERP) Systemen, wie z.B. SAP, Baan oder Peoplesoft.

Die verschieden Hersteller versuchen alle geschäftkritischen Prozesse in Ihren Systemen abzubilden. Nicht alle Softwareanforderungen von Unternehmen können jedoch von ERP-Systemen erfüllt werden (vgl. [KN99]).

Neben der rasanten Hardwareentwicklung entstanden auf Softwareseite - neben den ERP-Sytemen - neue Programmiersprachen und sogar neue Programmierparadigmen wie z.B. komponentenbasierte oder objektorientierte Softwareentwicklung, um mit der Entwicklung auf der Hardwareseite Schritt zu halten.

Besonders der Wechsel von Programmierparadigmen stellt Unternehmen vor eine Herausforderung.
Die eigene IT-Abteilung beherrscht meist „nur“ die alt bekannten (z.B. prozedural). Der Umstieg auf neue (z.B. Objektorientierung) ist mit erheblichen Aufwendungen für Schulungen – Motivation der Mitarbeiter vorausgesetzt - verbunden.
Stellen die Unternehmen jedoch nicht auf neue Programmierparadigmen um, sehen sie sich später der Tatsache ausgesetzt, keine externe Hilfe mehr für ihre Altsysteme zu bekommen.

Dem Thema alte und neue Software zu vereinen widmet sich Enterprise Applikation Integration

4.3 Formen von EAI[47]

Der Begriff EAI wird von den verschiedenen Herstellern sehr großzügig verwendet. Schaut man jedoch auf die Anforderungen, so bilden sich drei Formen von EAI heraus.

- Application-to-Application (A2A)
- Business-to-Business (B2B)
- Business-to-Consumer (B2C)

Bei der A2A Integration geht es darum, die verschiedenen Anwendungen innerhalb eines Unternehmens miteinander kommunizieren zu lassen. Besonders wichtig hierbei ist – historisch bedingt - die Integration der Legacy-Systeme.
Im Bereich der A2A-Integraion kommen besonders Message Broker zum Einsatz. Alternative Begriffe für A2A in der Fachpresse sind: interne Integration und inter-EAI.

Mit dem Erfolg des Internets spielt die Integration von Anwendungen verschiedener Unternehmen - ohne den Austausch von Datenträgern - eine immer wichtigere Rolle. Ein Beispiel für B2B ist e-procurement bei dem z.B. eine SAP-Anwendung bei einer anderen ERP-Anwendung eine Bestellung placiert. Dieser Bereich wird B2B genannt. Als Austauschformat wird XML eine besondere Bedeutung zugetragen.

Im Bereich B2C-Integration soll der Endkunde mit den Prozessen des Unternehmens verbunden werden. Idealerweise muss eine manuelle Bearbeitung des Kundenwunsches durch einen Sachbearbeiter nicht mehr vorgenommen werden. Im diesen Bereich hat sich der HTTP-Server in Verbindung mit einem Applikation Servers als Mittel zur Kommunikation herausgestellt.

[47] (Quelle [I.42])

4.4 Integrationsmöglichkeiten

Nachdem die Grundlagen und die Formen von EAI vorgestellt wurden, werden in diesem Kapitel die Möglichkeiten der Integration vorgestellt. Das Ausgangsproblem ist jeweils identisch:

Wie kommt man an Daten, Funktionen und Objekte anderer Anwendungen heran?

4.4.1 Datenzugriff

Die vermeidlich einfachste Art an Daten heranzukommen bieten Schnittstellen wie JDBC[48] oder ODBC[49]. Sie bieten die Möglichkeit nach einem einheitlichen Modell verschiedene „dahinterliegende" Datenbanken bzw. Datenquellen anzusprechen.
Verschiedene Anwendungen können solche Schnittstellen nutzen, um sich den Datenbestand „zu teilen".
In der Batchverarbeitung ist es üblich mit Kopien von Dateien zu arbeiten. Die Kopien dieser Dateien können je nach Verwendung[50] angepasst werden. Werden diese Daten jedoch innerhalb eines Programmdurchlaufs verändert, muss ein Mechanismus bereitgestellt werden, der die Änderung der Daten auch in der Ursprungsdatei vornimmt.

4.4.2 Remote Procedure Call

Mit Remote Procedure Calls (RPC) können Funktionen aus anderen Programmsystemen aufgerufen werden. Der Aufruf erfolgt hierbei über ein Netzwerk. Grundlage für diese Technologie bilden Client-Server Systeme (vgl. [MP02], S.12)

4.4.3 Objektorientierte Konzepte

Konzepte wie COM, CORBA, DCOM, RMI oder Enterprise Java Beans (siehe. [GT00]) werden oft als objektorientierte Erweiterung von RPC angesehen. Diese Konzepte erlauben die direkte Verwendung von kompletten Objekten - also nicht „nur" einzelner Funktionen aus anderen Anwendungen. Beispielsweise können mittels COM Inhalte aktuell geöffneter Programme - innerhalb eines anderen Programms - verwendet werden.

[48] Java Database Connectivity
[49] Open Database Connectivity
[50] z.B. Sortierung für den Gruppenwechsel

4.4.4 Messageorientierte Middleware

Messageorientierte Middleware (MOM) basiert auf der Idee, dass Anwendungen miteinander kommunizieren, indem sie Nachrichten[51] austauschen. MOM konzentriert sich darauf, Daten von einem Programm zum anderen zu transportieren. Die Integration via Nachrichten basiert i.d.R. auf asynchroner Kommunikation in einer verteilten Systemumgebung[52].

Message-Queuing setzt das Konzept[53] von MOM (vgl.[I.30]) am deutlichsten um. Deshalb werden die Begriffe MOM und Message-Queuing oft synonym verwendet. Hierbei kommunizieren Anwendungen, indem sie Nachrichten in Warteschlangen (Queues) innerhalb der Systemumgebung ablegen.

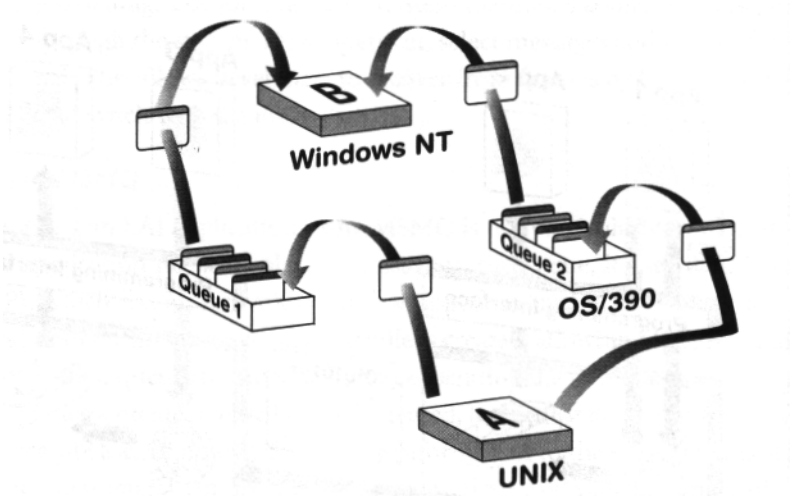


Abbildung 4-1 Message Queue[54]

Ein bekanntes Beispiel für diese Art der Kommunikation stellt MQSeries dar. Ein Quellsystem schreibt einen Datensatz in eine Output-Warteschlange (vgl. Abbildung 4-1). MQSeries liest diese „Queue“ aus und stellt den Datensatz in die „Input-Queue“ des Zielsystems.

---

[51] engl. messages
[52] vgl. Abbildung 4-1- hier zwischen Windows NT und einem Unix System.
53 neben Messag Quing werden in [I.30] noch Publish & Subscribe sowie Synchronous Messaging/Message Passing vorgestellt
[54] Quelle: [Lint00]

David S. Linthicum in [Lint00], S.170:

„In the world of MOM, IBM's MQSeries is the proverbial 600-pound gorilla. Dominating queuing middleware with a market share of over 65 percent, it can do just about anything is pleases. ...MQSeries sits aoip the EAI world as the preferred messaging layer for moving information throughout an enterprise"

4.4.5 Application Programming Interfaces

Programmiersprachen und Anwendungen stellen Zugriffsmechanismen in Form von Schnittstellen so genannter „Application Programming Interfaces" (API) bereit. Diese Schnittstellen erlauben das Aufrufen vorgefertigter Programme bzw. Programmteile, z.B. die SAP-BAPI. API's werden in aller Regel von allen vorher genannten Implementierungs-methoden verwendet.

5 Softwareentwicklung mit „Evolution“

Evolution ist ein Softwareentwicklungsmodell für webbasierte, unternehmensweite Anwendungsintegration.
Die Komplexität der zu erstellenden Software versucht Evolution durch Planung der einzelnen Aktivitäten zu kompensieren. Evolution gibt dabei feste Rollen und Artefakte[55] vor. Dabei ist Evolution nicht so schwergewichtig wie beispielsweise der Rational Unified Prozess (RUP)[56].
Ferner ist Evolution aber auch ein Framework das projektübergreifenden Java-Code vorgibt. Zur Entwicklung der Anwendung wird dabei ein geschäftsprozessgetriebenen Ansatz verfolgt. Evolution definiert einen Prozess als Folge von Vorgängen, die wiederum Aktivitäten enthalten. Diese Struktur wird bereits wärend der Analayse ermittelt und in UML-Aktivitätsdiagrammen abgebildet.
Diese Aktivitätsdiagramme werden erweitert und in Java-Code überführt, bevor dieser um die leztendliche Implementierung – und somit der Anbindung der verschiedenen Backendsysteme – erweitert wird. Dazu bietet das Framework verschiedene Basisklassen vor. Evolution unterstützt dabei die Integration auf Datenebene und den Zugriff über MQSeries.

5.1 Problemstellung der Softwareentwicklung

Durch die Zuname immer komplexer werdender Systeme in unserem Alltag, gewinnt die Entwicklung von Software immer mehr an Bedeutung (vgl.[Comp00]) . Eines der größten Probleme der heutigen Softwareentwicklung ist es, qualitativ hochwertige Software zu erstellen. Dabei lassen sich folgende Punkte ausmachen, die für das Scheitern von Softwareprojekten verantwortlich machen:

- Mangelndes Verständnis für die Bedürfnisse und Denkweise des Kunden bzw. falsch übermittelte Anforderungen,
- Inflexible Systeme, die bei Änderungen der Randbedingungen in sich zusammenbrechen,
- Projektfehler werden zu spät erkannt und können nur mit erheblichen Kosten behoben werden,

[55] zu erstellende Dokumente
[56] Der RUP sieht in jedem Projekt 31 Artefakte vor, (siehe www.ration.com)

- Schlechte Code Qualität, erschwert das Verständnis, das auf Entwicklerseiten notwendig ist um Software zu warten und
- Mangelnde Performance

5.2 Ziele von Evolution

Die zuvor beschriebenen Problemstellungen der Softwareentwicklung haben im Wesentlichen zur Entwicklung von Evolution als Framework beigetragen. Im Einzelnen werden folgende Ziele bei der Verwendung von Evolution verfolgt:

- Verwendung einer mehrschichtigen Architektur
- Mandantenfähigkeit
- Integration der bestehenden Systeme und Anwendungen
- 1:1 Abbildung der Geschäftsprozesse
- klar strukturierter und lesbarer (generierter) Code
- kurze Entwicklungszeiten durch Verwendung dieses Frameworks
- Einsatz getesteter Komponenten
- Verwenden einer erprobten Architektur
- und dadurch Kosten und Risikominimierung

Durch die Unterstützung des Softwareentwicklungsprozess als Ganzes (als Vorgehensmodell) sowie die Vorgabe des Frameworks – mit der Möglichkeit verschiedene Systeme anzubinden - sollen Webprojekte erfolgreich durchgeführt werden.

5.3 Grundbegriffe

5.3.1.1 Workflowelemente

Evolution verwendet verschiedene Workflowelemente. Diese Workflowelemente bestimmen den Ablauf einer Evolution-Anwendung.

Workflowelemente	Bedeutung
Prozess	Beschreibt die zu entwickelnde Anwendung. Dieses Element leitet den Workflow ein. Ein Prozess besteht aus einem oder mehreren Vorgängen.
Vorgang	Ein Vorgang ist eine fachlich abgeschlossene Einheit z.B. Erstellen eines Angebotes. In Evolution werden die identifizierten use cases auf Vorgänge abgebildet. Ein Vorgang enthält mindestens eine Aktivität.
Aktivitaet	Eine Aktivität ist eine abgeschlossene, atomare (eher technische) Einheit (beispielsweise ein Backend-Zugriff). Eis existiert keine Vorgabe zur Ermittlung von Aktivitäten.
AnzeigeAktivitaet	Eine AnzeigeAktivitaet ist eine besondere Aktivität. Sie wird modelliert, um den Workflow zu beenden und die ermittelten Daten an den Browser zu leiten.

Abbildung 5-1 Workflowelemente in Evolution

Um Verwechselungen zu vermeiden, sei bereits an dieser Stelle darauf hingewiesen, dass im weiteren Verlauf eine UML-Aktivität als „Aktivität“ und eine spezielle Evolutionaktiviät als „Aktiviaet“ – also mit „ae“ bezeichnet wird.
Die Workflowelemente werden in der Analyse als Stereotypen innerhalb UML-Diagramme verwendet. Innerhalb des Frameworks sind Basisklassen für die Workflowelemente enthalten. Sie werden durch die Projektabhängigen Aktivitäten erweitert.

5.3.1.2 Geschäftsobjekte

Geschäftsobjekte kapseln die Daten gemäß dem Model des MVC-Muster (vgl. S.14). Über die Geschäftsobjekte werden die Daten aus den verschiedenen Datenbanken bzw. aus den zu integrierenden Anwendungen über MQSeries angesprochen. Geschäftsobjekte ermöglichen also die Verwendung der generell zur Verfügung stehender Daten – gleichgültig vom dahinter liegenden System (Datenbank, SAP, OLAP).

5.3.1.3 Anzeigeobjekte

Anzeigeobjekte sind Java – Klassen, welche die Daten - Schnittstelle vom Client zum Server sowie Server und Client abbilden. Diese Java-Klassen besitzen nur Stringdatentypen bzw. Arrays von Stringdatentypen. Ein Anzeigeobjekt kann weitere Anzeigeobjekte enthalten. Anzeigeobjekte werden wie folgt zum Datenaustausch verwendet:

- Client → Server:
  Die Werte aus der Eingabeseite (z.B. die Eingabefelder in einer Suchmaske) werden ausgelesen, geprüft und in ein Anzeigeobjekt überführt. Dieses Anzeigeobjekt wird in der Evolutionsession hinterlegt, und steht somit im weiteren Verlauf der Anfrageabarbeitung zur Verfügung.
- Server → Client:
  Innerhalb der Aktivitaeten werden die Daten für die nächste anzuzeigende Seite ermittelt. Diese Daten werden aus den Geschäftsobjekten in ein neues Anzeigeobjekt[57] überführt. Das Framework sorgt dafür, dass aus diesem neuen Anzeigeobjekt ein XML-Dokument erzeugt wird (vgl. Kapitel 5.5.4.1), bevor es mittels XSL-Seiten nach HTML oder PDF überführt und an den Client zurückgeschickt wird.

[57] das weitere Anzeigeobjekt enthält

5.4 Evolution als Vorgehensmodell

5.4.1 Definition

„Ein Vorgehensmodell beschreibt auf abstrakte oder generische Weise, wie ein System entwickelt wird, d.h. in welcher Reihenfolge welchen Aktivitäten und Ergebnissen nachgegangen wird“ (vgl. [Oest99]).

Vorgehensmodelle enthalten oft Vorgaben für das Projektmanagement, Evolution jedoch nicht.

Das hier vorgestellte Vorgehensmodell ist *anwendungsfallgetrieben* sowie *iterativ und inkrementel.*

5.4.1.1 Anwendungsfallgetrieben

Die Anwendungsfälle werden am Anfang systematisch erarbeitet und bilden so die Grundlage für das weitere Handeln. Die Anwendungsfälle können priorisiert werden um vorzugeben, welche Anwendungsteile zuerst entwickelt werden.

5.4.1.2 Iterativ und Inkrementell

Iterativ-inkrementelle Vorgehensmodelle zeichnen sich im Gegensatz zum bekannten Wasserfallmodell (siehe unten) dadurch aus, dass sie von Beginn des Entwicklungsprozess berücksichtigen, das sich die Anforderungen an ein IT-System unvollständig sind oder sich im Verlauf ändern.

- Iterativ bedeutet, die Zerlegung der Entwicklungstätigkeit in mehrere gleichartige Schritte – den Prozessen.
- Inkrementell bedeutet in diesem Zusammenhang, dass die verschiedenen Prozesse wiederholt ausgeführt wird

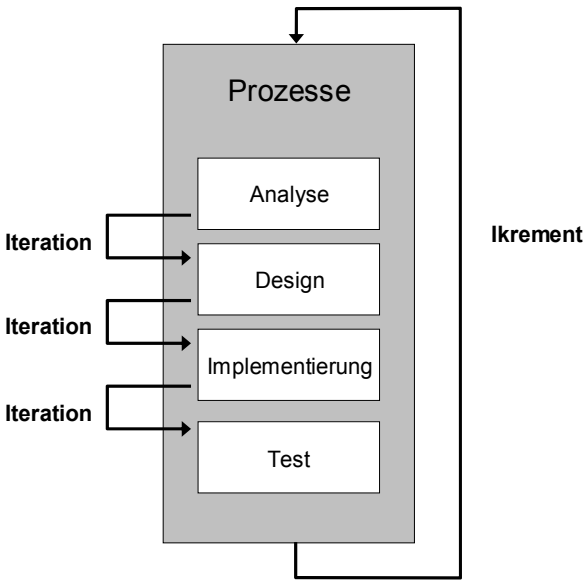


Abbildung 5-2 iterativ und inkrementelles Vorgehensmodell

Zu einem frühen Entwicklungszeitpunkt kann Prototyp - im iterativ und inkrementellen Modell - so ein mit einer gewissen Grundfunktionalität erstellt werden. Nach dem ersten Durchlauf wird dann diese Funktionalität erweitert und die entsprechenden Prozesse werden nochmals durchgeführt, bis schließlich die gewünschte Gesamtfunktionalität erreicht ist.

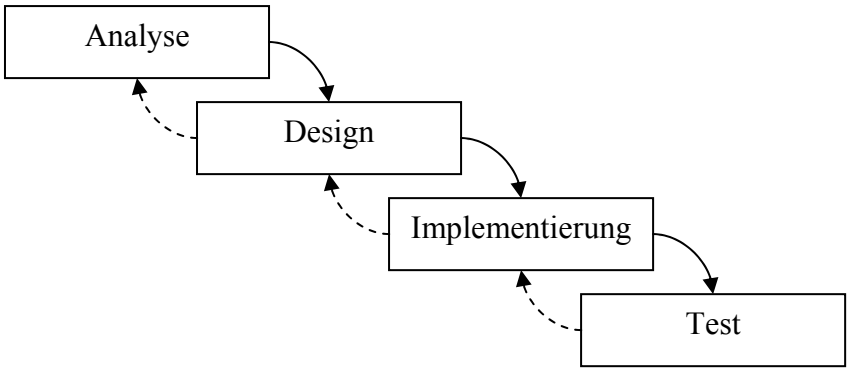


Abbildung 5-3 Phasen des Wasserfallmodells

Die Entwicklungstätigkeiten (Phasen) im Wasserfallmodell hingegen weitgehend unabhängig voneinander entwickelt. Ein Rückgriff ist nur auf die jeweils vorherige Phase vorgesehen. Die obige Abbildung zeigt die Phasen des Wasserfallmodells. Auffällig ist, dass die Phasen des Wasserfallmodells den Prozessen im iterativen und inkrementellen Vorgehensmodell entsprechen. Der wesentliche Unterschied sind die verschiedenen „Inkremente“, also das wiederholte Durchlaufen der Tätigkeiten unter der Verwendung der bereits erhaltenen Erkenntnisse.

Österreich (vgl. [Oest99]) stellt die iterative und inkrementelle Vorgehensweise bei der Entwicklung von Software als die Antwort auf die Probleme des Wasserfallmodells vor: *„In der Praxis ergeben sich viele Problemen erst während der Implementierung oder bei der Systemintegration. Dann wird festgestellt, dass Schnittstellen nicht richtig zusammenpassen oder die Performance nicht ausreicht.“*

5.4.2 Überblick über das Vorgehensmodell

Evolution enthält als Vorgehensmodell folgende Bestandteile:

- Phasen und Iterationen
- Rollen
- Prozesse
- Produkte (Artifacts)

5.4.2.1 Phasen und Iterationen

Phasen stellen eine *zeitliche* Gliederung des Entwicklungsprozesses dar. Jede Phase erfüllt einen definierten Zweck und führt zu einem Ergebnis. Die verschiedenen Phasen werden in Evolution als projektabhängige Meilensteine innerhalb eines Projektes festgelegt. Eine Phase kann aus mehreren Iterationen bestehen.

5.4.2.2 Rollen (Worker)

Die verschiedenen Tätigkeiten sind in Evolution werden verschiedenen Rollen zugeteilt. Während in kleinen Projekten eine Person mehrere Rollen übernehmen kann, werden führen verschiedene Personen eine Rolle aus.

- Analyst

  Der Analyst ist ein Spezialist in einem Fachgebiet, der in enger Zusammenarbeit mit dem Fachbereich die funktionalen Anforderungen an das zu entwickelnde System aufnimmt. Der Analyst ist zuständig für die Erstellung des Pflichtenheftes sowie die Erstellung der Use Cases.

- Designer

  Während der Analysephase arbeitet der Designer mit dem Fachbereich zusammen, um das Aussehen der Anwendung zu bestimmen, Felder und Buttons zu positionieren. Der Designer einer Anwendung ist – neben dem erwarteten Verhalten einer Anwendung – ausschlaggebend für den Erfolg der Anwendung. Er muss das Aussehen der Anwendung so entwerfen, dass die Anwender Tag für Tag gerne mit dem entwickelten System arbeiten.

- Architekt

  Aufgabe des Architekten ist wichtige, Architekturentscheidungen zu treffen und somit wichtige Vorgaben für die Entwickler zugeben. Der Architekt ist derjenige der über die Erweiterung oder die Änderung des Framework (vgl. Kapitel 0) bestimmt.

- Entwickler

  Der Entwickler implementiert den projektabhänigen Code der zu erstellenden Anwendung. Er ist der Anwender des Frameworks, in dem er dessen Komponenten benutzt, sie ggf. erweitert und spezialisierte Klassen entwickelt

- Tester

  Software kann nur erfolgreich eingeführt werden, wenn sie ausgiebig getestet wird. In Evolution ist vorgesehen, dass jeder Entwickler seine entwickelten

Teile auf technische Fehler überprüft. Der Analyst oder ein Sachbearbeiter kann zu einem frühen Projektzeitpunkt die Anwendung aus fachlicher Sicht testen.

5.4.2.3 Prozesse

Prozesse stellen eine *inhaltliche* Gliederung der Entwicklungsaktivitäten dar. Dazu zählen (vgl.):

- Analyse
- Design
- Implementierung
- Test

Ziel der Analyse ist es, die Wünsche und fachliche Anforderungen an ein neues Softwaresystem – aus der Sicht der Auftraggeber - zu ermitteln und zu beschreiben. Coad/Yourdan (vgl [Coad90]) haben den Begriff Problemraum eingeführt, nach dem sich die objektorientierte Analyse mit folgenden Inhalten beschäftigt: Klassen und Objekte identifizieren, Semantik der Klassen und Objekte sowie die Beziehungen zwischen den Objekten und Klassen festlegen.

Die Begriffe objektorientierte Analyse und Design werden in der Fachliteratur eng miteinander verbunden, so dass eine strikte Trennung sehr schwierig ist. Als grundlegende Aussage gilt, dass das Analysemodell im Design „verfeinert“ wird.

Heide Balzert (vgl. [Balz99]) beschreibt die Aufgabe des Entwurfs (Design) als die Realisierung der ermittelten Anforderung der Analyse auf einer Plattform unter den gegebenen technischen Randbedingungen.

Die letztendliche Umsetzung der Anforderungen erfolgt während der Implementierung. Dazu werden zuvor ermittelten und beschriebenen Funktionalitäten in den konkreten Quell-Code überführt. Es entsteht ein ablauffähiges Programm.

Dieses ablauffähige Programm wird im Test-Prozess anhand der funktionalen (ermittelt während der Analyse) Anforderungen als auch den nicht-funktionalen Anforderungen (beispielsweise Performance und Verhalten auf der Zielplattform) gemessen. Die Ergebnisse dieses Tests werden als wichtiger Input für die nächste Inkrement – also den nächsten Durchlauf der Prozesse - verwendet.

5.4.2.4 Artefakte[58]

Die folgende Übersicht enthält die verschiedenen Artefakte (alternativ werden auch die Begriffe Produkte und Arbeitsergebnisse verwendet) der verschiedenen Prozesse. Die verschiedenen Produkte werden später (S. 64ff) erklärt.

Prozess	Artefakt
OOA	Pflichtenheft
	Use Case
	Use Case Diagramm
	EPK
	Aktivitätsdiagramme
	Maskenentwurf
	Klassendiagramm
	Logisches Datenmodell
OOD	Physisches Datenmodell
	Verfeinerte Aktivitätsdiagramm
	Schnittstellenbeschreibung[59]
	HTML-Masken
Implementierung	Konfigurationsfiles
	XSL-Dateien
	Java Code
	Schnittstellenimplementierung

Tabelle 5 Artefakte in Evolution

5.4.3 Der Entwicklungsprozess

Die zuvor genannten Artefakte von Evolution werden in diesem Kapitel beschrieben. Anhand der verschiedenen Prozesse wird konkret gezeigt, wie Evolution zwischen Analyse, Design und Implementierung unterscheidet.

5.4.3.1 Analyse

Pflichtenheft

Die Analysephase beginnt in Evolution mit der Erstellung eines Pflichtenheftes. In diesem Pflichtenheft werden alle funktionalen und nicht-funktionalen Anforderungen an die Anwendung beschrieben. Das Pflichtenheft enthält alle Anforderungen an die Lösung. Es muss so abgefasst sein, dass es als Basis einer anforderungsgerechten Umsetzung dienen kann. Dabei stellt eine Beschreibung des gewünschten Lieferumfangs dar. Anhand des Pflichtenheftes soll dann das fertige Produkt

[58] engl. artifacts
[59] falls weitere Backendzugriffe notwendig sind

abgenommen werden. Die Struktur des Pflichtenheftes kann aus ([Balz99], S)entnommen werden).

Geschäftsprozesse

Im Bereich dieser Architektur wird ein geschäftsprozess-getriebener Ansatz verfolgt. Man betrachtet also die zu erstellende Anwendung im Kontext des oder der jeweiligen fachlichen Geschäftsprozesse, die sie unterstützen soll.

Die Dokumentation der Geschäftsprozesse ist dann wichtig, wenn das zu erstellende System in einem übergreifenden Prozess-Kontext verwendet soll, z.B. als eine Teilaktivität, oder wenn sich vorhandene Geschäftsprozesse ändern. Sofern dies für eine zu erstellende Anwendung nicht der Fall ist, ist die Erstellung nicht zwingend notwendig.

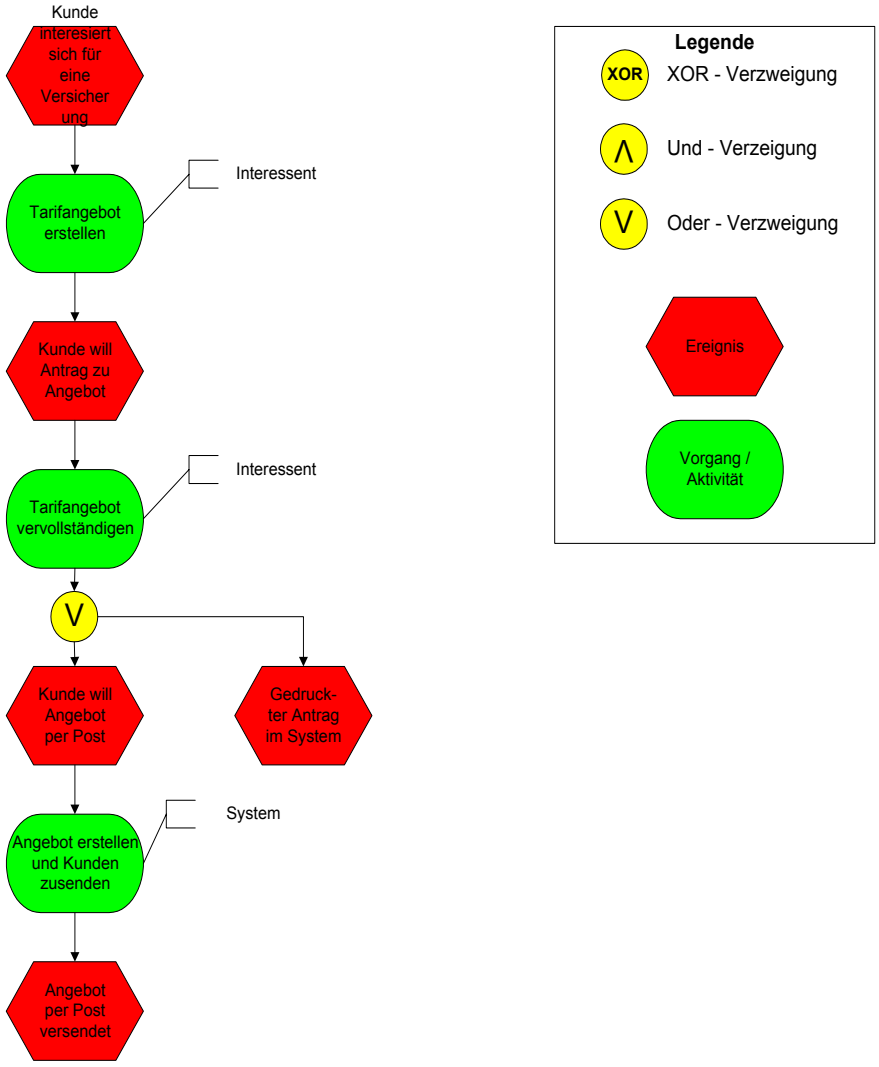


Abbildung 5-4 Geschäftsprozess in EPK-Notation von Aris

Use Cases

Use Cases (Anwendungsfälle) beschränken sich im Gegensatz zu Geschäftsprozessen immer mit der Interaktion eines Anwenders und dem IT-System. Beispielsweise zeigt das Ereignis Angebot per Post unterbreiten, dass nicht in der direkten Interaktion mit dem IT-System erstellt wird.

Use Cases bilden in Evolution die Grundlage für das weitere Handeln und werden wie folgt textuell beschrieben(vgl. [Oest98], S. 209 ff):

- **Akteure**
  Die beteiligten Akteure werden aufgelistet.
- **Auslöser**
  Der Auslöser für den Use Case wird dokumentiert. Dies kann entweder ein vorangehender Use Case oder ein Geschäftsvorfall sein, z.B. ein Kunde teilt die Änderung seiner Anschrift mit.
- **Voraussetzung**
  Einige Use Cases verlangen nach bestimmten Voraussetzungen, die erfüllt sein müssen, z.B. dass der Kunde, der bearbeitet werden soll, schon im System angelegt ist.
- **Nachbedingungen**
  Der Zustand des Systems nach Durchlaufen des Use Cases wird beschrieben. Ein Beispiel ist eine geänderte Adresse.
- **Standardablauf**
  Der Standardablauf beschreibt die einzelnen Arbeitsschritte aus Sicht des Akteurs. Die einzelnen Schritte werden durchnummeriert. Der Standardablauf dokumentiert den am häufigsten auftretenden Arbeitsfluss, eventuelle auftretende Abweichungen werden durch Referenzen auf die entsprechende Stelle im nachfolgenden Abschnitt dokumentiert.
  - **Variationen vom Standardablauf**
    Abweichungen vom Standardablauf werden erläutert.
  - **Ausnahmen und Fehlersituation**
    Beim Arbeitsablauf können Fehlersituationen auftreten, z.B. durch Falscheingaben. An dieser Stelle wird dies dokumentiert.
  - **Plausibilitäten**
    Es werden die Plausibilitäten, die im Rahmen des Use Cases eine Rolle spielen, dokumentiert.
  - **Offene Punkte**
    In diesem Abschnitt kann der Autor für ihn offene Fragen festhalten, die später geklärt werden müssen.
  - **Ansprechpartner/Verantwortlich für die Erstellung**
    Namen desjenigen, der für den Use Case erstellt hat bzw. die Verantwortung dafür trägt.

Use Case Diagramm

Die gefundenen Use Cases werden grafisch dargestellt.

Ein derartiges Diagramm sollte bei komplexeren Systemen erstellt werden, da es einem schnell einen groben Überblick über die funktionalen Anforderungen an das System bietet, und dessen Abgrenzung nach außen (zu anderen Systemen/Aktoren) verdeutlicht

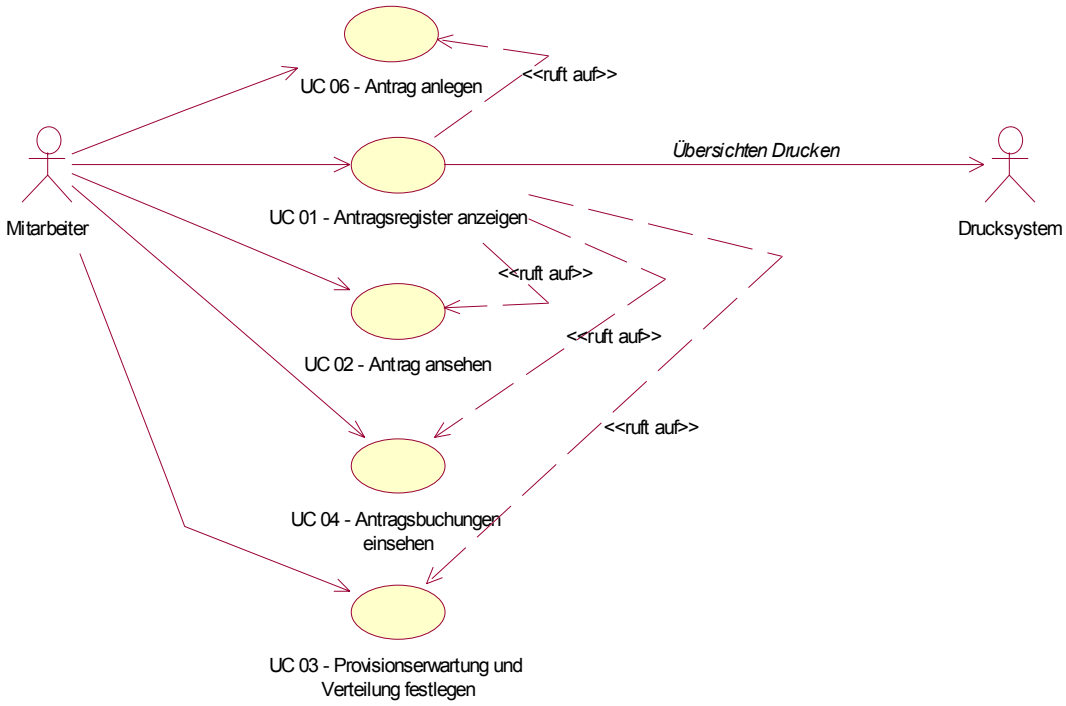


Abbildung 5-5: Beispiel für ein Use Case Diagramm[60]

Die Ovale symbolisieren jeweils einen Use Case, welche mit dem entsprechenden Namen versehen werden. Die außerhalb des Systems liegenden Akteure (sowohl Personen als auch Systeme) werden durch ein „Strichmännchen“ dargestellt (Abgrenzung des Systems). Welcher Aktor mit welchem Use Case in Verbindung steht, lässt sich an den durchgezogenen Linien ablesen. Die Beziehungen der Use Cases untereinander können durch gestrichelte Linien verdeutlicht werden. Die explizite Namensvergabe der Beziehungen verdeutlicht die Abhängigkeiten.

[60] Das Beispiel steht in keinem Zusammenhang mit dem Prototypen

Maskenentwurf

Bereits während der Analyse erstellt der Designer die grafischen Vorlagen der späteren Anwendung. Es ist ausreichend, wenn er dazu ein Grafikprogramm (z.B. Adobe Photoshop) oder Präsentationsprogramm verwendet. Anhand dieser Grafiken kann zusammen mit den Use Cases die Interaktionsreihenfolge in enger Zusammenarbeit mit dem Fachbereich diskutiert werden.

Aktivitätsdiagramme

Aktivitätsdiagramme beschreiben den Ablauf eines Systems mit Hilfe von Aktivitäten (siehe unten, NewProzess und NewSubprozess sind UML-Aktivitäten). Die Zuordnung der UML-Aktivitäten zu den Evolution-Workflowelementen erfolgt über ihre Stereotypen[61] (Prozess, Vorgang, Aktivitaet, AnzeigeAktivitaet). Im Folgenden werden die Aktivitätsdiagramme an dem Beispiel der Software Rational Rose in UML-Notation (vgl. [I.4]) gezeigt. Die Darstellung des Workflows und dessen Struktur (ein Prozess hat ein oder mehrere Vorgänge, ein Vorgang hat mindestens eine Aktiviaet) soll wie folgt am Beispiel der Abbildung 5-6 vorgenommen werden.

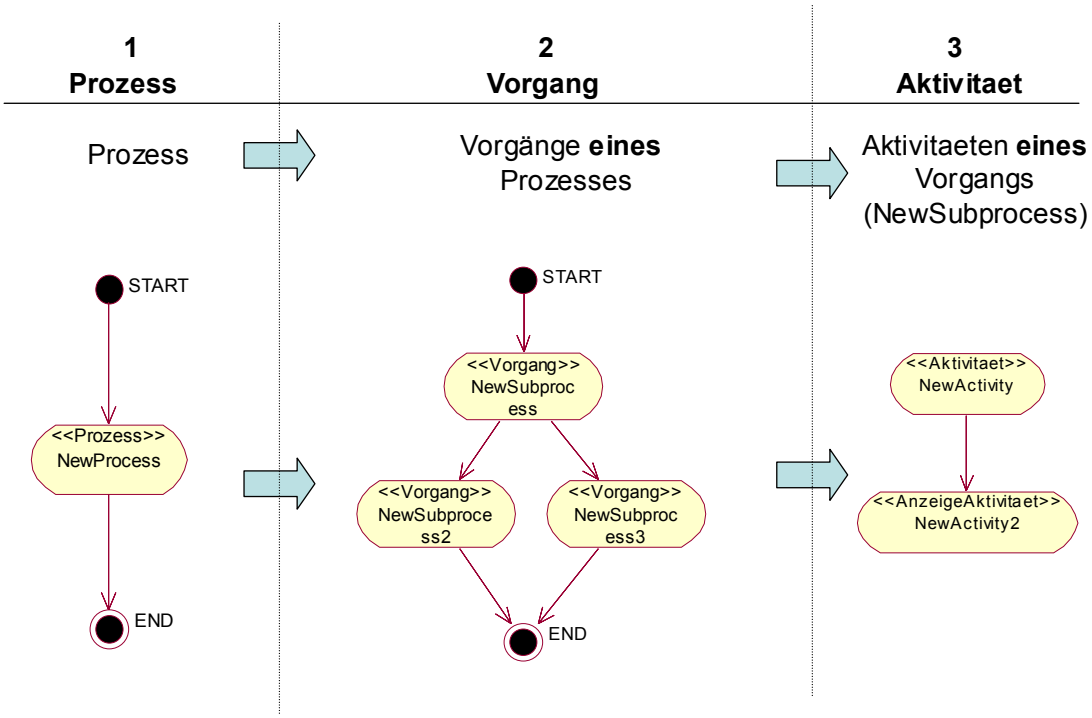


Abbildung 5-6 Workflow in Evolution

[61] Stereotypen dienen der Beschreibung von UML-Symbolen (z.B. einer Klasse, einer Aktivität) etc. In Abbildung 5-6 sind die Stereotypen an den doppelten, eckigen Klammern zu erkennen („<<“, „>>).

Den Startpunkt einer evolutionbasierten Anwendung bildet ein Aktivitätsdiagramm mit einer „Prozess“ -Aktivität – hier NewProcess in Spalte 1.

Für den Prozess ist ein weiteres Aktivitätsdiagramm zu erzeugen (Spalte2). Hier sind die Vorgänge (Aktivitäten mit dem Stereotypen Vorgang) einzutragen. Als Vorgänge sind die gefundenen Use Cases zu verwenden. NewProcess enthält also die Vorgänge Subprocess, Subprocess2, Subprocess3.

Für jeden Vorgang ist jeweils ein weiteres Aktivitätsdiagramm zu erzeugen. Dieses enthält die verschiedenen Aktivitaeten und die AnzeigeAktivitaeten je Vorgang.
Zur Gewährleistung der späteren Code-Generierung gelten folgende Vorschriften:

- Alle Worte, die Prozesse, Vorgänge und Aktivitäten bezeichnen, beginnen mit Großbuchstaben.
- Vorgangs- und Aktivitätsbeschreibungen sollten ein Verb beinhalten
- Die Bezeichnungen der Prozesse, Vorgänge, Aktivitaeten dürfen keine Leerzeichen oder Umlaute enthalten.

Logisches Datenmodell

Während der Analyse ist das logische Datenmodell zu erzeugen. Den Maskenentwürfen sind die Daten für das Datenmodell zu entnehmen. Dazu werden alle Ein- und Ausgabedaten ermittelt. Weitere Unterlagen sind zur Ermittlung der Datenstruktur hinzuzufügen. Beispielsweise können Vertragsvordrucke weitere Informationen zu Verträgen enthalten, die durch die Maskenentwürfe noch nicht abgedeckt sind.
Bereits vorhandene Datenmodelle sind im Bedarfsfall anzupassen bzw. zu erweitern.
Die Beschreibung des Datenmodells kann in Tabellenform oder unter zu Hilfenahme eines Datenbankdesigntools erstellt werden.

Entwicklung der Klassendiagramme

Für Geschäfts- und Anzeigeobjekte sind Klassendiagramme zu erzeugen. Aus den Maksenentwürfen können die zu erstellenden Klassen der Anzeigeobjekte[62] ermittelt werden, während aus dem logischen Datenmodell die Geschätsobjekte identifiziert werden können.

[62] Anzeigeobjekte sind Klassen!

5.4.3.2 Design

Im Design werden die Ergebnisse der Analyse verfeinert. An unserem Beispiel sei es wie folgt dargestellt:

- Verfeinerung der Maskenentwürfe
- Verfeinerung der Aktivitätsdiagramme
- Überführung des logischen in das physikalische Datenbankmodell
- Erweiterung der Klassendiagramme

Verfeinerung der Maskenentwürfe

Die während der Analyse erstellten Masken (in Form einer Grafik) werden in HTML überführt. Diese Masken enthalten statischen Inhalt. Auf eine Darstellung der HTML-Seiten wird an dieser Stelle verzichtet.

Verfeinerung der Aktivitätsdiagramme

Die Aktivitätsdiagramme sind im Design um Zustände zu erweitern[63]. Zustände dienen zum Übergang zwischen Prozess, Vorgängen und Aktivitaeten und somit der Steuerung des Workflows. Die Verbindung zwischen Workflowsteuerung (durch die neuen Zustände) und den HTML-Seiten übernimmt der Steuerungsparamter „ActionName“. Ihm wird der jeweilige Zustand aus dem verfeinerten Aktivitätsdiagramm zugewiesen

Bei der Verfeinerung der Aktivitätsdiagramme gelten folgende Vorschriften:

- Die Zustände werden in GROßSCHRIFT in das Aktivitätsdiagramm eingetragen.
- Wird ein Zustand durch mehrere Wörter beschrieben, sind diese durch das Zeichen „_“ zu verbinden.
- Als Bezeichnung sind sprechende Namen zu verwenden. Als Orientierungshilfe sind die Bezeichnungen der Buttons aus den HTML-Seiten zu verwenden. Sie beziehen sich auf die folgende Aktivität (z.B. START_SUCHE).
- Der Übergang zwischen Aktivitäten erfolgt ausschließlich über Zustände. Zwischen Aktivitaeten und AnzeigeAktivitaeten müssen keine Zustände enthalten sein.

[63] Damit entfernt sich Evolution von der Standardnotation UML(vgl. [I.4]). Aktivitäten sind in UML Sonderformen von Zuständen. Aktivitäten und Zustände werden nach dem UML Standard nicht in einem Diagramm verwendet.

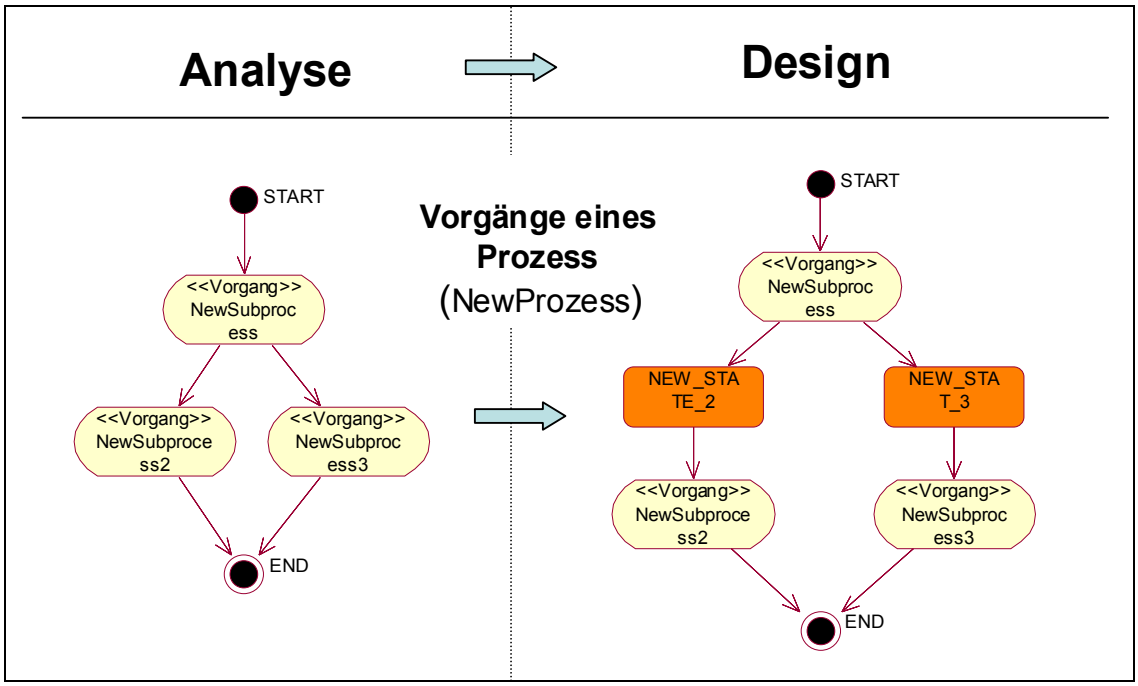


Abbildung 5-7 Verfeinerung der Aktivitätsdiagramme(Prozessebene)

Die obige Abbildung zeigt die Erweiterung des Aktivitätsdiagramms des Prozesses „NewProcess“ auf Vorgangsebene. Die hinzugekommenen Zustände sind farblich hervorgehoben. Diese Zustände entsprechen einer Auswahl eines Anwenders innerhalb des Vorgangs „NewSubprocess“. Dieser Vorgang muss diese Zustände als Ausgangszustände enthalten:

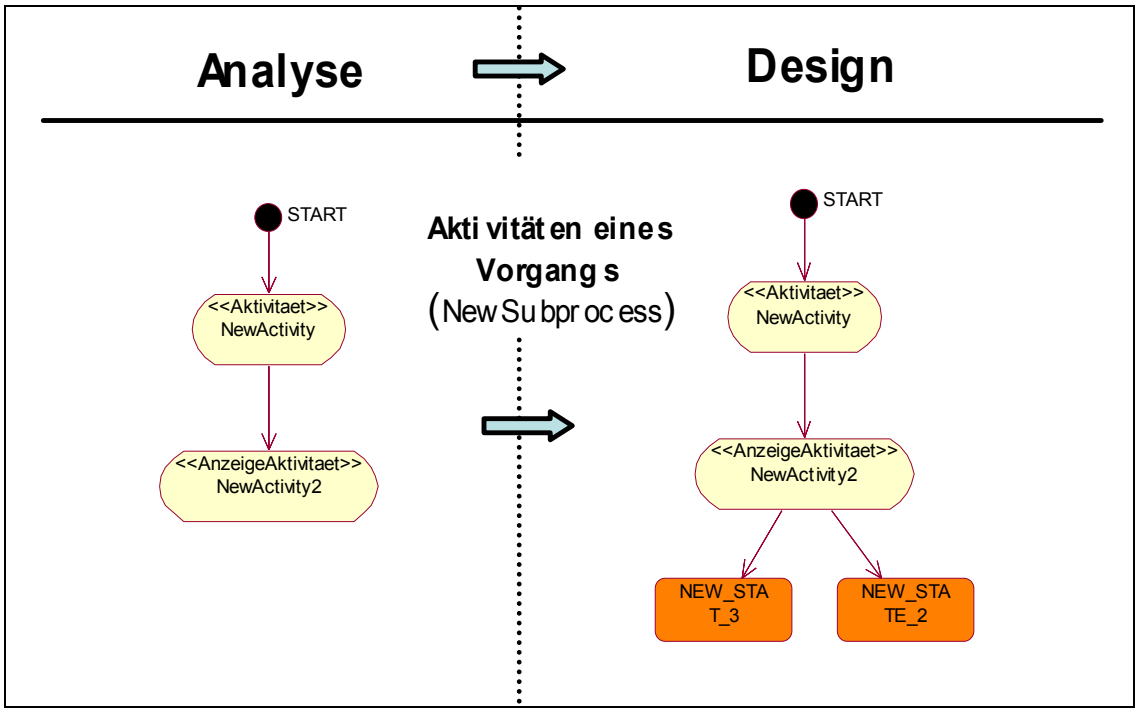


Abbildung 5-8 Verfeinerung der Aktivitätsdiagramme (Vorgangsebene)

Überführung des logischen in das physische Datenmodell

Das logische Datenmodell ist in das physikalische zu überführen. Dazu sind n:m Beziehungen aufzulösen und Datentypen festzulegen. Das physikalische Datenmodell enthält alle anzulegenden Tabellen. Sie können über die Data Definition Language (DDL) für relationale Datenbanken erzeugt werden.

Erweiterung der Klassendiagramme

Nachdem die zur Verfügung stehenden Datenbanken bzw. Daten beschrieben sind, ist ein Klassendiagramm für die Geschäftsobjekte zu erweitern. Dieses Klassendiagramm gibt Auskunft über die Methoden, die ein Entwickler zum Datenzugriff verwenden kann und enthält die Kardinalitäten zwischen den Geschäftsobjekten.

Das Klassendiagramm der Anzeigeobjekte gibt Auskunft über deren Zusammensetzung (ein Anzeigeobjekt kann wiederum mehrere enthalten) sowie deren Kardinalitäten. Für jeden Subprozess ist mindestens ein Anzeigeobjekt zu erstellen[64].

5.4.3.3 Implementierung

Ziel der Implementierung ist es ein lauffähigen Prototypen zu erstellen. Dazu muss ein Entwickler Evolution als Framework (vgl. S. 78 ff) kennen. Der Übergang zwischen Implementierung und Framework ist fließend. In diesem Abschnitt werden die Stellen beschrieben, an denen ein Entwickler die Implementierung des projektabhängigen Code vornehmen kann bzw. muss, während in dem Kapitel „Evolution als Framework“ die Abarbeitung einer Anfrage und die wesentlichsten Bestandteile des Frameworks beschreibt.

Zur Implementierung der Logik sind folgende Schritte notewendig:

- Generierung des Java-Codes aus dem Aktivitätsdiagramm
- Entwicklung der Geschäftsobjekte
- Entwicklung der Anzeigeobjekte.
- Erweitern des generierten Java-Codes
- Erzeugung der XSL-Seiten[65]
- Konfiguration der Anwendung (über XML)[66]

---

[64] Anhand des Klassennamens wird der einzuleitende Subvorgang ermittelt.

[65] Wie aus einem Anzeigeobjekt ein XML-Baum aufgebaut wird, ist im Kapitel 5.5.4.1 Der XML-Generator beschrieben.

[66] siehe Kapitel 5.5.3 Konfiguration, S. 82f

Generierung Java-Code

Die Generierung des Workflows ist derzeit für die Software Rational Rose und die Java-Entwicklung Visual Age for Java (VAJ) von IBM ausgelegt. Rational Rose und Visual Age for Java bieten jeweils Programmierschnittstellen (vgl. Kapitel Application Programming Interfaces4.4.5, 56) an. Rational Rose erlaubt es, die erzeugten Diagramme über eine API – unter Verwendung von COM - auszulesen. Visual Age for Java bietet mit der Tool-API die Möglichkeit Java-Code direkt in den Workspace[67] der Entwicklungsumgebung zu schreiben.

Den Übergang zwischen Rational Rose und VAJ bietet das zum Framework gehörende Programm „Rose2Java“. Dieses Programm liest die Aktivitätsdiagramme von Rational Rose über eine COM-Bridge aus und erzeugt eine XML Datei. Inhalt dieser XML-Datei ist die Beschreibung des Workflows – also der angelegte Prozess, dessen Vorgänge sowie deren Aktivitäten.

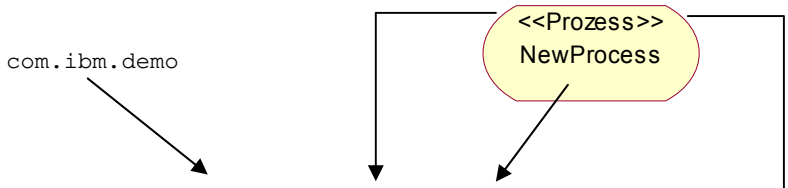
In einem weiteren Schritt[68] wird diese XML Datei ausgelesen und anhand dessen Inhalt Paketstruktur und fertige Java-Klassen samt Methoden generiert.

Zur Zuordnung zu den Java-Klassen und der Paketstruktur werden die Stereotypen des Aktivitätsdiagramms verwendet.

Stereotyp in UML	Java-Basisklasse	Paket
Prozess	Process	<Eingabeparameter im Programm Rose2Java>.processes
Vorgang	SubProcess	<Eingabeparameter im Programm Rose2Java>.processes
Aktivitaet	Activity	<Eingabeparameter im Programm Rose2Java>.activities

Beispiel:

Eingabeparameter in Rose 2 Java: **UML-Aktivität**



JavaKlasse: com.ibm.demo.processes.NewProcess extends

Basisklasse: com.ibm.evolution.control.Process

[67] Visual Age for Java speichert den Quellcode nicht direkt in einer Verzeichnisstruktur, sondern in einem eigenen „Datenbereich“ im Workspace bzw. im Repository. Das Repository ist der Datenbereich für releases und Versionen. Der Workspace enthält den aktuelle Code (auch ohne Versionsnummer), die der Entwickler verwendet.

[68] Der Zwischenschritt über die XML-Datei folgenden Vorteil: Zur Generierung können auch andere Designtools verwendet werden, sofern sie diese XML-Datei erzeugen können.

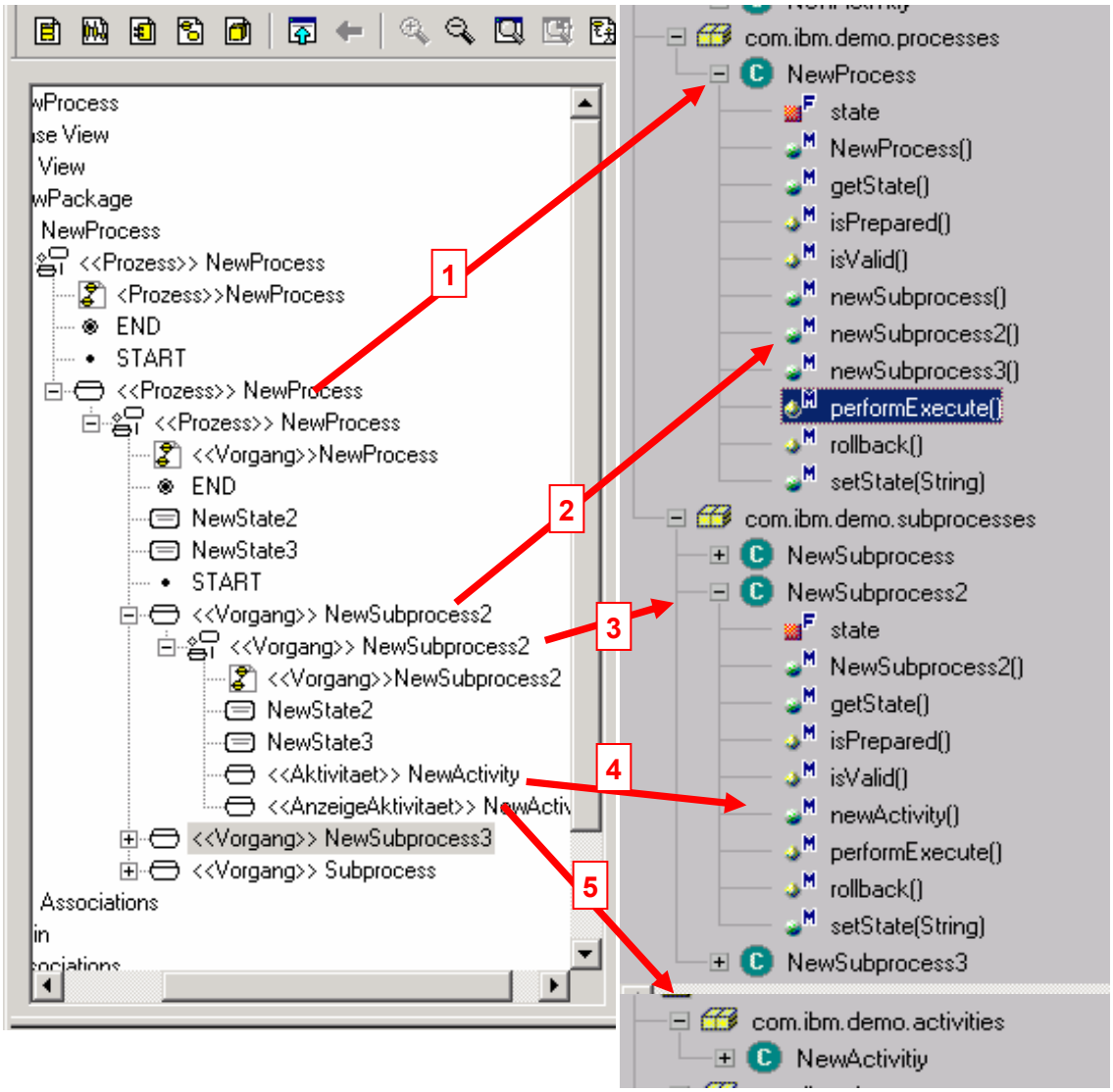


Abbildung 5-9 Codegenerierung: Pakete, Klassen und Methoden

Die obige Abbildung zeigt, wie der Workflow aus den Aktivitätsdiagrammen in Java-Code umsetzt wird. Der linke Teil der Abbildung enhält die Elemente[69] des Aktivitätsdiagramms[70]. Für jede Aktivität[71] wird jeweils eine Klasse in dem entsprechenden Paket angelegt.

Für den Übergang zwischen Prozess und Vorgang wird jeweils eine Methode[72] generiert (2). Sie sorgt für den Datenaustausch mit der Vorgangsklasse (3) und das Ausführen des entsprechenden Vorgangs. Entsprechend umgesetzt ist der Übergang zwischen Vorgängen und Aktivitäten (4,5).

[69] Zustände und Aktivitäten
[70] Abgebildet wird gemaß MVC-Muster das Model aus Ratiional Rose.
[71] Process, Subprocess, Activity
[72] In diesem Beispiel die Methoden newSubprocess, newSubprocess2 und newSubprocess3. Diese Methoden in der Methode performExecute() aufgerufen. Zur Steuerung werden dazu die Zustände verwendet.

Entwicklung der Geschäftsobjekte

Die zuvor beschriebene Funktionalität der Geschäftsobjekte ist jetzt zu implementieren. Dazu verbindet der Entwickler die Geschäftsobjekte mit den verschiedenen Datenquellen. Zwischen den Geschäftsobjekten und den Datenquellen werden „Datenzugriffklassen“ verwendet. Sie kapseln die Details des Datenzugriffs – bei relationalen Datenbanken beispielsweise die SQL-Statements.

Für den Datenzugriff über MQSeries werden jeweils Java Ein- und Ausgabedatentypen festgelegt. – Visual Age for Java bietet entsprechende Tools an, um beispielsweise aus einer Cobol Copy -Strecke. einen Java Datentyp zu generieren.

Neben den Geschäftsobjekten, die die Daten kapseln, müssen unter Umständen weitere entwickelt werden, deren Funktionalität noch nicht abgedeckt sind. Beispielsweise enthält der Prototyp solche weiteren Klassen z.B. Menubilder, ContractHolder

Entwicklung der Anzeigeobjekte

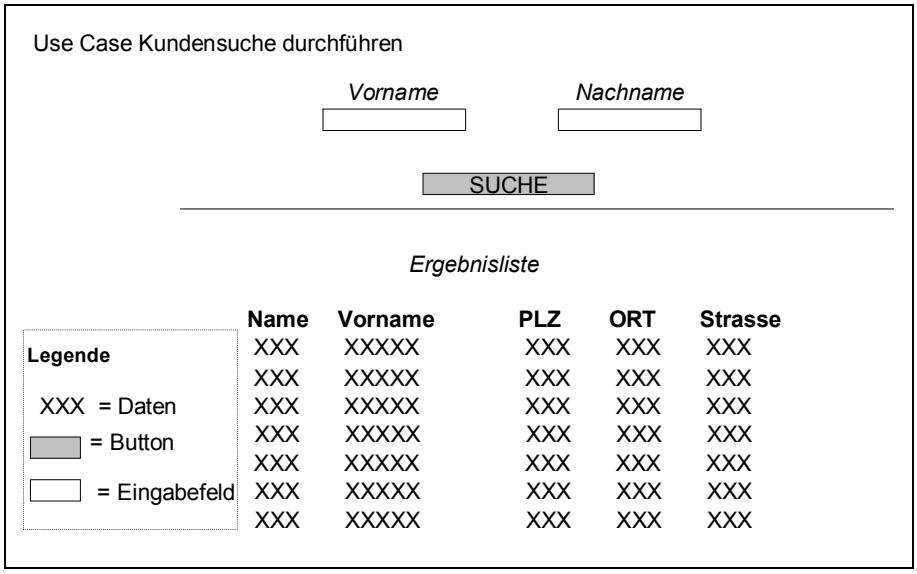


Abbildung 5-10 Entwicklung der Anzeigeobjekte

Die Anzeigeobjekte (AO’s) sind in Java Klassen zu überführen. Zuvor solle jedoch noch einmal an einem Beispiel gezeigt werden, wie sie „gefunden“ werden:
Für jeden Vorgang ist ein „Hauptanzeigeobjekt[73]“ zu erstellen. – Hier beispielsweise das Anzeigeobjekt Kundensuche[74]. Als Bestandteile des Anzeigeobjektes[75] kann die

[73] Das Hauptanzeigeobjekt enthält alle weiteren Anzeigeobjekte einer Seite. Es handelt sich also im ein Container-Klasse.

Eingabemaske (Felder: Vorname, Nachname) sowie eine Ergebnisliste identifiziert werden.

Das Anzeigeobjekt Ergebnisliste enthält ein Array von Ergebniszeilen – diese Ergebniszeile wird wiederum als Anzeigeobjekt mit folgenden Feldern abgebildet:

Name, Vorname, PLZ, Ort, Strasse

Aus dem vorherigen Beispiel könnten also folgende Java-Anzeigeobjekte verwendet werden[76]:

```
public class Kundensuche extends // ein Beispiel in Anlehnung an den Protoypen
             com.ibm.evolution.display.DisplayObject {

       private Suchmaske     input;           // 1:1 Beziehung
       private Ergebnisliste output;          // 1:1 Beziehung
}
public class Suchmaske extends
             com.ibm.evolution.display.DisplayObject {
                                                    //Felder der Suchmaske
       private java.lang.String Vorname;
       private java.lang.String Nachname;
}

public class Ergebnisliste extends
             com.ibm.evolution.display.DisplayObject {
                                                    // Die Liste besteht aus einem
       private Ergebniszeile[] eintrag;       // Array vom Typ Ergebniszeile

}


public class Ergebniszeile
             extends com.ibm.evolution.display.DisplayObject {

       private String Vorname;                //Felder einer Zeile
       private String Nachname;
       private String String;
       private String PLZ;
       private String Ort;
}
```

Abbildung 5-11 Beispiel Anzeigeobjekt (Java-Code)

[74] Ein vereinfachtes Beispiel, das ähnlich im Prototypen vorkommt.
[75] zur Erinnerung ein Anzeigeobjekt kann weitere Anzeigeobjekte enthalten
[76] Es werden nur Variablendeklarationen gezeigt. Die entsprechenden Zugriffsmethoden müssten noch erzeugt werden.

Erweiterung des generierten Code

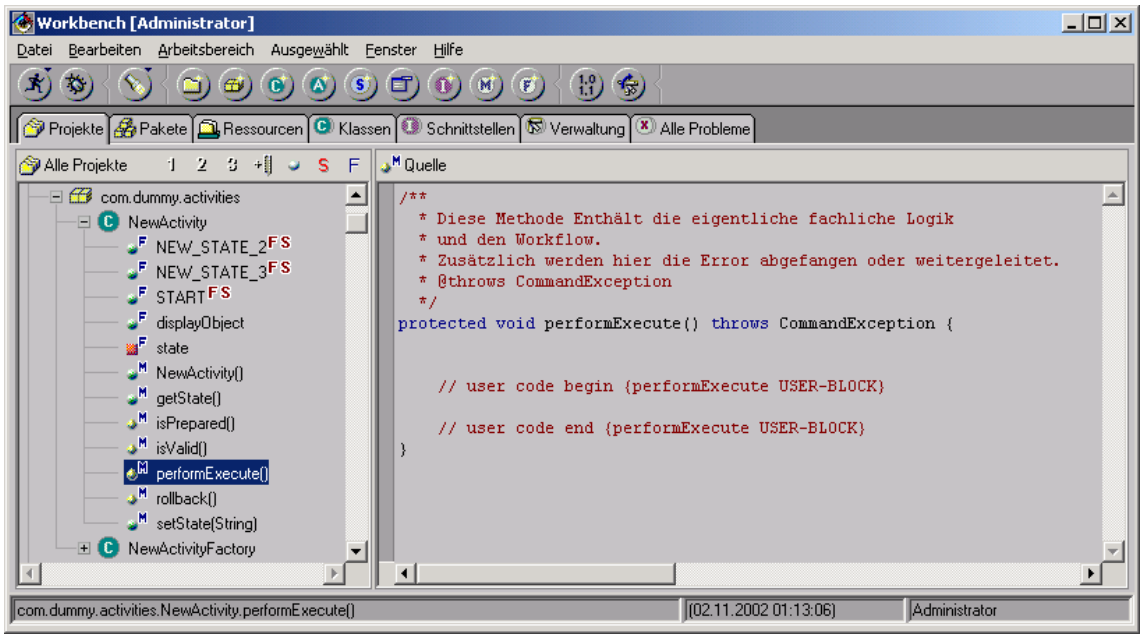


Abbildung 5-12 Beispiel generierter Code: NewActivity

Die obige Abbildung zeigt einen Ausschnitt aus dem generierten Code der Klasse NewActivity.

Dieser Klasse können die Methoden isPrepared(), isVaild(), performExecute() und setState() sowie die im Aktivitätsdiagramm erstellten Zustände[77] (START, NEW_STATE_2, NEW_STATE_3) entnommen werden.

Diese Methoden werden aus der gemeinsamen Basisklasse der Workflowelemente (Process, Subprocess, Activity) – der Klasse Command vererbt.. Alle Process-, Subprocess- und Activityklassen enthalten diese Methoden. An dieser Stelle wird die Verwendung des Befehlsmusters ersichtlich.

Innerhalb dieser Methoden und den dafür vorgesehenen „user code sections“[78]

```
//user code begin..
//user code end ..
```

implementiert der Entwickler die Logik. Während isPlausibel() sowie isValid() der Fehlerbehandlung dienen (vgl. S 81), wird die eigentliche Funktionalität in die Methode performExecute() eingetragen. Hierzu verwendet der Programmierer die zuvor erstellten Geschäfts-, Anzeigeobjekte und setzt die gültigen Endzustände gemäß dem Aktivitäts-diagramm.

---

[77] sie werden als String-Konstanten generiert

[78] Diese user code section bietet innerhalb von Visual Age for Java den Vorteil, das bei einer Neugenerierung des Workflows der bereits geschriebene Quellcode erhalten bleibt.

Erzeugung der XSL-Seiten

Im Kapitel Grundlagen wurden die benötigten Informationen zur Anzeige einer Seite mittels XML und XSL erläutert. Um XML und XSL zur Ausgabeaufbereitung zu verwenden ist im folgenden Kapitel der XML-Generator beschrieben. Er zeigt, wie aus einem Anzeigeobjekt ein XML-Baum erzeugt wird.

Konfiguration der Anwendung

Als wesentlicher Bestandteil des Frameworks wird die Struktur der Konfiguration im Kapitel Evolution als Framework erklärt.

Zwingend erforderlich sind in der Konfiguration die Einstellungen für den zu startenden Prozess, sowie die Zuordnung zwischen Anzeigeobjekten, Zuständen und den daraus resultierenden XSL-Dateien, die zur Transformation verwendet werden (vgl. Kapitel 6.2.5.6, S. 107). Ferner sind Einstellungen zur Datenbank, MQSeries, Logging (sollen Warnungen, Fehler auggezeichenet werden, und wenn ja wie – Datei, Konsole oder sollen E-Mails verschickt werden. (vgl. beigefügte CD-Rom)

5.5 Evolution als Framework

Evolution als Framework sorgt im Wesentlichen für die Abarbeitung des Workflows einer Anfrage und zur Anzeige neuer Daten.

Das Framework enthält zahlreiche Klassen und Utilities (z.B. Logging, Tracing etc.), die den Entwickler bei seiner Tätigkeit unterstützen.

Innerhalb dieses Kapitels werden die wichtigsten Bestandteile des Frameworks sowie die Abarbeitung einer Anfrage zur Laufzeit erklärt. Hierzu zählen im Einzelnen

- Steuerung,
- Fehlerbehandlung,
- Konfiguration des Frameworks sowie die verschiedenen
- Generatoren.

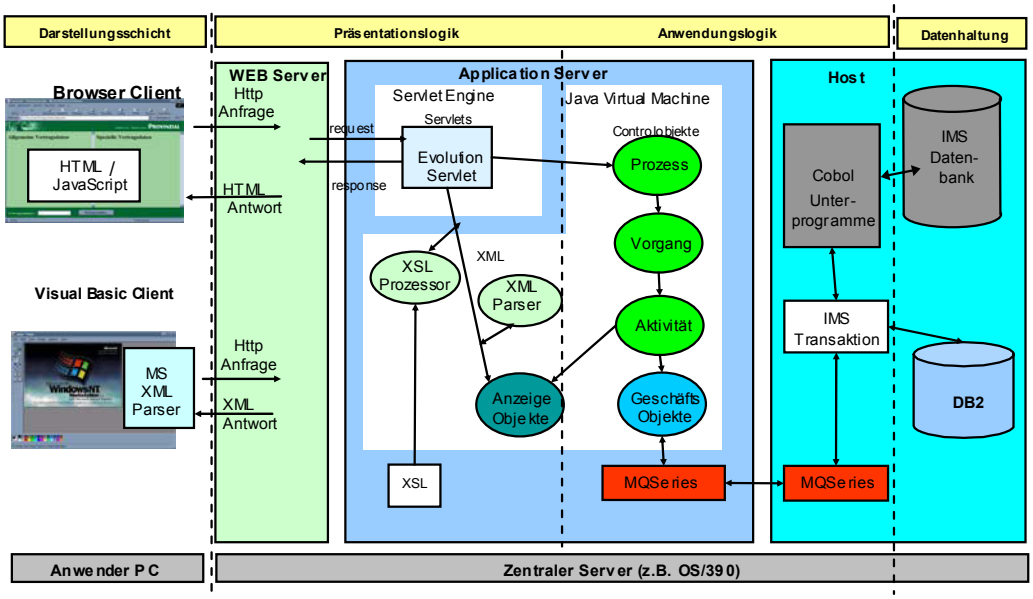


Abbildung 5-13 Abarbeitung einer Anfrage an Evolution

Die obige Abbildung zeigt einen Überblick, wie innerhalb einer Anfrage Workflowelemente, Anzeige- und Geschäftsobjekte verwendet werden. Die Backendanbindung ist hier über MQSeries angedeutet - beispielsweise werden Cobol-Programme aufgerufen.

5.5.1 Steuerung

Als technische Steuerungsroutine kommt ein Java-Servlet zum Einsatz – das EvolutionServlet (vgl. Abbildung 5-13). Diesem Servlet werden mit jeder Anfrage die vom User eingegebenen Daten sowie die Steuerungsparameter übermittelt:

Parameter	Inhalt (Beispiel)	Bedeutung
actionName	Start	Zustand, der zum nächsten Aktivität führt – er entspricht dem Zustand aus dem Aktivitätsdiagramm
nameDO	com.ibm.icis.display.SearchDO	Anzeigeobjekt der aufrufenden Seite
serviceID	internet_customer_search	Bezeichnet den aktuellen Mandanten

Anhand dieser Steuerungsparameter überführt das EvolutionServlet[79] zunächst die eingegebenen Daten (beispielsweise die eingegebenen Suchkriterien einer Suchmaske) in ein Anzeigeobjekt.

Dieses Anzeigeobjekt wird der EvolutionSession[80] unter dem Parameter input hinzugefügt. Das Anzeigeobjekt bietet die verschiedenen getXXX() Methoden, um auf

[79] bzw. die Komponente RequestParser

den Inhalt dieses Objektes während der Prozessverarbeitung zuzugreifen, z.B getName().

Der Parameter serviceID wird den Fabrikklassen (vgl. Abbildung 2-6) übergeben, damit sie zur Laufzeit den Erzeugungsprozess der zu verwendenden Klassen ermöglichen. Zur Erinnerung sei das Beispiel Provisionsdatenerfassung erwähnt, bei denen unterschiedliche Mandanten unterschiedliche Implementierungen vorsehen.

Über den „actionName“ bestimmt das Framework die nächste auszuführende Aktivität. Der Actionparameter entspricht einem Zustand aus den Aktivitätsdiagrammen. Der Anwender bestimmt den Inhalt dieses Parameters durch Betätigen eines Buttons bzw. Verwendung eines Hyperlinks.

Sind alle Parameter ausgelesen bzw. bestimmt, bestimmt das Framework den zu startenden Prozess. Der Prozess ist für fachliche Steuerung des Workflows – also der Vorgänge und Aktivitaeten zuständig ist.
Während der Prozessverarbeitung bieten die Worklowklassen einen Fehlerbehandlungs-mechanismus (siehe unten).

Sind alle eingegebenen Daten gültig und vollständig, glenagt der Workflow - über den Prozess und Vorgang - bis zu einer Aktivitaet.
Innerhalb der performExecute() Methode einer Aktivitaet wird die vom Entwickler implementierte Funktionalität ausgeführt, beispielsweise die Umsetzung der Kunden-suche.
Anhand der eingegebenen Suchkriterien der vorherigen Seite wird die Suche durchgeführt. Hierzu werden die Geschäftsobjekte – die die Daten der verschiedenen Backendsysteme kapseln – verwendet.

Der Teil der Daten, die aus den Geschäftsobjekten angezeigt werden soll, wird unterhalb des Hauptanzeigeobjektes[81] abgelegt. Über die Methode „setStauts()“ setzt der Entwickler den aktuellen Status. In Verbindung mit dem Hauptanzeigeobjekt

80 Bei der Evolutionsession handelt es sich um ein gekapseltes Objekt innerhalb der HTTP-Session. Sie wird zum Datenaustausch über die verschiedenen Anfragen verwendet.

[81] Container Klasse für weitere Anzeigeobjekte

bestimmt der Status das zu Transformatioin zu verwendende XSL-Dokument[82]. Zuvor wird vom Framework aus dem neuen Hauptanzeigeobjekt eine XML-Struktur im Hauptspeicher – in Form eines DOM-Baums (vgl. Kapitel 5.5.4.1) aufgebaut.

5.5.2 Fehlerbehandlung

Das Framework bietet grundsätzlich drei Fehlerklassen, die Einfluss auf den Ablauf des Workflows besitzen. Dazu zählen:

- FatalException
  Wird eine FatalException geworfen, wird dem Anwender eine Standard-Fehlerseite „Bitte versuchen sie es später noch einmal“ angezeigt. Mit dieser Fehlerseite wird der Workflow abgebrochen. Diese Fehlermeldung ist beispielsweise zu verwenden, wenn Datenbanken nicht erreichbar sind.
- NotPreparedException
  Sie ist zu verwenden, wenn der Anwender nicht alle Pflichtfelder ausgefüllt hat. Dieser Ausnahme ist eine Fehlermeldung hinzuzufügen. Das Framework stellt einen automatischen Mechanismus bereit, der die letzte Seite, zusammen mit der Fehlermeldung anzeigt.
- InvalidException
  Entspricht eine Eingabe des Anwenders nicht seiner Plausibilitätsbestimmung verwendet der Entwickler diese Ausnahme. Die Fehlermeldung kann dieser Ausnahme hinzugefügt werden. Auch hier stellt das Framework einen Mechanismus bereite, der vorherige Seite – mit der eingetragenen Fehlermeldung – anzeigt.

Diese Fehler können innerhalb der performExecute() Methode geworfen werden.
Für die Invalid- und NotPreparedException bieten die Workflowklassen die Methoden isPlausibel() bzw. isPrepared() an (vgl. Abbildung 5-12). Eine Implementierung dieser beiden Methoden erspart evtl. Backendzugriffe, da sie vor der performExecute()[83] Methode aufgerufen werden.

[82] Die Zuordnung zwischen Anzeigeobjekt und Zustand zu einer XSL Seite ist Bestandteil der Konfiguration
[83] hier die Geschäftsobjekte verwendet werden

5.5.3 Konfiguration

Evolution als Framework besitzt eine sehr flexible Konfigurationsmöglichkeit. Die Konfiguration basiert auf einer XML-Datei (configuration.xml[84]). Diese Datei kann Verweise auf weitere Konfigurationsdateien enthalten. Beispielsweise können für die unterschiedlichen Mandanten unterschiedliche Konfigurationsdateien verwendet werden.

Die Konfigurationsdokumente werden beim ersten Start des Steuerungsservlets einmalig rekursiv eingelesen und zur Laufzeit im Speicher gehalten.
Ein erneutes Laden der Dokumente kann manuell über ein Administrationsservlet getriggert werden. So können neue Konfigurationen ohne Neustart der Applikation aktiviert werden.

5.5.3.1 Aufbau

Grundsätzlich besteht die Konfiguration aus einer „Category“, die „Attribute” und „Category” (XML) Elemente enthält. An dieser Stelle sei auf den Namenskonflikt hingewiesen:
Das Konfigurationselement „Attribute“ besitzt ein (XML) Attribut „Name“. Es kann nicht weiter unterteilt werden.
Die „Category“ gruppiert verschiedene „Attribute“ oder „Category“ Elemente. z.B. die Zugangsdaten für die Datenbank[85]:

„Attribute“ als XML-Element

```
<Category NAME="database">
      <Attribute NAME="Version">1.0</Attribute>
      <Attribute
NAME="driver">COM.ibm.db2.jdbc.app.DB2Driver
      </Attribute>
      <Attribute NAME="url">jdbc:db2:ICIS</Attribute>
      <Attribute NAME="user">db2admin</Attribute>
      <Attribute NAME="passwd">db2admin</Attribute>
</Category>
```

Abbildung 5-14 Beispiel Konfiguration

Eine Category kann ferner dazu genutzt werden, Konfigurationsparameter auf verschiedene Dateien aufzuteilen. (vgl. Kapitel 0 Konfiguration des Prototypen)

[84] wo sich diese Datei befindet, wir in den Konfigurationsparametern des Evolutionservlets festgelegt
[85] Diese Einstellungen könnten wieder einem Mandanten zugeordnet sein.

5.5.3.2 Lesen der Konfigurationsparameter

Wäre die obige Category „Database“ in der Ausgangskonfigurationsdatei „configuration.xml“ vorhanden, könnte der zu verwendende Datenbanktreiber über

```
String driver = Configuration.readAttribute(„database&driver“)
```

ausgelesen werden. Eine Instanziierung der Confiugration Klasse ist nicht notwendig.[86]

5.5.4 Generatoren

Das Framework enthält verschiedene Generatoren, die für die gewünschte Transformation vornehmen. Grundlage für den HTML- bzw. PDF-Generator[87] ist ein XML-Baum sowie ein XSL-Dokument[88]. Zum Verständnis wie aus einem Anzeigeobjekt ein XML-Baum erzeugt wird, wird der XML-Generator als Bestandteil des Frameworks vorgestellt.

5.5.4.1 Der XML-Generator

Der benötigte XML-Baum wird durch den XML-Generator erstellt. Er generiert aus dem Hauptanzeigeobjekt – einer Java-Klasse - das während des Workflows angelegt wurde ein XML-Baum.

Wichtig an dieser Stelle ist, wie ein Anzeigeobjekt in XML (in Elemente und Attribute) überführt wird:

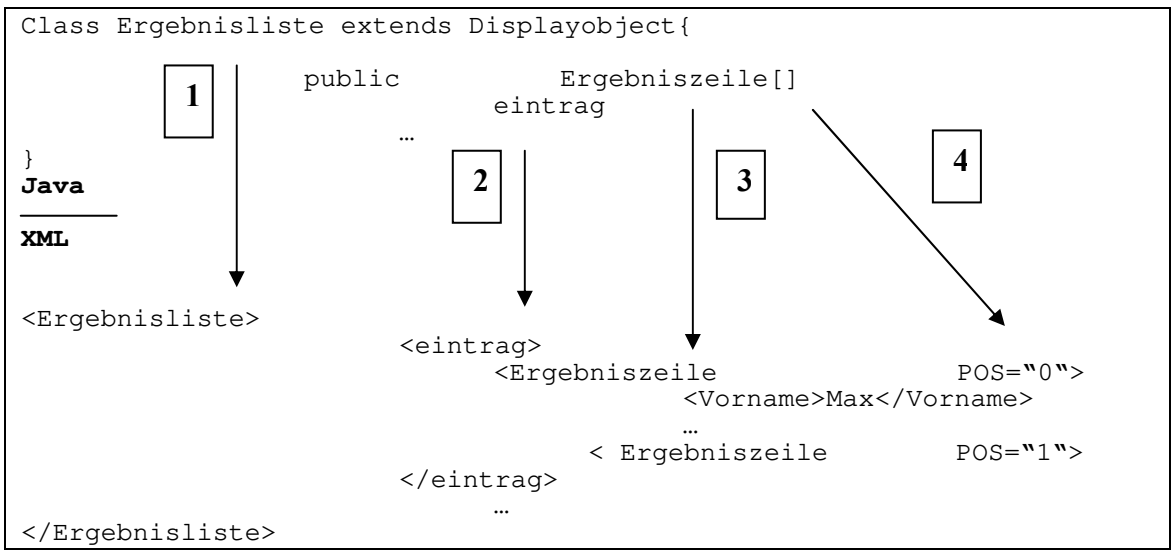


Abbildung 5-15 Beispiel Umsetzung eines Java-Anzeigeobjektes in XML

[86] Die readAttribut Methode ist statisch.
[87] HTML- und PDF-Generator kapseln die Transformation des XML-Baumes in HTML bzw. PDF. Ihnen liegen unterschiedliche XSLT-Prozessoren zu Grunde.
[88] Der Verweis auf dieses Dokument ist in dem Hauptanzeigeobjekt enthalten.

Die obige Abbildung zeigt, die Überführung[89] des Beispiel-Anzeigeobjektes Ergebnisliste (vgl. Abbildung 5-10) in XML-Format. Die wie folgt:

(1) Der Klassenname wird als Element verwendet
(2) Der Variablenname wird als Element abgebildet
(3) Der Datentyp (die Klasse) der Variablen werden als Elemente in (2) abgebildet
(4) Werden Arrays innerhalb eines Anzeigeobjektes verwendet, wird das Element für den Datentyp (3) um das Attribut POS erweitert. Der POS-Wert wird durch den aktuellen Arrayeintag bestimmt, er beginnt bei 0.

Auf eine weitergehende Beschreibung weiterer Komponenten (z.B die die Code-generierung ermöglichen) von Evolution als Framework wird verzichtet.

[89] Das Beispiel wurde stark vereinfacht und wichtige Teile weggelassen – symbolisiert durch „…“ Beispielsweise besitzt jeweils ein Element des EinSubmenuAnzeigeobjekt Bezeichnungen der Submenues, Ein verweis auf eine Grafik und weitere Parameter.

6 Prototyp

6.1 Anwendung im Applikationserver

6.1.1 Problemstellung

6.1.1.1 Zielgruppe

Dieser Prototyp soll als Beispielimplementierung von Evolution innerhalb der Versicherungsbranche dienen. Es ist die Webanwendung „Kundenspiegel“ zu erstellen.

6.1.1.2 Überblick über die Problemstellung

Der Kundenspiegel ist als Inter- bzw. Intranetanwendung zu verstehen, die Versicherungen einsetzen können, um Maklern Zugriff auf ihr IT-System zu gewähren. Im Besonderen soll Ihnen die Möglichkeit gegeben werden, Informationen ihrer Kunden über ein Web-Interface abzufragen und Änderungen der Kundendaten vorzunehmen.
Bei einem Makler handelt es sich im Gegensatz zu einem Agenten um einen versicherungsexternen und unabhängigen Kundenberater. Er vermittelt Verträge zwischen Kunden und den verschiedenen Versicherungsunternehmen auf Provisionsbasis.

Als Versicherungsunternehmen ist es aus folgenden Gründen sinnvoll eine Webanwendung wie den Kundenspiegel bereitzustellen:

a) Entlastung der eigenen Mitarbeiter

Der Makler ist primärer Ansprechpartner für die Versicherungskunden. Änderungen der Kundendaten (beispielsweise Adressänderungen) gehen bei ihm zuerst ein. Reicht der Makler diese Änderung beispielsweise via Telefon oder per Post weiter, führen die Sachbearbeiter der Versicherung die notwendigen Arbeitsschritte aus. Bietet die Versicherung dem Makler hingegen ein Werkzeug, das er „online“ nutzen kann, ist eine Bearbeitung durch die Sachbearbeiter nicht mehr nötig.

b) Bindung der Makler an ein Versicherungsunternehmen

Ein Makler ist nicht an ein bestimmtes Versicherungsunternehmen gebunden. Bietet eine Versicherung ihm den besonderen Service einer komfortablen Kundenverwaltung, wird er dieses Versicherungsunternehmen evtl. einer anderen vorziehen und vermehrt Verträge abschließen.

6.1.2 Anforderungen

Die Anforderungen werden in Evolution über die Use Cases beschrieben. Ein Beispiel für den „evolutionkonformen“ <Use Case Kundensuche durchführen> ist im Anhang, sowie die restlichen auf der beiligenden CD-Rom enthalten. Sie sind wesentlich detaillierter als die hier folgenden Kernanforderungen:

6.1.2.1 UC1 Kundensuche durchführen (SearchPartner)

Der Use Case Kundensuche durchführen dient – nach erfolgreicher Authentifizierung – als Einstiegspunkt in den Kundenspiegel. Vom System wird eine Suchmaske angezeigt.
Der Makler füllt die Suchmaske aus und startet die Suche durch das Betätigen des „Suche“ Buttons. Das System ermittelt die Kunden mit den gewünschten Kriterien und zeigt diese in Form einer Liste an.
Der Makler kann eine erneute Kundensuche durchführen oder einen Kunden aus der Kundenliste auswählen. Durch diese Identifizierung gelangt der Makler zum Use Case <UC Kundenübersicht>.

6.1.2.2 UC2 Kundenübersicht einsehen (ShowSummary)

Nachdem der Makler einen Kunden identifiziert hat, ermittelt das System die Verträge des Kunden und zeigt sie in Form einer Liste an. Die Anzeige erfolgt, sofern der Makler eine Zugriffsberechtigung besitzt. Die Liste enthält die zur eindeutigen Identifizierung benötigten Eigenschaften des Vertrages wie z.B. Vertragsnummer, Sparte (Hausrat, KFZ, Krankenversicherung), Kurzinfo (Kennzeichen bei Kfz-Versicherung).
Der Makler gelangt über die entsprechende Auswahl zu den Use Cases Vertragsdetails einsehen, Kundenreport einsehen, Kommunikationsdaten bearbeiten, Schadenliste einsehen, Kundensuche durchführen

6.1.2.3 UC 3 Vertragsdetails bearbeiten (EditContract)

Nachdem der Makler in der Kundenübersicht einen bestimmten Vertrag ausgewählt hat, werden ihm sämtliche Vertragsdetails angezeigt. Der Makler hat die Möglichkeit, die Kontodaten zu einem Vertrag zu ändern und die Schäden zu einem Vertrag einzusehen

6.1.2.4 UC 4 Kundenreport einsehen (ShowReport)

Der Kundenreport dient dem Makler dazu, sämtliche Kundenzahlungen des Jahres gruppiert nach Sparten sowie deren Anzahl und Höhe der Schäden anzuzeigen. Anhand der Schadensdaten kann der Makler Rückschlüsse auf das Schadenverhalten des Kunden schließen.

Der Schadenszeitraum kann begrenzt bzw. erweitert werden. Sofern der Makler den Schadenszeitraum nicht selbst bestimmt, werden fünf Jahre als Standardwert verwendet.

Eine Summenzeile über die verschiedenen Sparten ist anzuzeigen.

6.1.2.5 UC5 Kommunikationsdaten bearbeiten (EditCommunication)

Das System ermittelt die Kommunikationsdaten eines Kunden. Dazu gehören die Adresse des Kunden, seine E-Mail Adressen sowie seine Telefonkontaktdaten

Einer Adresse, E-Mail und Telefonnummer kann eine Art (jeweils privat und geschäftlich, sowie Handy und Fax bei Telefondaten) zugewiesen werden.

Der Makler kann weitere E-Mail Adressen sowie Telefondaten hinzufügen, bereits existierende ändern oder löschen. Die Anschrift eines Kunden darf der Makler nicht ändern.

6.1.3 Beschreibung der Umsetzung

Als Bestandteil der Umsetzung wird jeweils nur ein Beispiel der verschiedenen Arbeitsergebnisse vorgestellt.

6.1.3.1 Use Case Diagramm

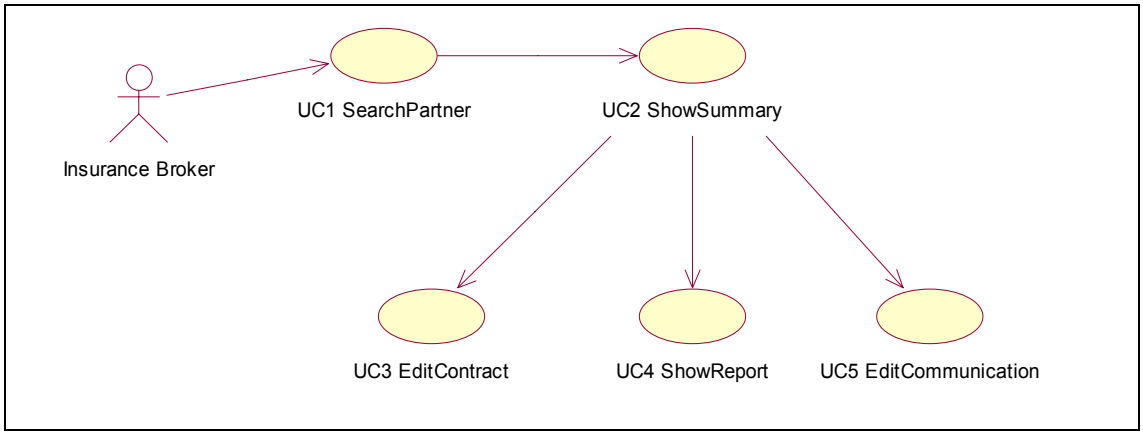


Abbildung 6-1 Use Case Diagramm des Prototypen PartnerSearch

Die Abbildung zeigt die ermittelten Use Cases. Dieses Diagramm dient der Übersicht über die fachlich abgeschlossenen Funktionalitäten. Sie geben noch keine Auskunft darüber, welche weiteren Bestandteile die Use Cases enthalten.

6.1.3.2 Datenmodell

Die Darstellung des Datenmodells erfolgt am Beispiel der Software Erwin. Die folgende Abbildung zeigt die verschiedenen Tabellen sowie deren Verbindungen mit ihren Kardinalitäten. Der schwarz ausgefüllte Kreis signalisiert eine „n“ Beziehung. Ist Keine Angabe vorhanden, gilt die Kardinalität 1. Ein Client (Versicherungskunde) kann also mehrere Kfz-Versicherungen abgeschlossen haben, während eine Kfz-Versicherung immer einem Versicherungskunden zugeordnet ist.

Logisches Datenbankmodell

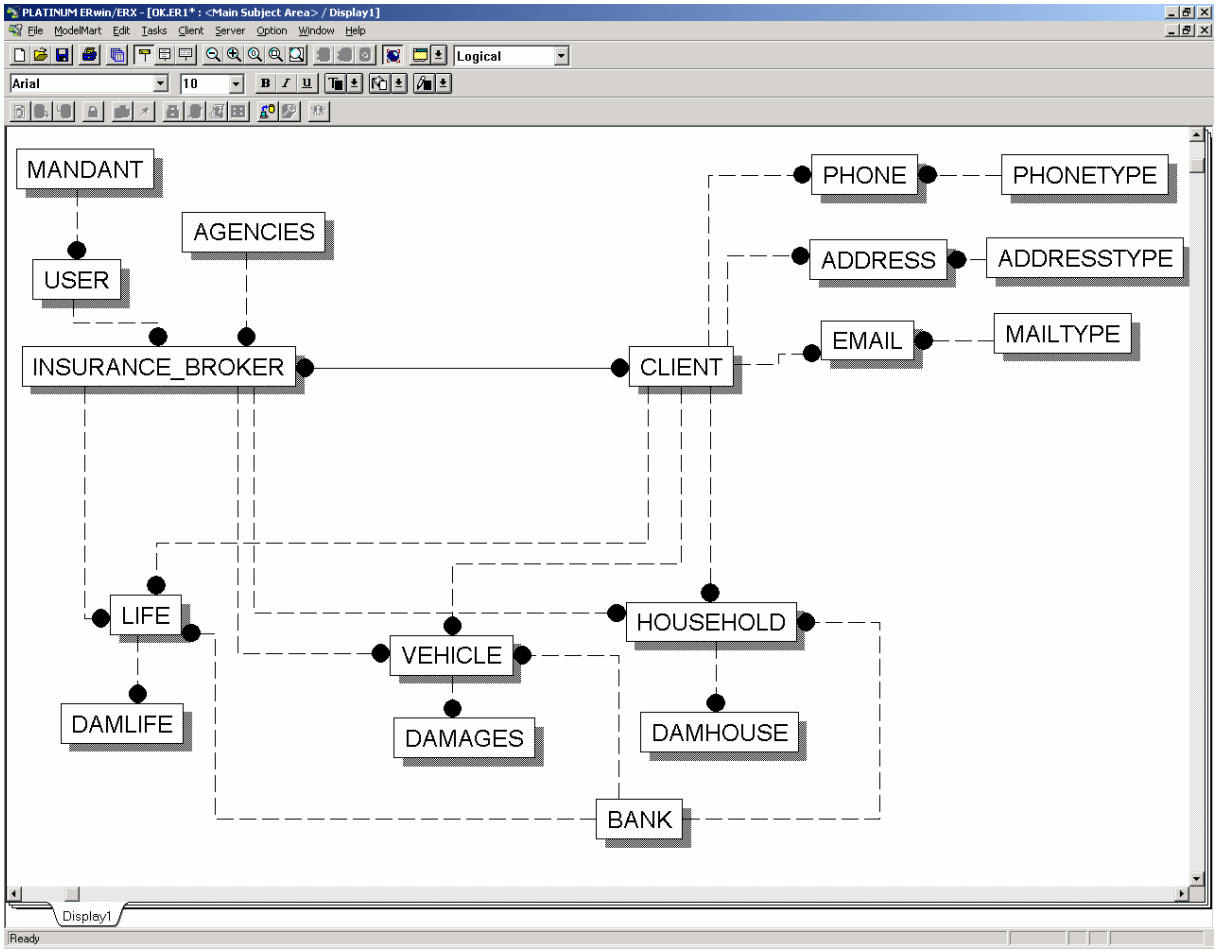


Abbildung 6-2 Physikalisches Datenbankmodell am Beispiel der Software Erwin

Dem linken, oberen Teil des Modells ist der Insurance_Broker (Makler) zu entnehmen. Ihm wird ein User und dem wiederum ein Mandant zugeordnet. Mehrere Makler (ersichtlich durch den schwarzen Punkt) werden einer Agentur zugewiesen.

Dem rechten, oberen Teil wird der Client (Versicherungskunde) beschrieben. Ihm zugeordnet sind mehrere Einträge für Phone (Telefonkontakt), Address (Anschrift) und Email. Diesen Einträgen können jeweils verschiedene Typen zugeordnet werden: z.b. Privat, geschäftlich, bei Telefon kommen Handy, Fax etc. hinzu

Im unteren Teil werden verschiedene Vertragsarten (Life, Vehicle, Household) mit den entsprechenden Tabellen für die Schadensfälle ausgewiesen. Die Bank-Tabelle enthält die Bankleitzahlen und Bezeichnungen deutscher Banken und Sparkassen.

Erwähnenswert ist die Tatsache, dass ein Konto direkt einem Vertrag zugeordnet ist. Eine Kontenverwaltung ist nicht vorgesehen. Diese könnte als Beispiel des iterativ- und inkrementellen Vorgehensmodell von Evolution (und der damit verbundenen Auswirkungen auf Analyse, Design, Erweiterung des DB-Modell, Neue Generierung des Workflows etc.) hinzukommen.

Physikalisches Datenbankmodell

Die Überführung des logischen Datenbankmodells in das physikalische zeigt, dass die Tabelle „Client_Insbroker“ hinzukommt. Sie löst die vorherige n:m Beziehung zwischen Client und Insurance_Broker auf. Ein Client kann seine Verträge bei unterschiedlichen Insurance_Brokern abschließen, während ein Insurance_Broker verschiedene Clienten an das Unternehmen vermittelt. Das ausführliche physikalische Datenbankmodell (incl. Datentypen) kann dem Anhang entnommen werden.

6.1.3.3 Aktivitätsdiagramme

Es werden die verfeinerten Aktivitätsdiagramme – die Ergebnisse des Design-Prozesses (incl. der Zustände) - ausgewiesen.

Startprozess

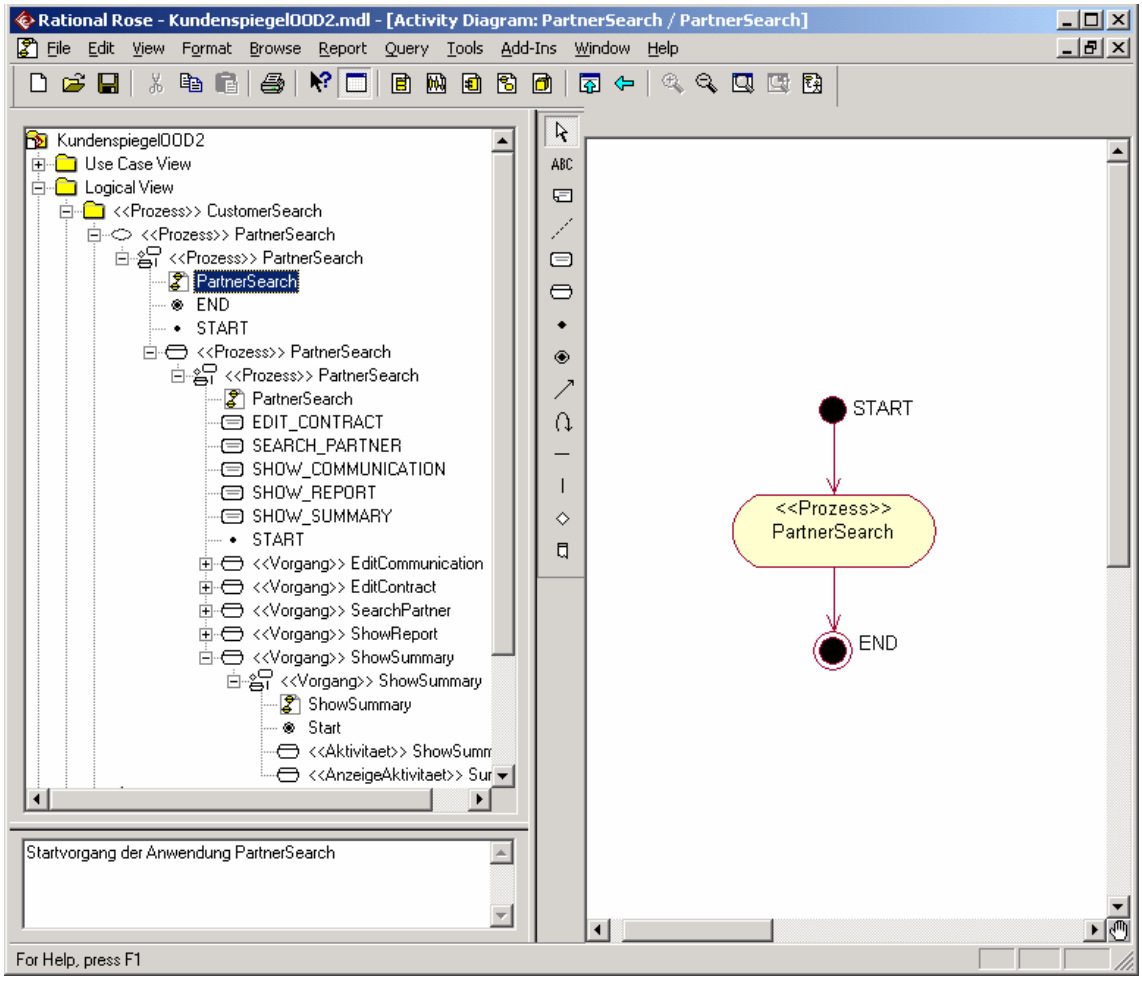


Abbildung 6-3 Aktivitätsdiagramm des Prozesses PartnerSearch

Neben der Startaktivität – einer Aktivität mit dem Stereotypen Prozess – sind dieser Abbildung wichtige Informationen zu entnehmen. Im linken Teil ist die zu erstellende Struktur einer Evolutionanwendung zu entnehmen. Der *Prozess* PartnerSearch besteht aus den verschiedenen *Vorgängen*

EditCommunication, EditContract, SearchPartner und ShowSummary.

Hierbei handelt es sich um die ermittelten Use Cases. Als Steuerungsparameter werden die Zustände

(EDIT_CONTRACT, SEARCH_PARTNER, SHOW_COMMUNICATION, SHOW_REPORT und SHOW_SUMMARY)

verwendet.

Bei den folgenden Aktivitätsdiagrammen ist zu beachten, dass die Zustände, die auf der Vorgangsebene des Prozesses verwendet werden auch in dem tiefer geschachtelten Aktivitätsdiagramm – mit den *Aktivitaeten* eines Vorgangs übereinstimmen.

Vorgangesebene des Prozesses (PartnerSearch)

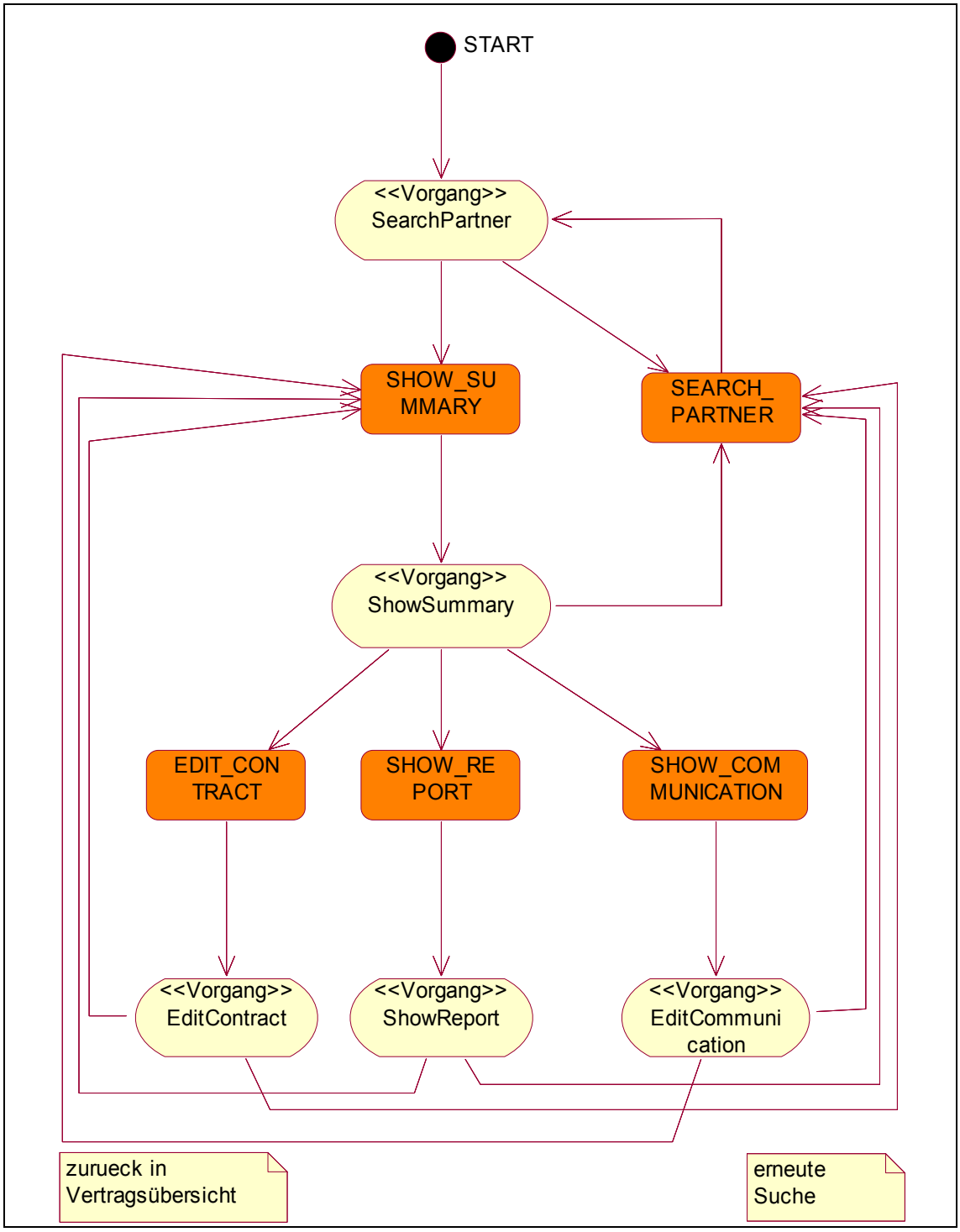


Abbildung 6-4 Vorgänge des Prozesses PartnerSearch

Der Startvorgang führt den Prozess zunächst in den Vorgang SearchPartner.

Aktivitaetsebene eines Vorgangs (SearchPartner)

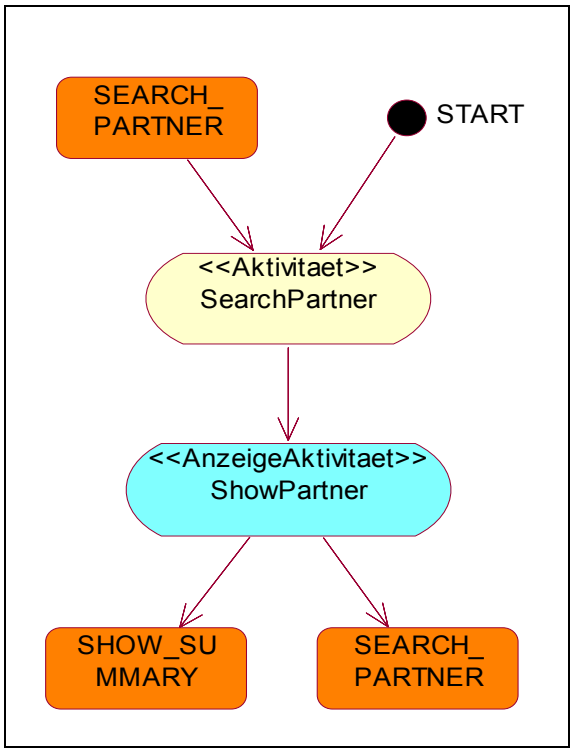


Abbildung 6-5 Aktivitäten des Vorgangs SearchPartner

Der Vorgang SearchPartner besteht nur aus einer Aktivitaet - in ihr wird die Ergebnisliste ermittelt. Bevor die Anzeigeaktivität dafür sorgt, das die ermittelten Daten im Browser angezeigt werden.

Als Eingangsstatus ist der Startzustand der Anwendung - „START“ (er wird beim erstmaligen Aufruf der Anwendung übermittelt) sowie „SEARCH_PARTNER“ vorgesehen.

Nach erfolgter Suche kann ein Anwender eine erneute Suche durchführen, deshalb ist der Eingangszustand „SEARCH_PARTNER“ auch Ausgangsstatus.

Wählt der Anwender in der Ergebnisliste einen Kunden aus, wählt er den Status „SHOW_SUMMARY“ und gelangt zu dem Use Case < UC Kundenübersich einsehen>.

Auf der Vorgangsebene des Prozesses (vgl. Abbildung 6-4) sind die identischen Ein- und Ausgangszustände zu entnehmen

6.1.3.4 Geschäftsobjekte

Als Ausschnitt aus den Geschäftsobjekten zeigt folgende Abbildung die Klassen ClientBO und CommunicationData, Email, Phone und Adress. Diese Klassen können innerhalb der Aktivitaeten verwendet werden. Beispielsweise kann einem Kunden über die Methode clientBO.getCommunications().createEmail() ein Geschäftsobjekt vom Typ „Email“ übergeben werden. Innerhalb dieser createEmail() Methode wird dann die Backendverbindung aufgebaut und die neue E-Mail Adresse in einer Datenbank – oder über MQSeries in einem anderen System – eingetragen. Das Geschäftsobjektmodell ist losgelöst von dem darunterliegenden Datenmodell zu betrachten.

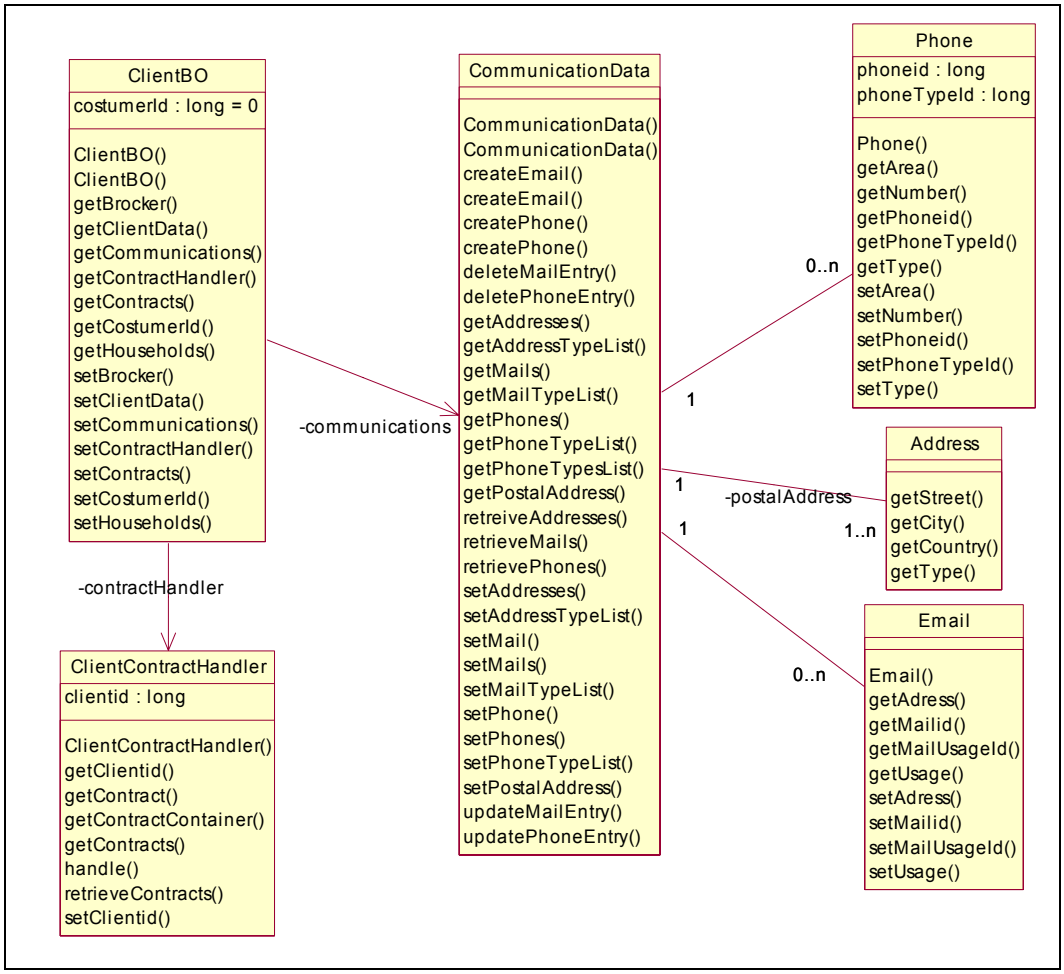


Abbildung 6-6 Beispiel Geschäftsobjekte

6.1.3.5 Anzeigeobjekte

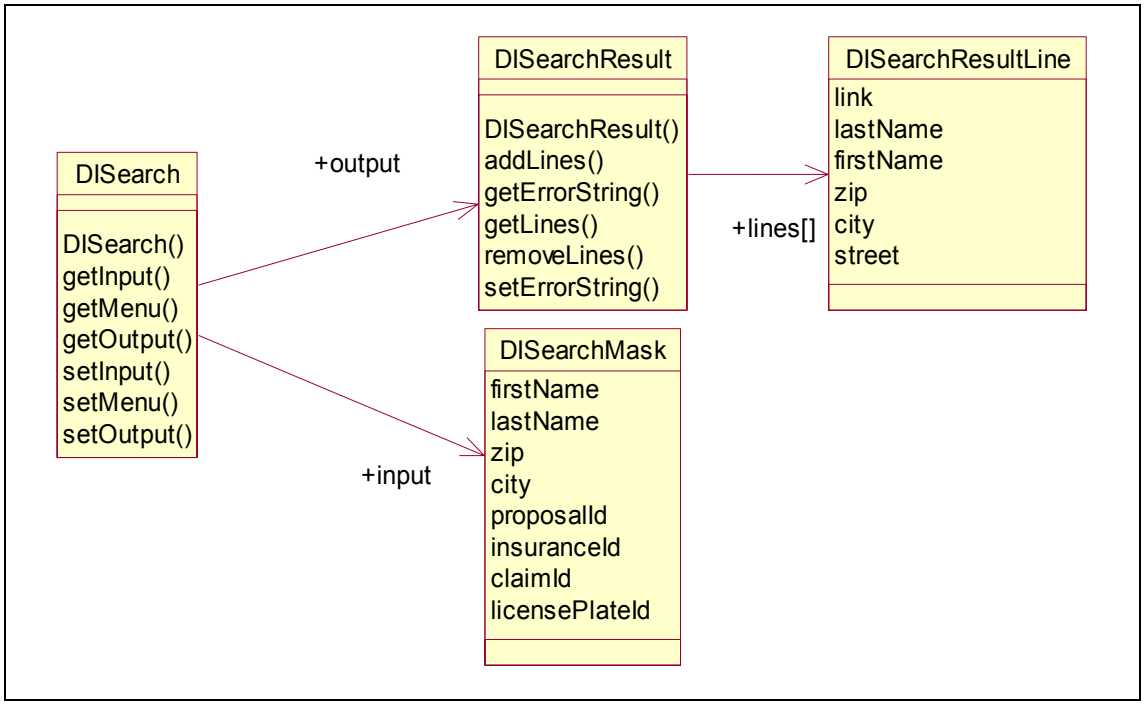


Abbildung 6-7 Beispiel Anzeigeobjekte

Für jeden Vorgang wurde ein „Hauptanzeigeobjekt“ angelegt. Als Beispiel soll das Anzeigeobjekt DISearch des <Use Case 1 Kundensuche durchführen (SearchPartner)> dargestellt werden. Das Anzeigeobjekt „DISearch“ setzt sich aus der Suchmaske „DISearchMask“ (die Suchkriterien, die ein Makler eingeben kann) sowie dem Suchergebnis „DISearchResult“ zusammen.Das Suchergebnis besteht aus Ergebniszeilen „DISearchResultLine“, die wiederum verschiedene Einträge „DISearchResultLine“ besitzen.

Aus Gründen der Übersicht wird auf die Darstellung der getXXX() und setXXX() Methoden der Anzeigeobjekte „DISearchMask“ und „DIResultLine“ sowie auf das Menue-nzeigeobjekt „DIMenue“ verzichtet. Das Menue-Anzeigeobjekt wird in allen XSL-Dateien verwendet.

6.1.3.6 Implementierung der Logik

Die innerhalb der Geschäftsobjekte verwendete Logik ist innerhalb des Workflows in den Aktivitaeten – jeweils in den Methoden performExecute() zu verwenden. Aus den insgesamt 15 Aktivitäten folgt ein stark verkürzter Auszug der Aktivitätsklasse SearchPartner (vgl. UC1 Kundensuche durchführen (SearchPartner), S. 86), die nach erfolgreicher Anmeldung am Prototypen die Kunden eines Maklers ermittelt. Die

wichtigsten Kommentare sind hervorgehoben. Im Anhang ist der gesamte Quellcode der Klasse SearchPartner vorhanden.

```
// Implementierung SearchCustomer innerhalb der performExecute() Methode
protected void performExecute() throws CommandException {


       // user code begin {performExecute USER-BLOCK}
       PartnerSearchSession session = (PartnerSearchSession) getSession();

    DISearchMask inputMask = new DISearchMask();
       // holen derEingabedaten aus dem Anzeigeobjekt
    if (session.getUserInput() instanceof DISearchMask) {
        inputMask = (DISearchMask) session.getUserInput();
    }

    try {
       // neues „Haupt" Anzeigeobjekt erzeugen und Werte überführen
        DISearch search = new DISearch();
        search.setServiceId(getServiceId());

// ... *** Code rausgenommen -                                          ***
//     *** wie die Suche durchgeführt wird, kann dem Quellcode          ***
//     *** auf der beiliegenden CD entnommen werden                     ***

//     *** result ist ebenfalls ein Geschäftsobjekt! - als kein JDBC-ResultSet

            if (!result.isEmpty()) {

                for (Enumeration e = result.elements(); e.hasMoreElements();)
{
                    ClientBO clientBO = (ClientBO) e.nextElement();

//     *** Daten aus den Geschäftsobjekten in Anzeigeobjekte überführen  ***

                    DISearchResultLine line = new DISearchResultLine();
                    line.setLastName(clientBO.getClientData().getSurname());

// ** weitere Rausgenommen ***

                   com.ibm.cis.business.communication.Address postalAddress =
                   clientBO.getCommunications().getPostalAddress();
                    if (postalAddress != null) {
                        line.setZip(postalAddress.getZip());
                        line.setCity(postalAddress.getCity());
                        line.setStreet(postalAddress.getStreet());
                    }
//     ***Code rausgenommen
//     *** erzeugtes Anzeigebojekt (line) in das darüberliegede überführen

                    searchResult.addLines(line);
                } //for end
            } else {
                searchResult.setErrorString("Keine Treffer gefunden!");
            }

        }
//     Ergebnisse der Suche innerhalb des Hauptanzeigeobjektes hinterlegen
        search.setOutput(searchResult);

        displayObject = search;

    } catch (Exception e) {
          e.printStackTrace();
        throw new FatalExecutionException();
    }

       // user code end {performExecute USER-BLOCK}
}
```

6.1.3.7 XSL-Seiten

Die Verwendung von XML, XSL wurde in den Grundlagen beschrieben. Innerhalb des Kapitel 5.5.4.1 wurde gezeigt, wie aus Anzeigeobjekten (Java-Klassen) innerhalb des Frameworks XML erzeugt wird. An dieser Stelle wird ein verkürztes Beispiel zeigen, wie XSL in den Prototpyen verwendet wird. Ein ausführliches Beispiel kann dem Anhang entnommen werden.

Die folgende Regel ist in der Datei „search.xsl“ enhalten. Diese Datei sorgt dafür, dass die Ergebniszeilen der Kundensuche angezeigt werden. Dabei setzt sich eine Ergebniszeile aus einem Link, Nachnamen, Vornamen, PLZ, Ort und Strasse zusammen (vgl. Abbildung 5-10 Entwicklung der Anzeigeobjekte).

```
<xsl:template match="lines" mode="mode22">
<xsl:for-each select=".">
      <tr><td>
            <a><xsl:attribute name="href"><xsl:value-of
select="com.ibm.cis.displayobjects.DISearchResultLine/link"/>
            </xsl:attribute>
            <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISearchResultLine/lastName"/>
            </a>
      </td>
      <td>  <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISearchResultLine/firstName"/>
      </td> <td>
            <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISearchResultLine/zip"/></td>
      <td>
            <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISearchResultLine/city"/></td>
      <td>
            <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISearchResultLine/street"/>
      </td>
      </tr>
</xsl:for-each>
</xsl:template>
```

6.1.3.8 Konfiguration

Wie bereits erwähnt wird die Konfiguration von einer zentralen Datei – configuration.xml – begonnen. Sie verweist auf weitere Konfigurationsdateien wie die im Anschluss folgende internet_customer_search.xml.

```
<!-- configuration.xml -->

<Category NAME="ICIS">
      <Attribute NAME="Version">1.0</Attribute>
      <Category    NAME="internet_customer_search"
                   PATH="internet_customer_search"
                   XML="/internet_customer_search.xml"/>
      <Category    NAME="service_independent"
                   PATH="service_independent"
                   XML="/service_independent.xml"/>
      </Category> …
</Category>
```

```
<!-- datei internet_customer_search.xml für den Mandanten der Webapp-->

<Category NAME="internet_customer_search">
       <Attribute NAME="process">
              com.ibm.cis.processes.PartnerSearch</Attribute>
       <Attribute NAME="evolutionSession">
              com.ibm.cis.sessions.PartnerSearchSession</Attribute>
       <Category NAME="logging" PATH="/../logging" XML="/logging.xml"/>
       <Category NAME="database" PATH="/../database" XML="/database.xml"/>
       <Category NAME="xsl">
              <Attribute NAME="path">c:/kundenspiegel/app/web/</Attribute>
              <Category NAME="com.ibm.cis.displayobjects.DISearch">
                     <Attribute NAME="start">search/search.xsl</Attribute>
              </Category>
              <Category NAME="com.ibm.cis.displayobjects.DISearch">
                     <Attribute NAME="search_partner">search/search.xsl
                     </Attribute>
              </Category>
              …
       </Category>
</Category>
```

Die (gekürzte) Datei „internet_customer_search.xml“ enthält den zu startenden Prozess des Mandanten internet_customer_search sowie die über das Fabrikmuster anzulegende Session. Die Einstellungen für den Datenbankzugriff (user, passwort etc.) sind in der Datei database.xml angegeben.

Innerhalb der Categorie XSL werden neben dem Pfad für die XSL-Seiten (`<Attribute NAME="path">`) den einzelnen Anzeigeobjekten und den Zuständen die zu verwendenden XSL-Seiten zugewiesen.

6.1.3.9 Screenshots

„Ein Bild sagt mehrals 1000 Worte“

Die Screenshots sind im Anhang enthalten (vgl. 8.8, S 139 ff.)

6.2 Erweiterung der Webanwendung im Portalserver

6.2.1 Problemstellung

6.2.1.1 Zielgruppe

Zielgruppe der Anwendung ist auf fachlicher Seite ein Versicherungsunternehmen. Zielgruppe der technischen Umsetzung sind die Architekten des Evolutionframeworks.

6.2.1.2 Überblick über die Problemstellung

Das Evolutionframework soll erweitert werden, um den geschäftsprozessgetriebenen Ansatz in ein Portal zu integrieren. Dazu ist die Webanwendung Kundenspiegel zu erweitern.

6.2.2 Anforderungen

Um Evolution mit dem Portalserver zu integrieren, gelten folgende Anforderungen:

1.) Komplexität

Änderungen des Frameworks sollten minimal bleiben. Umfangreiche Änderungen führen dazu, dass die Komplexität des Frameworks steigt. Erhöhte Komplexität bedeutet, dass Verständnisprobleme bei den Entwicklern (den Anwendern des Frameworks) auftreten können. Diese Probleme können zur Folge haben, dass der Entwicklungszeitraum einer Anwendung ausgeweitet wird.

2.) Wiederverwendung

Die bereits entwickelten projektabhängigen Implementierungen (beispielsweise die erzeugten Geschäftsobjekte) sollen in einer Portallösung wiederverwendet werden.

3.) Wartbarkeit

Neben der Wiederverwendung ist darauf zu achten, dass eine Weiterentwicklung des Frameworks sowie eine Weiterentwicklung der projektabhängigen Klassen bestmöglich von der Portallösung unterstützt wird. Eine Migration des Frameworks, der Geschäftsobjekte sowie der Anzeigeobjekte sollten mit minimalem Änderungsaufwand der Portallösung verbunden sein.

4.) Übernahme der Konfiguration

Die zentrale und flexibele Konfigurationsmöglichkeit ist ein wesentlicher Bestandteil des Frameworks. Sowohl die Konfigurtionsmöglichkeit als auch der Mechanismus, der es erlaubt, die Konfiguration während des Betriebes zu aktualisieren, sollte in der Portallösung anwendbar sein.

5.) Nutzen der zusätzlichen Möglichkeiten der Portlet-API

Wie bereits erläutert bietet die Portlet-API Erweiterungen der Servlet-API. Diese Erweiterungen sollten in dem zu erstellenden Portlet (beispielsweise das Maximieren einer Seite) verwendet werden.

6.2.3 Lösungskonzepte

Grundsätzlich existieren zwei Wege, wie Evolution und die Portallösung integriert werden können.

Wie bereits in dem Kapitel 3.6.2 erläutert, ist ein Portlet bzw. eine Portletapplikation eine Anwendung, die in einem Portalserver läuft. Die Portletapplikation wird in einem PAR-File ausgeliefert.

Hieraus ergibt sich die Möglichkeit, das Framework als Bestandteil der Applikation auszuliefern.

In der zweiten Integrationsmöglichkeit werden Portlet und Framework voneinander getrennt. Dieser Weg sieht vor, dass ein Portlet eine Verbindung zum Framework aufbaut. Das Framework steuert dann den Workflow und generiert den Output[90]. Dieser Output wird an das Portlet übergeben, welches dann wiederum dann für die Anzeige der Daten im Portal sorgt. Folgend werden insgesamt 5 Lösungsvarianten den beiden Integrationsmöglichkeiten zugeordnet.

[90] Als Output ist HTML, XML, Java Byte Stream denkbar

6.2.3.1 Framework wird Bestandteil des Portlets

1. Variante: *Mappen des PortletRequest auf HttpRequest*

   Die Portlet-API bietet die Möglichkeit, einen PortletRequest auf einen HttpRequest zu mappen. Das bedeutet, dass nach dem Mappen die Methoden gemäß der Servlet-API für das Request-Objekt vorhanden sind. Das Portlet übernimmt somit die Steuerungslogik und sorgt für die Abarbeitung des Prozesses. Damit löst das Portlet das Steuerungsservlet des Frameworks ab.

2. Variante: *Erweiterung des Frameworks gemäß Portlet-API*

   Anhand eines gekapselten Objektes innerhalb der HTTP-Session sorgt das Framework für den Datenaustausch innerhalb eines Prozessdurchlaufs bzw. über mehrere Anfragen hinweg. Das Framework verwendet Methodenzugriffe der HTTP-Session und des HttpRequest, die in den Klassen der Portlet-API nicht vorhanden sind. Diese Methodenzugriffe müssen in dieser Variante angepasst bzw. beseitigt werden.

6.2.3.2 Verbindungsaufbau vom Portlet zum Framework

3. Variante: *Nutzen der Generetic Portlets*

   Der Portalserver wird zusammen mit den so genannten Generetic Portlets ausgeliefert. Diese erlauben es, Portlets zu erstellen, ohne Java-Code implementieren zu müssen. Als Teil dieser Portlets erlaubt das Servlet-Invoker-Portlet bzw. der ContentAccessService das Aufrufen von Servlets außerhalb des Portalservers.

4. Variante: *Nutzen des I-Frame Portlets*

   Das I-Frame Portlet nutzt den I-Frame Mechanismus der HTML und erlaubt das Anzeigen einer dem Portalserver fremden URL. Ein Nutzen der Portlet-API entfällt, da das die Kommunikation innerhalb des I-Frames ohne Portalserver stattfindet

5. Variante: *Portlet nutzt Framework unter Verwendung eines neuen Mandanten*

   Das Portlet baut eine Verbindung zum Portlet auf. Java bietet mit der abstrakten Klasse HttpURLConnection die Möglichkeit diese Verbindung aufzubauen und HTTP spezifische Besonderheiten zu nutzen. (vgl.

6.2.4 Bewertung der Lösungskonzepte

Zur Ermittlung der zu implementierenden Variante werden den Anforderungen folgende Ausprägungen zugewiesen:

	Anforderungen				
Aus-prägung	Komplexität	Wiederver-wendung	Wartbarkeit	Konfiguration	Nutzen der Portlet-API
5	gering	hoch	gut	voll	vollständig
3	mittel	mittel	mittel	mittel	mittel
1	hoch	gering	aufwendig	keine	keine

Tabelle 6 Bedeutung der Ausprägungen

Die folgende Tabelle zeigt die Zuordnung der Ausprägungen zu den Anforderungen je Variante. Eine Erläuterung ist im Anhang enthalten.

Variante	Komplexität	Wiederver-wendung	Wartbarkeit	Konfiguration	Nutzen der Portlet-API
1	5	5	1	1	1
2	1	3	1	1	5
3	5	5	5	5	1
4	5	5	5	5	1
5	3	5	5	5	5

Tabelle 7 ungewichtete Bewertung der Lösungskonzepte

Die Anforderungen werden unterschiedlich stark bewertet, dazu werden folgende Faktoren fesgelegt:

	Komplexität	Wiederver-wendung	Wartbarkeit	Konfiguration	Nutzen der Portlet-API
Faktor	2	1	3	1	3

Tabelle 8 Gewichtung der Anforderungen

Gewichtet ergibt sich daraus folgende Tabelle:

Tabelle 9 gewichtete Bewertung der Lösungskonzepte

Variante	Komplexität	Wiederver-wendung	Wartbarkeit	Konfiguration	Nutzen der Portlet-API
1	10	5	3	1	3
2	2	3	3	1	15
3	10	5	15	5	3
4	10	5	15	5	3
5	6	5	15	5	15

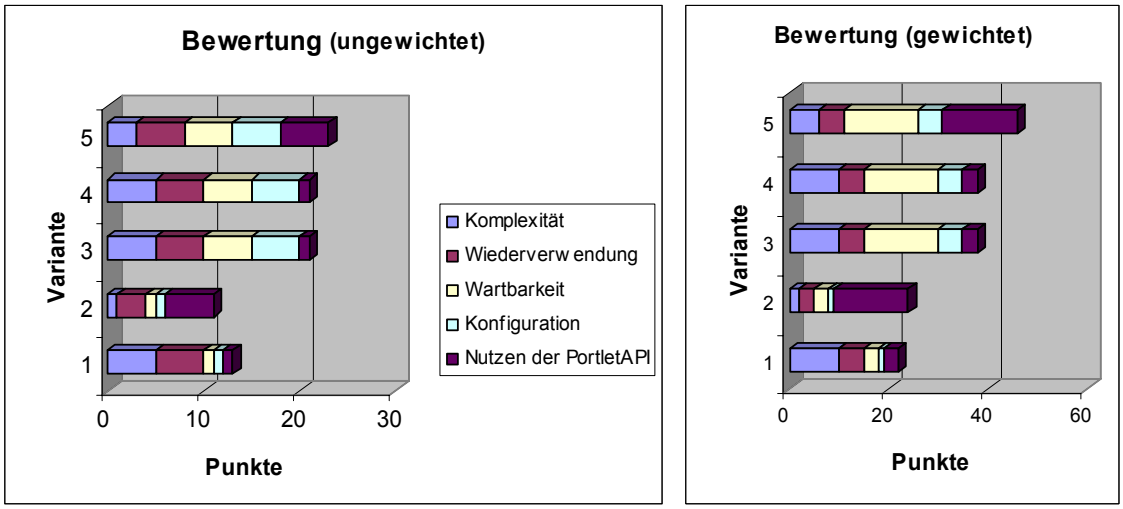


Abbildung 6-8 Bewertung der Lösungsvarianten

Aus der obigen Abbildung zu entnehmen ist :

Variante 5 ist zu implementieren.

6.2.5 Beschreibung der Umsetzung

6.2.5.1 Ausweitung der bisherigen Anwendung

Die lauffähige Kundenspiegelanwendung wurde um den Mandanten „internet_customer_search_wps“ erweitert. Dieser Mandant entspricht dem „internet_customer_search“, greift jedoch über das Portal auf die Anwendung zu. Im Einzelnen werden folgende Änderungen beschrieben:

- Authentifizierung und Autorisierung
- Portlet-Entwicklung
- Anpassen des Frameworks
- Anpassen der XSL-Dateien
- Konfiguration

6.2.5.2 Authentifizierung und Autorisierung

Um den Zugang zum Kundenspiegel zu gewähren, wurde im LADP eine Gruppe Kundenspiegel angelegt worden. Dem Portaladministrator obliegt es, die verschiedenen Benutzer der Gruppe Kundenspiegel zuzulassen.

Zur Autorisierung werden die Zugangsdaten des Portalanwenders verwendet

6.2.5.3 Portlet-Entwicklung

Die Beschreibung des Portlets wird anhand der darin verwendeten Methoden init() und service() erläutert. Der Quellcode des Portlets ist im Anhang enthalten.

init()

in der init-Methode wird aus dem Portletimplementierunsdeskriptor (portlet.xml) die URL ausgelesen. Sie bestimmt den Rechner, auf dem die Evolution-Anwendung installiert ist. Der Portaladministrator kann diese URL über die Konfigurationsmöglichkeit eines Portlets ändern.

service()

In dieser Methode wird die Verbindung zwischen Portlet und dem EvolutionServlet aufgebaut.

Zur Kommunikation übermittelt das Portlet die Parameter über zwei verschiedene Arten.

1. Parameterübergabe durch Anhängen an die URL
   Wie bereits erwähnt, ist die Portlet-API nicht identisch mit der Servlet-API. Um Den Datenaustausch zwischen Portal und dem EvolutionServlet zu gewährleisten, werden die Portlet-Parameter „servletkonform“ in der HashTable „parameter“ abgelegt. Diese HashTable enthält die Daten, die der Anwender in Formularen eingegeben hat.Diese darin enthaltenen Parameter werden mit jedem Verbindungsaufbau übermittelt.
2. Parameterübergabe per HTTP-Header
   Bevor der Evolution-Prozessablauf gestartet werden kann, wird der dafür zuständigen HttpURLConnection noch um folgende HttpHeader Informationen ergänzt:

   cookie

   Anhand des „cookie“ Header-Parameter des HTTP-Protokolls erkennt ein Application Server, dass bereits eine Anfrage des Client vorhanden war. Dazu wird vom Applikationserver nach erstmaligen Aufruf einer Anwendung bzw. Servlets der Parameter „set-cookie“ über das HTTP-Protokoll mitgeliefert. Beim nächsten Verbindungsaufbau wird der Inhalt dieses Parameters als „cookie“

Header-Parameter mitgeliefert. Somit kann der Applctionserver die Session – und somit die gespeicherten Daten - einem Client wieder zuordnen.

user-agent

Der HTTP Header-Parameter (vgl. [I.12]) „user-agent“dient dazu, einem Applicationserver bzw. einem Webserver Auskunft darüber zu geben, von welcher Software (normalerweise von welchem Browser, z.B. Netscape) eine Anfrage an den Server gerichtet wurde.
Innerhalb des Portlets wird dem „user-agent“ der Wert „WPS2.1“ zugewiesen. Somit wird dem Framework ermöglicht, die spezielle Portalserver Aufbreitung auszuführen.

wps_ImagePath

Der WPS speichert ein Par-file (also die Porteltapplikation) unter Verwendung eines Zeitstempels. Mit jedem Update der Applikation wird ein neuer Zeitspempel erzeugt. Der wpsImagePath Parameter wird dem Framework übergeben, damit die Grafiken – als Bestandteil des Portlets – lokalisiert und somit angezeigt werden.

wps_windowState

Die Verwendung der Erweiterungen der Portlet-API wird anhand des Parameters wps_windowState realisiert. Diesem Parameter werden die Werte „max“ oder „normal“ zugewiesen. Sie dienen dazu, unterschiedliche XSL-Dateien und somit unterschiedliche Transformationsregeln anzuwenden. Enthält der wps_windowState den Wert „max“[91], werden alle Informationen angezeigt. Enthält er den Wert „default“ wird eine andere XSL-Datei – mit entsprechend weniger Regeln, bzw. HTML-Code - verwendet.

Im Gegensatz zu den Parametern, die an die URL angehängt werden, enthalten die HTTP-Header Daten keine Daten, die ein Portalanwender beeinflussen kann. (wie z.B. durch das Füllen von Formularfeldern)

[91] der Wert „max“ wird gesetzt, wenn das Portlet maximiert wird, d.h. wenn nur noch das eine Portlet auf der Portalseite sichtbar ist.

6.2.5.4 Anpassen des Framework

Das Framework wurde an folgenden Stellen angepasst

1.) Anzeigeobjekte
2.) Steuerungsservlet

Anpassen der Anzeigeobjekte

Damit der zuvor beschriebene Parameter „wps_imagePath“ in die XML-Struktur – die für die Anzeige der Daten notwendig ist - überführt wird, wurde die Basisklasse der Anzeigeobjekte (com.ibm.evolution.Displayobjekt) um diese Parameter erweitert.
Dieser Parameter wird nur in der XML-Struktur abgebildet, sofern er in den Anzeigeobjekten mit einem Wert gefüllt ist[92].

Anpassen des Steuerungsservlet

Das Steuerungsservlet wurde an verschiedenen Stellen erweitert:

1. Bevor das Hauptanzeigeobjekt in die XML-Struktur überführt wird, werden die Parameter wps_ImagePath den Anzeigeobjekten zugewiesen.
2. Vor der XSLT-Transformation wird bei einem Request vom Portalserver der wpsWindowState weitergeleitet.
3. Falls ein Fehler während des Workflows stattfindet, wird bei der Erzeugung der Fehlerseite (Sie ist Bestandteil des Steuerungsservlets) der HTTP-Responsecode auf `SC_CONFLICT (vgl.` [I.5]) gesetzt. Durch diesen Mechanismus, wird dem aufrufenden Portlet der Fehlerfall mitgeteilt.
   Das Portlet zeigt die Ausgabe des Frameworks nur bei erfolgreicher Verarbeitung der Anfrage an. Nach einem Fehlerfall wird im Portlet der Cookie-Parameter innerhalb der PortletSession zurückgesetzt. Somit startet die darauf folgende Anfrage mit dem Startvorgang des Prozesses.

Der Quellcode ist im Anhang enthalten (vgl. S. 148).

[92] Bei der XML-Struktur des Mandanten internet_customer_search wird dieses Elemente also nicht genriert.

6.2.5.5 Anpassen der XSL-Dateien

Basis für die XSL-Seiten – also die Ausgabeaufbereitung - sind die zuvor entwickelten des internet_customer_search Mandanten.

Es wurden folgende Änderungen vorgenommen:

- Kürzen der HTML-Ausgabe
- Hinzufügen Portalspezifischer Aufbereitung

Kürzen der HTML-Ausgabe

Innerhalb einer Portalseite können mehrere Portletapplikationen enthalten sein. Das hat zur Folge, dass die HTML-Ausgabe eines Portlets geändert werden musste. Die XSL-Seiten, die für die Aufbereitung des internet_customer_search_wps Mandanten zuständig sind erzeugen HTML-Code, der von einem Table-Tag umschlossen wird. (<TABLE> …. </TABLE>).

Der Rahmen um diesen Code (<html><head></head><body> … <TABLE></TABLE> …</body><html> wird nicht mehr generiert.

Hinzufügen portalspezifischer Aufbereitung

Die portalspezifische Aufbereitung fand wie folgt statt:

1.) Zur einheitlichen Darstellung des Portals wurden die Cascading Stylesheets des Portalservers (vgl. [I.35]) verwendet. Somit erscheinen die Portlets in einheitlichem Layout
2.) Der zuvor festgelegten Parameter wpsImapgePath wird zu Beginn der XSL-Dateien des Portalserver mandanten ausgelesen.
3.) Jeder Grafik ist der entsprechende wpsImagePath voranzustellen:

Beispielcode:

```
zu 1.)
Verwendung der Cascading Stylesheets des WPS:
       <tr>m
           <th class="wpsTableHead">PLZ/Ort</th>
           <th class="wpsTableHead">KFZ-Kennzeichen</th>
           <th class="wpsTableHead">Schadennummer</th>
       </tr>

zu 2.)
Auslesen des wpsImagePath zu Beginn des Stylesheet

<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:param name="imagepath" select="//wpsImagePath"/>
..
zu 3.)

der Parameter ist jeder Grafik wie folgt voranzustellen:
```

```
<xsl:template match="com.ibm.cis.displayobjects.DIMenuItem"
mode="mode2">
      <input type="image" name="MenuItem">
            <xsl:attribute name="src">
                  <xsl:value-of select="$imagepath" />
      <xsl:value-of select="label"/>
            …
      </input>
</xsl:template>
```

6.2.5.6 Konfiguration

Die Konfiguration des Prototypen im Portalserver wird an zwei Stellen vorgenommen:

1.) Konfiguration über das Framework
2.) Konfiguration des Portlets - über die Datei Portlet.xml

Konfiguration über das Framework

Die Konfiugration wurde um den Mandanten „internet_customer_search_wps“ erweitert.

Dieser nutzt im Wesentlichen die Klassen des Mandtanten „internet_customer_search“. Ihm werden allerdings andere XSL-Dateien zugewiesen. Innerhalb der Verzeichnisstruktur der XSL-Dateien wurden zwei Unterverzeichnisse eingefügt:

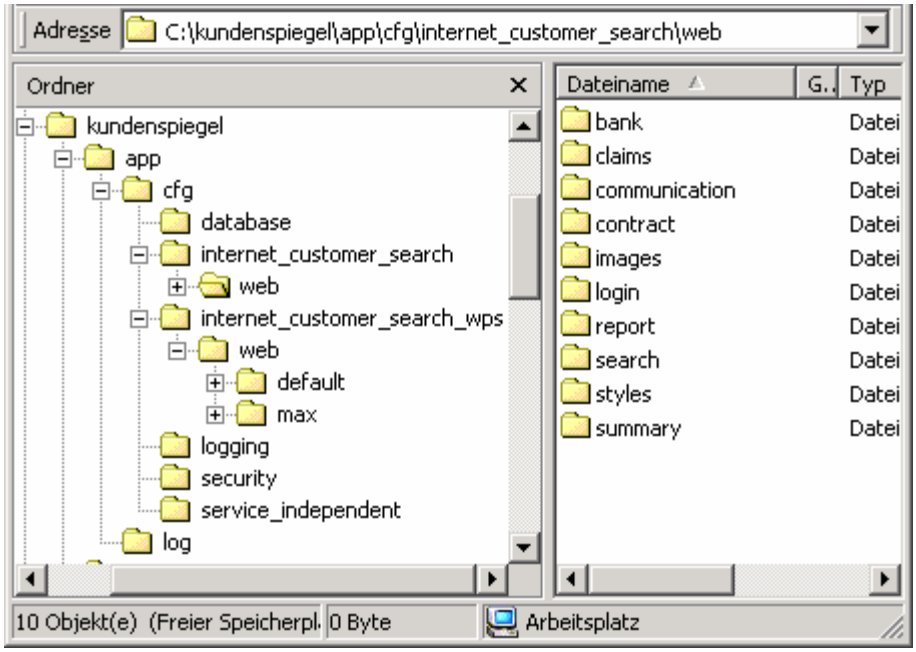


Abbildung 6-9 Erweiterte Konfiguration

Der obigen Abbildung sind die verschiedenen Mantanten „internet_customer_search“ sowie „internet_customer_search_wps“ zu entnehmen. Beide Mandanten sind in der Datei „configuration.xml“ eingetragen, die im cfg-Verzeichnis abgelegt ist. Die Einstellungen für die Datenbank, logging, security sind bei beiden Mandanten identisch.

Die XSL-Dateien werden jeweils unterhalb des „web“-Verzeichnis vorgenommen. Aus Gründen der Übersicht, wurden die XSL-Dateien den Unterverzeichnissen (rechter Teil der Abbildung) zugeordnet.

Der Verzeichnisstruktur des „wps“-Mandanten können die Unterverzeichnisse „default“ und „max“ entnommen werden. Unterhalb dieser Verzeichnisse sind jeweils die XSL-Dateien für den entsprechenden Modus „normal“ und „maximized“ hinterlegt. Die Abbildung Anhang B-0-10 zeigt die Verwendung der unterschiedlichen XSL-Dateien sowie den Portlet-Modus „maximized“ und „normal“.

Konfiguration des Portlets

Das erstellte Portelt wird über die Datei Portlet.xml konfiguriert.Der Portaladministrator legt diese in der Datei die Steuerungsparameter von Evolution (nameDO[93], actionName, sowie Mandant) fest und bestimmt die URL fest, unter der das Steuerungsservlet des Frameworks erreichbar ist.

Neben dem erstellten Portlet sind die Grafiken Bestandteil des PAR-Files des Portlets.

[93] nameDO = nameDisplayObject

7 Bewertung und Ausblicke

7.1 Evolution

Diese Vorgehensweise zur Entwicklung einer Webapplikationen hat sich in folgenden Punkten als besonders hilfreich erwiesen:

- Die Ablauflogik ist visualisiert.
  Das visuelle Prozessmodell repräsentiert die Ablauflogik der Web-Applikation. Die Ablauflogik muss damit nicht manuell in Form der XSL bzw. HTML Seiten eingebunden werden. Das Prozessmodell ersetzt damit das klassische Prozessmodell.
- Dokumentation
  Da das Prozessmodell grafisch visualisiert und gleichzeitig Web-Applikation ist, hat man jederzeit eine exakte und verständliche Dokumentation der Web-Applikation – das Prozessmodell.
- Wartung
  Dies ist der eigentliche Hauptvorteil dieser Lösung. Ein grafisches Prozessmodell ist kann viel schneller verstanden und modifiziert werden als bei der „Durchforstung“ von Quellcode. Es existiert immer ein Prozessmodell, das den Workflow enthält. Als Besonderheit der Entwicklungsumgebungen Visual Age for Java kommt hinzu, dass nach einer Neugenerierung des Workflows die bereits implementierten Klassen und Methoden nicht überschrieben werden.

Entwickler sollten dank Aktivitätsdiagramm schnell den Worfklow einer „Evolutionanwendung“ verstehen. Mit der Überführung des Workflows in Java-Code ist sehr gut lesbarer Quelltext vorhanden. Die zu wartenden Stellen können schnell ausgemacht werden. Welchen Quelltext jedoch ein einzelner Entwickler innerhalb der dafür vorgesehenen Stellen implementiert, kann Evolution nicht kontrollieren[94].
Ein Problem löst Evolution jedoch nicht: Es kann immer nur ein Prozess „gleichzeitig“ also nicht „quasi-parallel“ bearbeitet werden.

[94] vielleicht ist es per Tool-API von Visual Age möglich, die Entwicklungsumgebung zu beenden, wenn ein goto im Quelltext erscheint ;-).

Angenommen ein Makler benutzt den in Rahmen dieser Diplomarbeit erstellten Kundenspiegel um Vertragsdetails einzusehen. Dazu meldet er sich an, führt die Kundensuche durch und wählt den entsprechenden Vertrag aus.
Meldet sich zu dieser Zeit ein anderer Kunde telefonisch und bittet um Auskunft die Änderung seiner Bankdaten vorgenommen wurde, wird die Bearbeitung des Prozesses durch die erneute Kundensuche des Telefonkunden beendet. Die bereits erledigten Schritte (Kundensuche, Auswahl des Vertrages) müssen nach Beendigung des Telefonates erneut durchgeführt werden.
In diesem Beispiel sind die Folgen noch gering. Bei unfangreicheren Prozessen - beispielsweise das Anlegen eines Vertrages: zahlreiche Eingaben sind vorzunehmen - sind die Auswirkungen deutlich höher.

Durch den Austausch der Daten über die Session wird in dieser Version des Frameworks nicht gewährleistet, dass ein unterbrochener Prozess später fortgesetzt werden kann. Hierzu wäre es notwendig, einen Prozess sowie die Daten eines Prozesses innerhalb der Session eindeutig zu kapseln.

7.2 Prototypen im Portal

7.2.1 Bewertung anhand der Anforderungen

Die Bewertung des implementierten Prototypen erfolgt anhand der zuvor ermittelten Anforderungen:

1.) *Komplexität*✓

Die Änderungen des Frameworks sind minimal. Innerhalb des Frameworks wurden nicht mehr als 30 Zeilen hinzugefügt. Die Stellen, an denen eine portalspezifische Bearbeitung stattfindet, sind einfach zu erkennen. Es wird jeweils der Http-Header-Parameter „user-agent“ auf „WPS2.1“ abgefragt.

2.) *Wiederverwendung*✓

Die bereits entwickelten Klassen des vorherigen Mandanten (internet_customer_search) wurden in der Portallösung wieder verwendet.

3.) *Wartbarkeit*✓✓

Die besondere Stärke dieser Lösung ist, das eine Erweiterung des Frameworks (z.B. eine neue Funktionalität, Anbindung weiterer Backendsysteme, Verwendung schneller Parser etc.) umgehend innerhalb der Portallösung verwendet wird. Es existiert nur ein Framework und nicht eins für die Portallösung und eins für die „normalen" Webanwendungen. Dementsprechend muß das Framework auch nur an einer Stelle gewartet werden.

Das erstellte Portlet muss zur Nutzung der hinzukommenden Funktionalitäten weder kompiliert noch im Portalserver aktualisiert werden. Das Portlet muss lediglich angepasst werden, wenn Steuerungsparameter oder Grafiken geändert werden bzw. hinzukommen.

Das beste Beispiel der Wartbarkeit ist der Prototyp selbst. Evolutionbasierte Anwendungen können mit dieser Frameworkversion mit geringem Aufwand – lediglich die Anpassung der Konfiguration sowie der XSL-Dateien ist noch notwendig - in das Portal integriert werden.

4.) *Übernahme der Konfiguration*✓

Die Konfiguration findet – bis auf die Steuerungsparameter des Portlets - zentral innerhalb der Frameworkkonfiguration statt. Über diese Konfiguration können Änderungen am Portlet-Layout[95] vorgenommen werden, ohne das das Pportlet dazu neu kompiliert und deployed werden muss.

5.) Nutzen der zusätzlichen Möglichkeiten der Portlet-API✓

Innerhalb dieses Prototypen werden der Konfigurationsmechanismus sowie die verschiedenen Portlet-Modi (normal, maximized – vgl. S. 40) genutzt.

Die entwickelte Lösung unterstützt keine Eventverarbeitung. Dazu müssen alle Übergabeparameter mit einer Kennung - die ein Portlet eindeutig im Portalserver kennzeichnet - versehen werden. Diese Kennung vergibt der Portalserver während der Installation eines Portlets. Sie bleibt im Gegensatz zu dem eingeführten Parameter „wpsImagePath" auch nach einem Update-Vorgang des Portlets konstant.

Beim Versuch die Ereignisverarbeitung im Prototypen zu verwenden enstand das Problem, das der „codierte" Parameter „actionName" - er hat etwa folgenden Aufbau „PC_49_actionName" – asl Steuerungsparameter des Webshpere Portal Server erkannt

[95] Über die Anpassung der XSL-Dateien

wrude, d.h. der Portalserver ordnet diesen Parameter nicht mehr Evolution zu, sondern wird zur Portalsteuerung verwendet. Somit ist eine Eventverarbeitung nicht möglich. Zur Eventverarbeitung müste dieser Steuerungs-parameter in Evolution umbenannt werden.

Als Beispiel für die Eventverarbeitung ist durchaus vorstellbar, dass alle evolutionbasierten Anwendungen innerhalb eines Portlets als Navigation zusammengestellt werden – z.B. ein Link zum Kundenspiegel, ein Link zur Schadensmeldung etc.).

Der daraus erzeugte Event könnte in einem zweiten Portlet – einer Erweiterung des in dieser Diplomarbeit erstellten - aufgefangen werden. Dieses Portlet zeigt dann die gewünschte Applikation - je nach Auswahl - an.

Somit könnten zwei Portlets die Basis für zahlreiche Eigenentwicklungen eines Kunden sein[96].

7.2.2 Bewertung der alternativen Lösungsvorschläge

Die alternativen Lösungsvarianten wurden im Rahmen der Diplomarbeit ebenfalls auf Realisierbarkeit getestet. Die Ergebnisse werden wie folgt kurz vorgestellt.

Variante 1)

Diese Variante führte innerhalb des Workflows in der Aktivität PartnerSearch zu einem „unautorisierter Sessionzugriffsfehler“. Die Ursache dieser Fehlermeldung konnte nicht gefunden werden.

Variante 2)

Innerhalb der Frameworks wir die HTTP-Session in 43 verschiedenen Klassen verwendet. Da die PortletSession nicht der HTTP-Session entspricht müssten diese Klassen der PortletSession angepasst werden. Aufgrund der Anforderung Komplexität wurde diese Variante nicht umgesetzt.

Variante 3)

Der ContentAccessService sowie das ServletInvoker-Portlet können lediglich zum einmaligen Aufruf einses Servlets verwendet werden. Der Sessionmechanismus der Servlet-API wird von den beiden Implementierungen nicht unterstützt. Das hat bei dem

[96] Dazu bedarf es jedoch einer Festlegung, welche Eigenentwicklungen ein User nutzen darf. Diese Aufgabe kann das Administrationsportlet nicht leisten.

erstellten Prototyp zur Folge, dass lediglich die erste Aktivität des Workflows ausführt wird d.h. es wird lediglich die leere Suchmaske angezeigt wird. Bereits die Ergebnisliste der Suche wird nicht angezeigt.

Variante 4)
Die Verbindung zwischen IFrame-Portlet und dem Prototypen konnte innerhalb weniger Minuten hergestellt werden. Diese Version unterstützt jedoch weder die Single Sign On Funktionalität noch die Portlet-API
Variante 5)
Neben der als Lösung vorgestellten Version wurde ein weiteres Portlet entwickelt. Der wesentliche Unterschied ist, das die Transformation nach HTML innerhalb des Portalservers stattfindet. Das Framework liefert also den XML-Baum an das Portlet zurück. Aus Gründen der Wartbarkeit[97] sowie der eintheitlichen Konfiguration über das Framework wurde jedoch die vorgestellte Lösung vorgezogen.

7.3 Portal

Diese Diplomarbeit hat einige Beispiele aufgezeigt wie Portale eingesetzt werden können. Die Mächtigkeit einer Portallösung läst sich jedoch mit einem einzelnen Prototypen darstellen, dazu müssen verschiedene Anwendungen in ein Portal integriert werden. Mit dieser Diplomarbeit sind jedoch die Grundlagen gelegt, um „Evolution“ basierte Anwendungen nicht nur als losgelöste Anwendung, sondern als Teil des Portals zu verwenden.

Für das Projektgeschäft erweisen sich die Portalfunktionen wie Benutzerverwaltung und Authorisierung als besonders hilfreich. Diese Funktionalitäten müssen somit nicht in jeder Anwendung neu entwickelt werden.
Während dieser Diplomarbeit ergaben sich folgende Probleme bei der Entwicklung von Portlets:

Deployment
Bei jeder Aktualisierung – auch bei der Entwicklung - eines Portlets sind viele Schritte notwendig, um ein Portlet erneut ausführen zu können.

1.) Packen des Portlets in ein PAR-File

[97] Änderungen des HTML-Generators wirken sich automatisch auf die Transformation aus.

2.) Auswhählen der Portletadministrationsseite
3.) Auswählen des Portlets und betätigen des update Vorganges
4.) Bestätigen des Updates
5.) Wechseln auf die Seite, die das Portlet enthält

Diese Schritte sind sehr zeitaufwendig (siehe unten).

Fehlende Debugmöglichkeit

Die Portlet-API bietet die Möglichkeit zur Laufzeit Informationen eines Portlets in eine Protokolldatei zu schreiben. Diese Möglichkeit ist im Vergleich zur Debugmöglichkeit eines Servlets unzureichend. VisualAge for Java ermöglicht beispielsweise das schrittweise Verfolgen des Quelltextes und Auslesen von Objekten zur Laufzeit einer Anfrage. Diese Schritte dienen einer kürzeren Entwicklungszeit, da die Problemstellen im Quelltext schneller gefunden werden können.

Eventverarbeitung

Die Eventverarbeitung ist im der verwendeten Portlet-API nicht ausgereift. Vielleicht ist das auf das Fehlen einer Standard-API zurückzuführen. Um ein Event zu verarbeiten muss es zunächst durch einen User ausgelöst werden – durch Betätigen eines Buttons, z.B „Suche“. Der Actionlistener „fängt“ das Event und wirft Innerhalb der Implementierung ein Messageevent. Dieses Übermittelt die Daten an die Messagelistener der weiteren Portlets.

An dieser Stelle ergibt sich das Problem, dass die Daten, die beim Messagelistner ankommen, nicht direkt an das entsprechende Portlet übergeben werden können. Sie müssen entweder in der PortletSession oder in einem Objekt der Klasse PortletData hinterlegt werden.

Im Gegensatz zum einfachen Durchreichen der Daten belegen sie in der PortletSession oder PortletData zusätzliche Ressourcen.

Performance

Für die Entwicklung der Prototypen wurde ein aktueller Rechner (P IV 1800, 1 GB Arbeitsspeicher) zur Verfügung gestellt. Trotz des verwendeten Arbeitsspeichers dauerte der alleinige Updatevorgang – also nach betätigen des update-Buttons - eines einzelnen Portlet bis zu drei Minuten.

Zu beachten ist, dass nur die benötigte Software (vgl S. 5) auf diesem Rechner und keine weiteren (evtl. ressourcenverbrauenden) Portlets installiert waren. Im LDAP wurden nur zwei Benutzerkennungen angelegt.

Diese beiden ersten Probleme der Entwicklung werden in der künftigen Version des Portalservers behoben sein. Für den WebSphereApplication Developer und den Portalserver in der Version 4.1 wird ein Portal-Toolkit angeboten.

7.4 EAI

Neben anderen Softwareherstellern hat auch IBM erkannt, dass Enterprise Application Integration für Unternehmen ein bedeutendes Thema für Unternehmen ist.

Mit der Übernahme des Softwareherstellers Crossworlds wurde eine EAI-Software an Bord der IBM Produktpalette genommen. Crossworlds wird Bestandteil der WebSphere Reihe von IBM.
Mit EAI-Software soll es möglich sein, Prozesse „on the fly“ zu definieren, modellieren und die verschiedenen Anbindungen (MQSeries, Corba, EJB`s) in einer grafischen Benutzeroberfläche auszuwählen. Hierin zeigt sich ein wesentlicher Unterschied zu Evolution.
In Evolution wird der Workflow definiert und die Backendanbindung innerhalb Quellcode – unter Verwendung verschiedener Tools – vorgenommen. Sie werden nicht während der Modellierung ausgewählt.

Als weiterer Unterschied ist festzuhalten, das EAI-Software auch Geschäftsprozesse unterstützt, die nicht auf den Einsatz im Inter- bzw. Intranet beschränkt sind. Sie können dementsprechend auch genutzt werden, um den Workflow zwischen ERP und CRM Software – ohne Internet - zu gewährleisten.
Ob EAI-Softwarelösungen allerdings konkrete Vorgaben zur Entwicklung von Software – wie Evolution durch die Codegenerierung unter Verwendung eines Frameworks - machen kann, wird sich zeigen müssen.

Möglicherweise kann Evolution als Vorgehensmodell an Crossworlds – und den darin enthaltenen Designtools - angebunden werden .

7.5 Abschließende Bewertung

Das Thema *Enterprise Application Integration als Basis für Portale* erforderte neben Aufarbeitung der Grundlagen (im Besonderen XML), die Einarbeitung in die Bereiche Portale, EAI, Evolution (Bestandteile der Kapitel eins bis fünf) und die eingangs genannten Software (vgl. Kapitel 1.3).
Zur Entwicklung des Datenmodells und der funktionalen Anfroderungen des Prototypen wurden zahlreiche Webseiten von Versicherungen untersucht.
Als problematisch hat sich das Fehlen einer Dokumentation über Evolution und der daraus resultierenden Einarbeitung in den Quelltext, dessen Klassenhierarchien - und somit letztlich jeder einzelnen Klasse - erweisen.
Neben der Erzeugung der Testdaten[98] hat sich besonders die Installation des WebSphere Portal Server als aufwendig[99] und fehleranfällig erwiesen.

Nach der Entwicklung des umfangreichen Protoypen[100] und der anschliessenden Integration mit dem IBM WebSphere Poral Server in der Version 2.1 bleibt bei der Verwendung von Portalsoftware derzeit das Fehlen einer Standard-API zu bedenken:

Zum einen binden sich Unternehmen momentan an einen Portalserverhersteller und sind somit von seinen Vorgaben abhängig. Ein einzelnes Unternehmen stellt eine API schneller um, als der Java Community Process (JCP) mit seinen zahlreichen Mitgliedern. Zum anderen werden die verschiedenen Portalserver Hersteller mit einer Standard-Portlet-API ihre Programmierschnittstellen anpassen - müssen.

Der Erfolg von Portalsoftware hängt neben Preis, Performance Integrierbarkeit in die bestehenden Systeme usw. wesentlich mit der Anzahl und Qualität der verfügbaren Portlets ab. Bereits jetzt existieren für den WPS zahlreiche kostenlose und unter Lizenz veröffentlichte Portlets. Hervorzuheben sind hier SAP, Lotus etc.
Die verschiedensten Softwarehersteller (also nicht nur die Portalserver Anbieter) arbeiten derzeit mit Hochdruck daran, Portlets für ihre Software zu entwickeln und

[98] Insgesamt wurden 17 Tabelen mit 158 Spalten und 4869 Zeilen erzeugt.
[99] zahlreiche Fixpacks sind zu installieren
[100] mit sechs beschriebenen Use Cases, 15 Aktivitäten (sie alleine – ohne die Geschäfts- oder Anzeigeobjekte und XSL-Seiten enthalten 3321 Zeilen Code)

somit den eingangs erwähnten „Single Point of Access“ im Inter- bzw. Intranet zu ermöglichen.

Mit der Integration der Eigenentwicklung der Unternehmen in ein Portal ist ein Projektgeschäft verbunden, das immer von den Randbedingungen eines Projektes und der Architektur vor Ort abhängig ist. Es ist im Einzelfall vom Kunden abzuwägen, wie neu zu erstellende Anwendungen in das Portal gelangen und bereits existierende integriert werden. Ein Weg wurde mit dieser Diplomarbeit vorgestellt. Auf die Anbindung von MQSeries wurde aufgrund des Prototypen-Charakter verzichtet.

Aktueller Stand der Portlet-API

Betrachtet man die aktuelle Version von Jetspeed – der geplanten Referenz-implementierung der bevorstehenden Portal-API - sowie die mittlerweile erschienene Version 4.1 des IBM WebSphere Portal Server, sind bereits jetzt einige Änderungen ersichtlich.

Neben kleineren Detailänderungen und weiteren Events[101], werden die Klassen der Portlet-API von denen der Servlet-API abgeleitet. Damit verbunden, wird ein Portlet ein Bestandteil einer WAR-Datei. Zur Migration von einem PAR- zu einem WAR-File muß das Verzeichnis PORTLET-INF in WEB-INF umbenannt werden. Konfiguriert wird dann ein WAR-File sowohl über die Datei web.xml. als auch portlet.xml.(vglportlet**)

Die ursprüngliche für Dezember 2002 angekündigte Referenzimplementierung ist jetzt für März 2003 geplant. Der JCP sieht jedoch noch weitere Schritte bis zur Verabschiedung einer Standard-API vor (.vgl. jcp.org). Bis dahin gilt: solange eine API nicht standardisiert ist, kann sie im JCP abgelehnt werden!

Zukunftsbetrachtung

Blickt man in die Zukunft, werden Webservices als Problemlöser für die verschiedenen Integrationsprobleme gehandelt.

Ein Web Services ist ein Dienst, der im Internet registriert ist. Er ähnelt einem Servlet, das mit verschiedenen Parametern aufgerufen werden kann. Im Unterschied zu Servlets

[101] (es wird beispielsweise ein Event geworfen, wenn sich ein Wert in der Session ändert.)

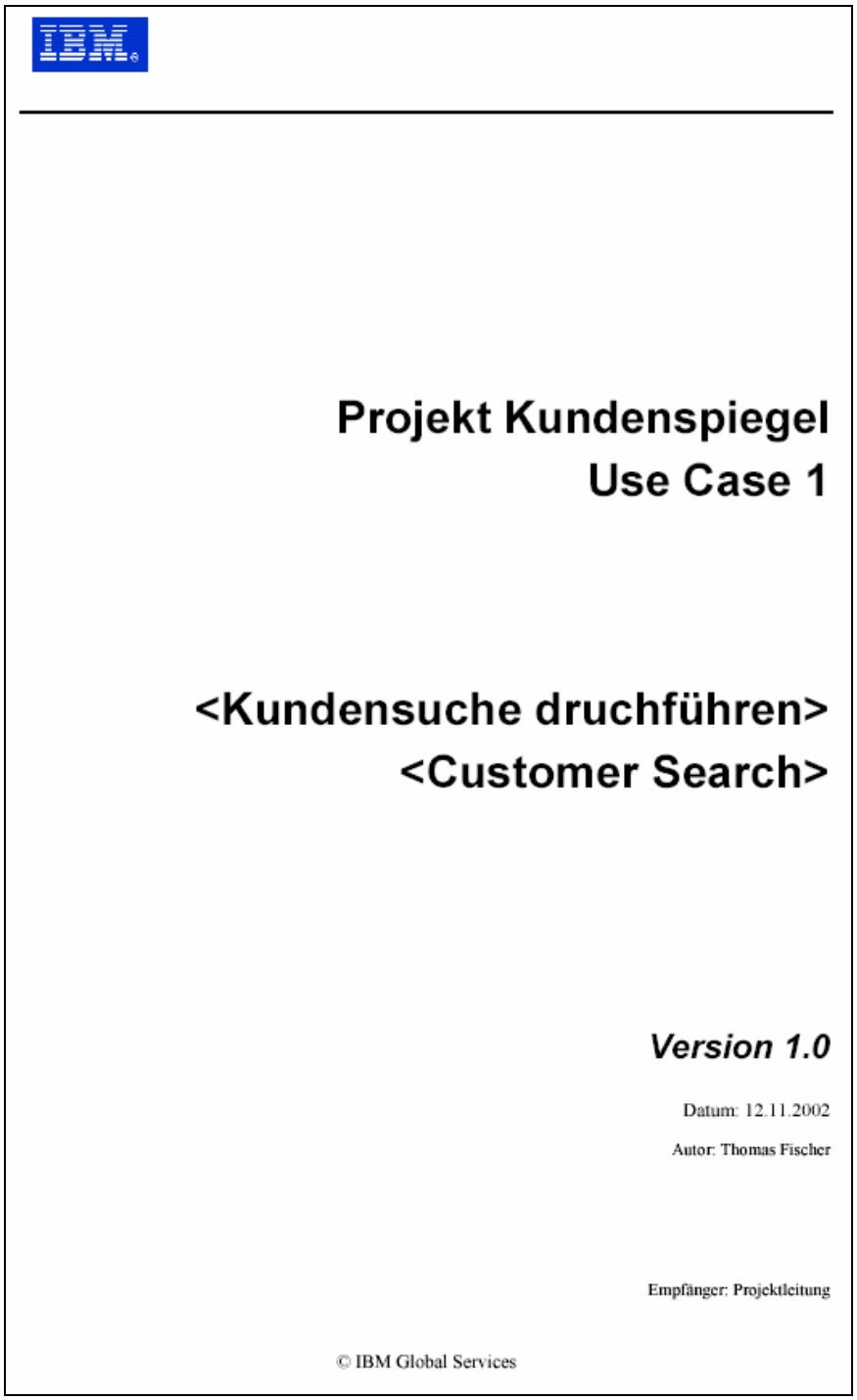
liefert der Web Service jedoch keine Ausgabe für einen Browser (also beispielsweise HTML) zurück, sondern eine XML Datenstruktur.
Zum Aufruf eines Webservice und zur Rückantwort wird das SOAP-Protokoll benutzt. Die Beschreibung des Dienstes erfolgt über die Web Services Description Language (WSDL) . Der Dienst selbst wird in der Universial Description Discovery & Integration (UDDI) Registry abgelegt.

Web Services sind also ein weiterer Weg, wie verschiedene Anwendungen miteinander kommunizieren können. Doch warum noch einen weiteren Weg?
Unternehmen schützen Ihre Rechner durch Firewalls vor Angriffen aus dem Internet. Zugriffe über Protokolle wie Corba, RMI, DCOM oder EDI sind meist blockiert. SOAP ist ein so genanntes HTTP-Tunneling-Protokoll und verwendet die freien Ports des HTTP-Protokolls und gelangt so durch die Firewalls der Unternehmen. Ob sich Web Services jedoch durchsetzen werden – und nicht lediglich als „eine nette Idee“ in Vergessenheit geraten , wird sich zeigen müssen.

8 Anhang

Anhang A: Prototyp im Applikationserver

8.1 Beispiel Use Case : Kundensuchehe druchführen



Projekt Kundenspiegel
Use Case 1

<Kundensuche druchführen>
<Customer Search>

Version 1.0

Datum: 12.11.2002

Autor: Thomas Fischer

Empfänger: Projektleitung

1 Allgemeine Informationen

1.1 Ablage

Papierform
Elektronisch xxxxxxxxxxx

1.2 Versionskontrolle

Version	*Datum*	*Beschreibung der Änderungen*	*Autor*	*Änderung markiert*
1.0	***01.08.2002***	***Erstellung***	***Fischer***	***nein***

1.3 Genehmigungen/Reviews

Dieses Dokument benötigt die folgenden Genehmigungen:

Version	*Name*	*Verweis auf Reviewergebnis/Genehmigung*

1.4 Verteiler

Dieses Dokument muß wie folgt verteilt werden:

Name / Verteilerliste	*Abteilung / Gesellschaft*

2 UseCase Beschreibung

2.1 Use Case Name / Use Case ID

<Kundensuche druchführen>
<CustomerSearch>

2.2 Kurzbeschreibung – Zweck

[Kurze und prägnante Beschreibung über den fachlichen Zweck dieses UC. Diese Beschreibung drückt das durch die Handlung der Akteure zu erreichende Ziel aus.]

Der UC Kundensuche durchführen dient – nach erfolgreicher Authentifizierung – als Einstiegspunkt in den Kundenspiegel. Vom System wird eine Suchmaske angezeigt. Nach dem Ausfüllen der entsprechenden Felder und dem Starten der Suchanfrage durch den Makler wird vom System eine Liste der Kunden angezeigt, die den Suchkriterien entsprechen.
Durch Auswahl eines Kunden gelangt der Makler anschließend zur Kundenübersicht <UC Kundenübersicht einsehen>

2.3 Akteure

System, Makler

2.4 Auslöser

Der Makler möchte Kundendaten einsehen, bzw. ändern

2.5 Voraussetzungen

[Voraussetzungen die gegeben sein müssen, dass der UC gestartet werden kann (z.B. Aussagen über den Status von Objekten der Domäne).]

Makler ist autorisiert

2.6 Nachbedingungen

[Markante Statusänderungen als Ergebnis der UC-Bearbeitung.]

2.7 Standardablauf

[Detaillierte Beschreibung des normalen Ablaufes der Interaktion zwischen dem Aktor und dem System. Der Ablauf wird bis zur Erreichung des Ziels aus Sicht des Aktors beschrieben. Ausnahmen wie z.B. Eskalationsstufen, werden hier nicht beschrieben. Die Beschreibung muss deutlich machen, welche Aktionen durch den Aktor und welche durch das System erbracht werden. Mögliche alternative Abläufe der Interaktion müssen erkennbar sein!]

2.7.1 Kundendaten suchen

Akteur	Aktionen
System	Das System erzeugt eine Suchmaske mit folgenden Eingabefeldern: Name, Vorname PLZ Ort Angebotsnummer Versicherungsnummer

	Schadennummer Kfz Kennzeichen Geburtsdatum
Makler	Der Maklegibt die gewünschten Suchkriterien ein und betätigt den Suche Button:
System	1.) Das System prüft die Eingabedaten (siehe 2.10 Plausibilitäten 1-8) Wenn Fehler aufgedeckt werden, zeigt das System diese an. 2.) Gibt der Akteur mehrere Kriterien ein, werden diese durch eine Und-Verknüpfung gesucht. 3) Die gefundenen Daten werden als Liste mit folgenden Feldern dargestellt: Name, Vorname PLZ Ort Strasse Geburtsdatum Die Liste wird erst nach Name und dann PLZ sortiert.

2.7.2 Kundendaten wählen

Akteur	Aktionen
Makler	Der Makler wählt einen Kunden aus der vorher erstellten Kundenliste aus. Dadurch wird ein Kunde identifiziert. Diese Identifizierung ist Grundlage um in die Kundenübersicht zu gelangen.

2.8 Variationen vom Standardablauf

[Detaillierte Beschreibung der Abweichungen vom Standardablauf. Die Beendigung der Bearbeitung bei fachlichen Ausnahmesituationen muss hier dargestellt werden.]

2.8.1 Kein Kunde gefunden

Akteur	Aktionen
System	Wird vom System kein Kunde entsprechend den Suchkriterien des Maklers gefunden, wird die Fehlermeldung Meldungen.doc#F10 angezeigt

2.8.2 weitere Eingrenzung der Ergebnisliste

Akteur	Aktionen
System	Das System zeigt die eingegebenen Suchkriterien nach Ermittlung der Trefferliste erneut an.
Makler	Der Makler fügt weitere Suchkriteriterien in die Suchmaske ein und betätigt die Suche
System	Das System zeigt die ermittelten Daten in der Ergebnisliste an.

2.8.3 Makler ist nicht berechtigt den Kundeneintrag einzusehen

Akteur	Aktionen
System	Die Kundendaten werden nicht angezeigt

2.9 Ausnahmen und Fehlersituationen

[Liste von fachlichen Ausnahmen bei der Beendigung des UC (z.B. der Kunde unterschreibt den Antrag nicht: Was ist zu tun...).]

- keine

2.10 Plausibilitäten

[Liste der UC spezifische Plausibilitäten (Plausibilitäten werden unabhängig hiervon zentral erfasst).]

1. Das Format der Schadensnummer, Versicherungsnummer ist zu beachten

2.11 Quellenangabe

[Liste von Referenzen auf weitergehende Dokumente oder Dokumente, die Grundlage für die Erstellung dieses Use Cases sind]

- Keine

2.12 Offene Punkte

[Liste von offenen und zu klärenden Fragen zu dem UC]

Nr.	Beschreibung	Verantw.	Lösung bis:	Status

8.2 Klassendiagramme

8.2.1 Geschäftsobjekte

Aus Gründen der Übersichtlichkeit die Geschäftsobjekte auf zwei Abbildungen aufeteilt.

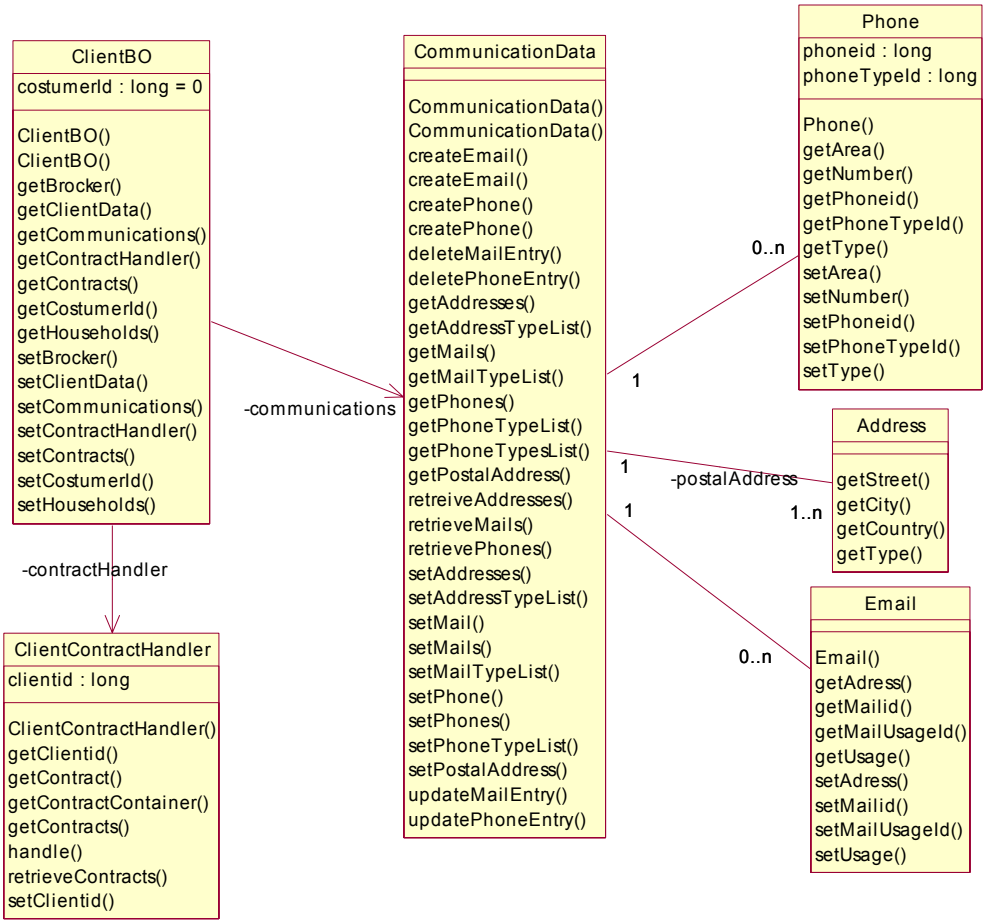


Abbildung Anhang A-0-1 Geschäftsobjekte (1)

Einem „ClientBO“ Geschäftsobjekt ist ein ClientContractHandler zugeordnet. Dieser Handler ermittelt die Verträge eines Kunden. Die jeweilige Vertragsart –bzw. Sparte (Household-, Live- und VehicleContract) ist in eigenen Tabellen abgelegt.

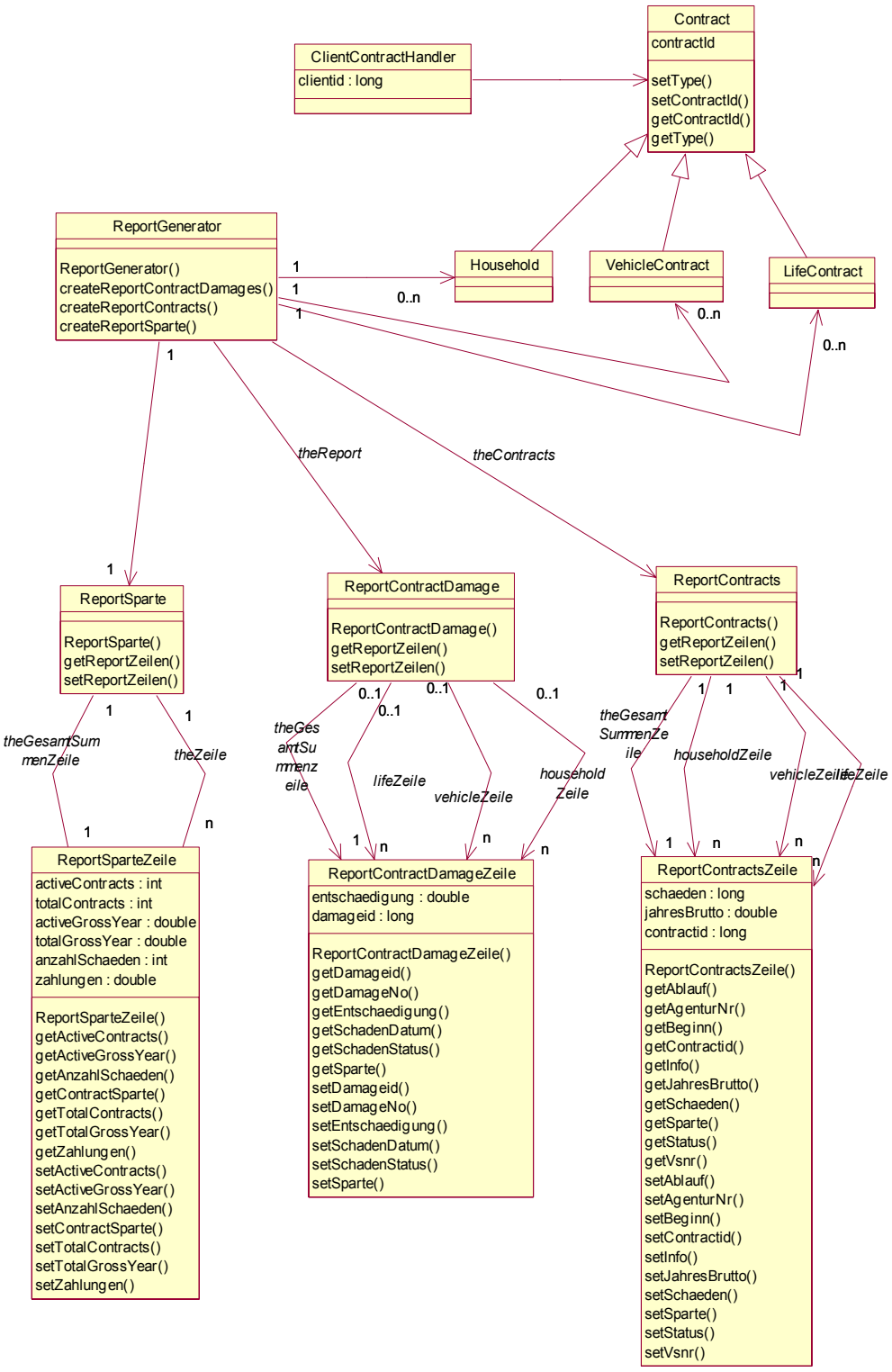


Abbildung Anhang A-0-2 Geschäftsobjekte (2)

Der ReportGenerator kapselt die Erzeugung der verschiedenen Reports. Jeder Report erzeugt verschiedene Zeilen. Diese Zeilen werden innerhalb der Aktivitaetsklassen (innerhalb der Methode performExecute()) in Anzeigeobjekte überführt.

Die Klasse ReportContractDamage wird zur erzeugt die ReportDaten des <UC 4 Vertragsdetails bearbeiten - Schadenliste einsehen> (vgl. Abbildung Anhang A-0-9). Im Einzelnen gehören dazu: Schadensnummer, Sparte, Datum, Status sowie Entschädigungsbetrag. Die Daten werden aus den entsprechenden Vertragstabellen sowie deren Schadenstabellen verwendet.

Die Klasse ReportContract erzeugt die ReportDaten des <UC 2 Kundenübersicht einsehen> (vgl. Abbildung Anhang A-0-7).
Neben der Ermittlung der einzelnen Verträge samt Sparte, Status, Schäden, Beginn, Ablauf, Jahresbrutto, einer Informationsspalte wird dynamisch – und nicht innerhalb der Datenbank - die Summe der Spalte Jahresbrutto ermittelt.

Die Klasse ReportSparte erzeugt die ReportDaten des <UC 4 Kundenreport einsehen>. Im Gegensatz zur Klasse ReportContract werden die Gruppensätze der verschiedenen Sparten (KFZ, Hausrat und Leben) erstellt. Als Bestandteil werden jeweils ermittelt:
Gesamtzahl sowie Anzahl der aktiven Verträge
Jahresbruttobeträge der gesamten bzw. aktiven Verträge
Anzahl der Schäden, sowie deren Schadenswert des ausgewählten Schadenszeitraums.

8.2.2 Klassendiagramm Anzeigeobjekte

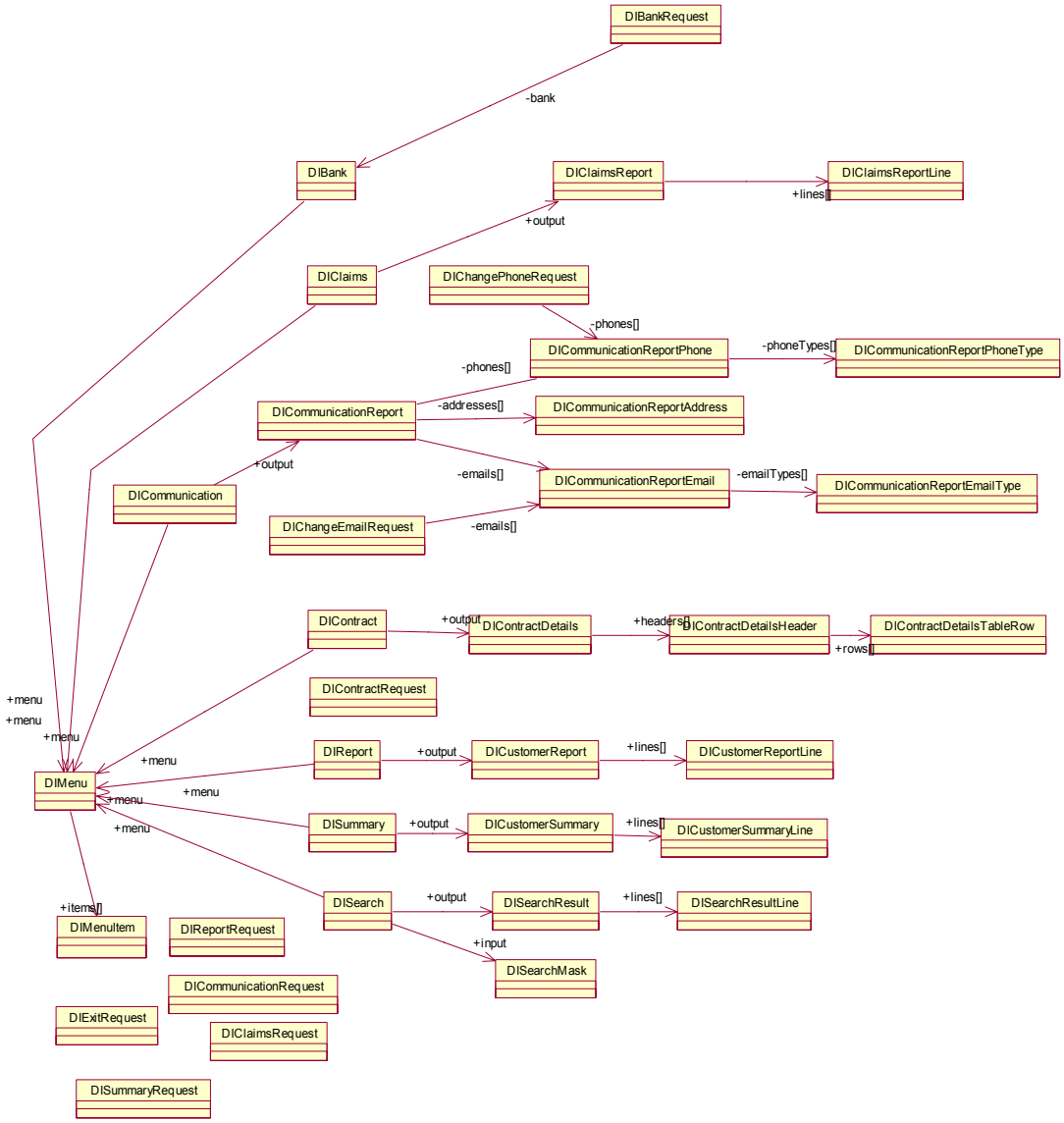
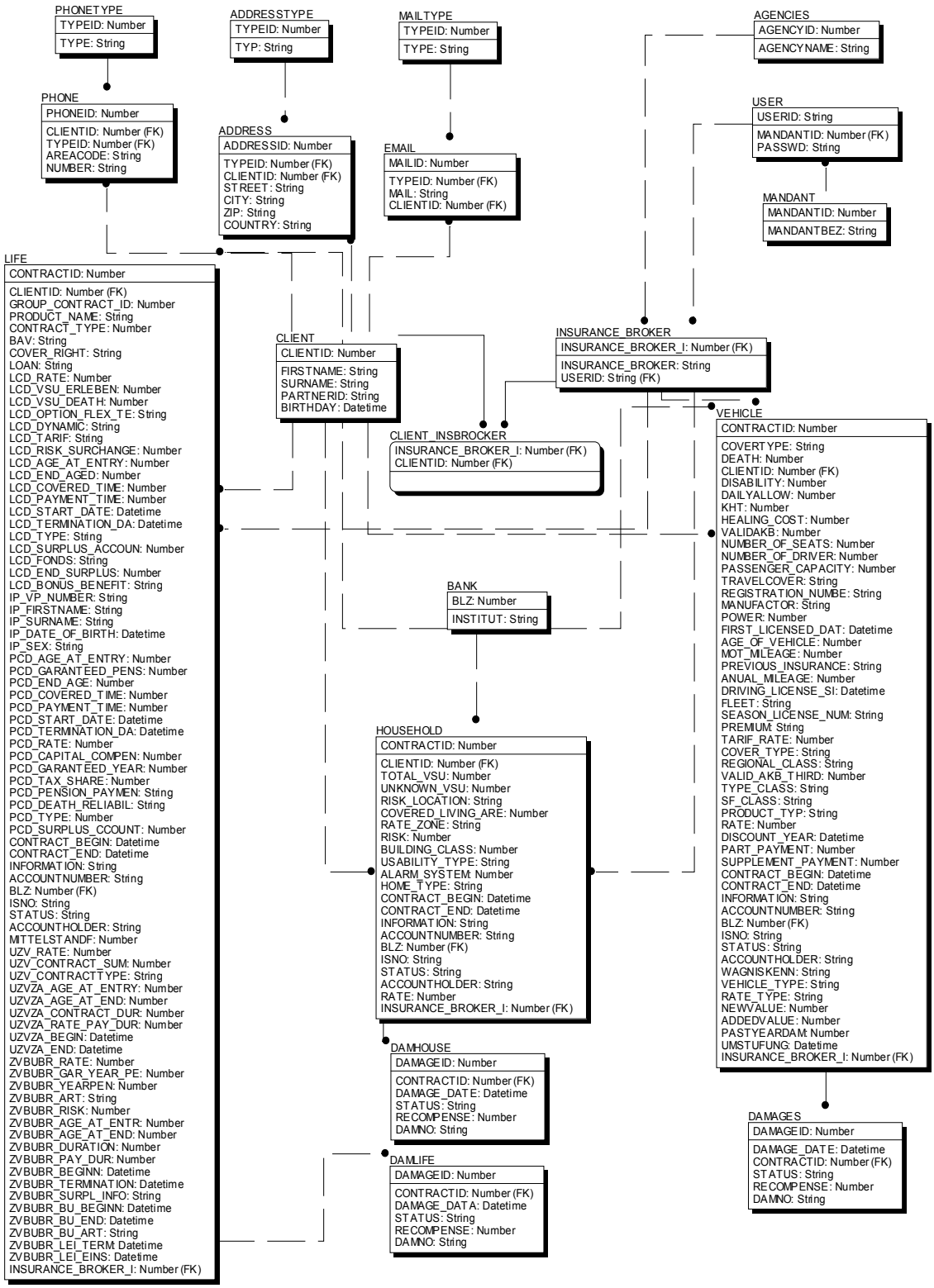


Abbildung Anhang A-0-3 Überblick Anzeigeobjekte

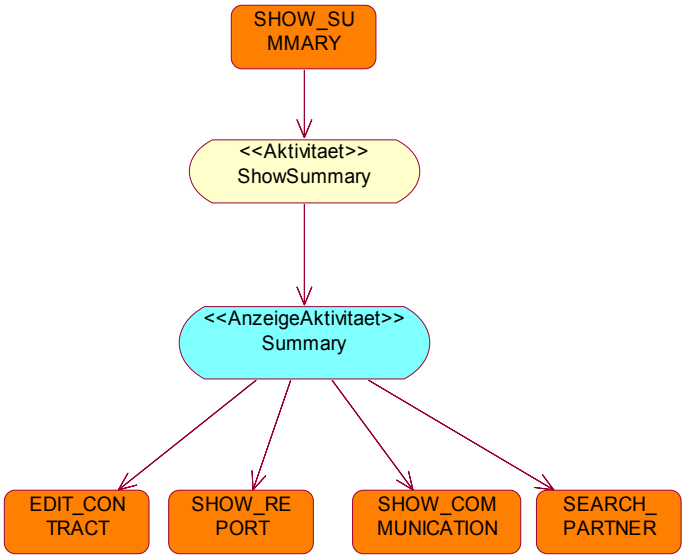
8.3 physikalisches Datenmodell



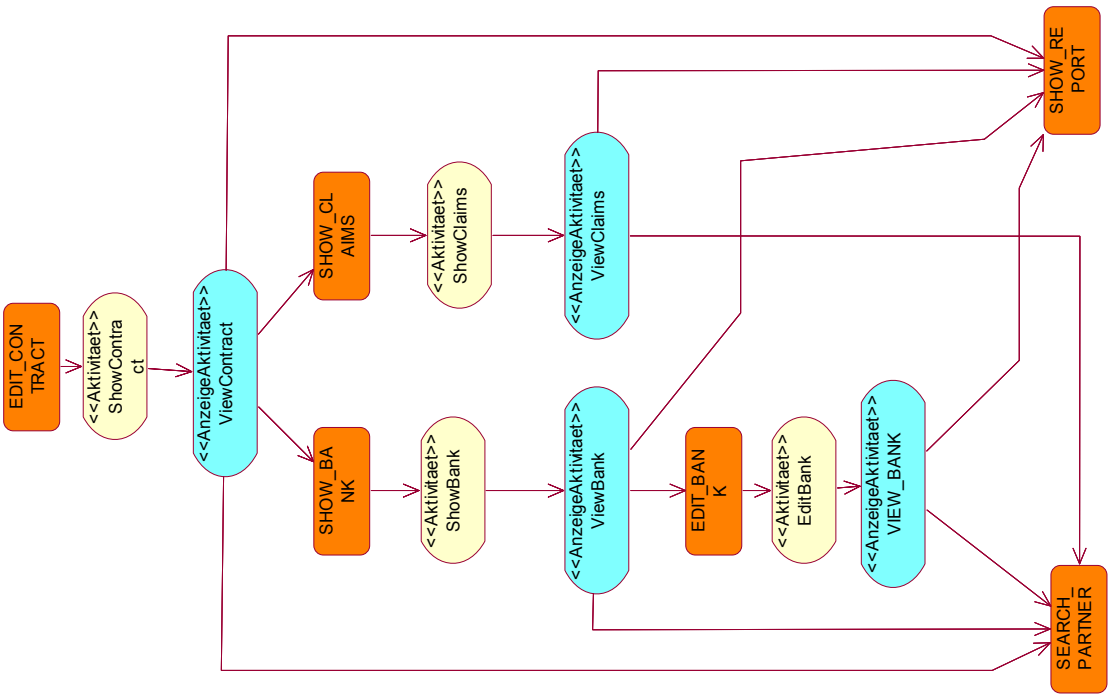
8.4 Der Workflow

Als Ergänzung zu den Abbildung 6-3 Aktivitätsdiagramm des Prozesses PartnerSearch und Abbildung 6-4 Vorgänge des Prozesses PartnerSearch folgen die weiteren Aktivitätsdiagramme des Prototypen

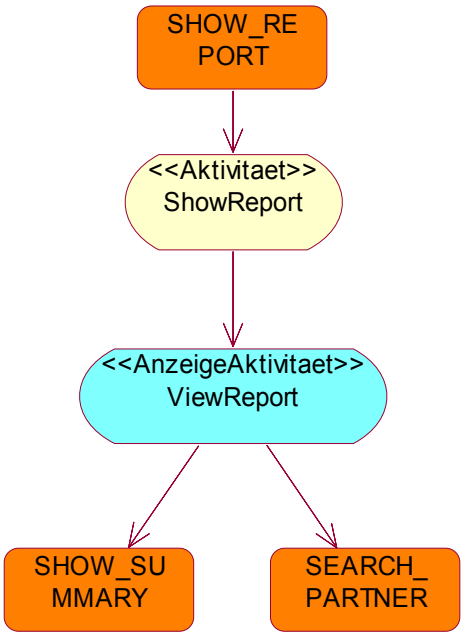
8.4.1 UC 2 Kundenübersicht einsehen:



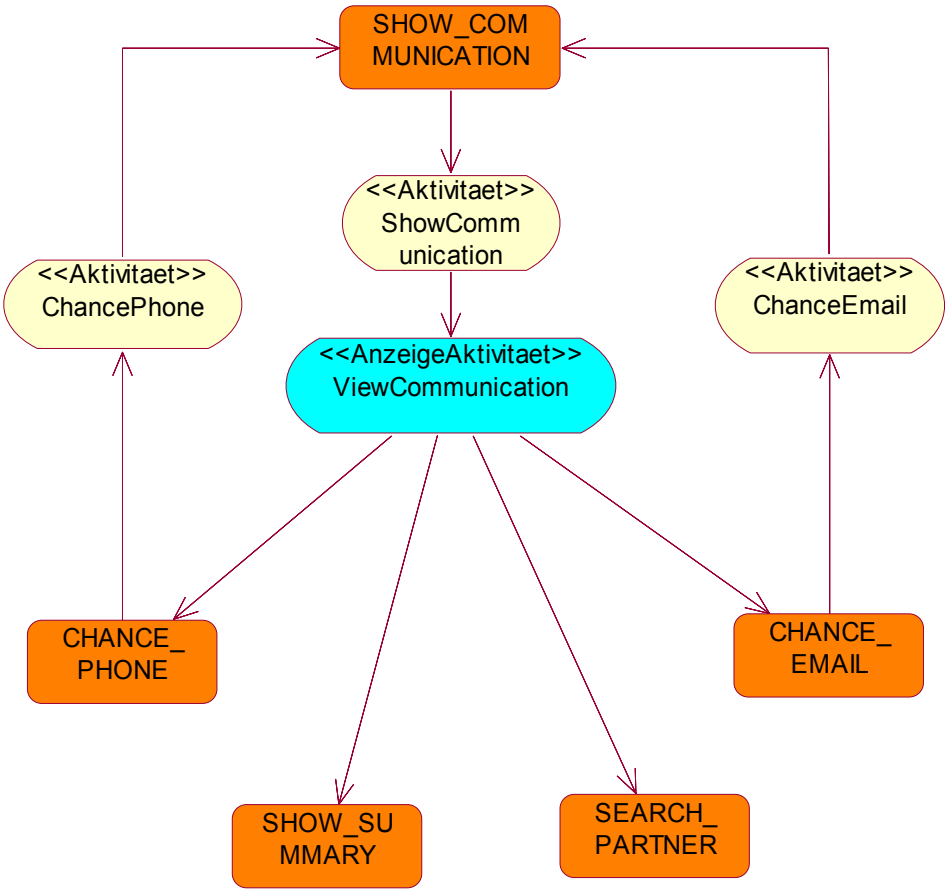
8.4.2 UC 3 Vertragsdetails bearbeiten



8.4.3 UC 4 Kundenreport einsehen



8.4.4 UC5 Kommunikationsdaten bearbeiten



8.5 Quellcode einer Aktivitaetsklasse

Es wurden insgesamt wurden 15. Aktivitaeten erzeugt – alle sind dem Paket com.ibm.cis.activities zugeordnet. Es folgt exemplarisch der Quellcode der Klasse SearchPartner.

```
package com.ibm.cis.activities;

import com.ibm.evolution.patterns.*;
import
com.ibm.evolution.patterns.exceptions.*;
import com.ibm.evolution.control.*;
import com.ibm.evolution.configuration.*;
import com.ibm.evolution.constants.*;


// user code begin {IMPORT-STATEMENTS}
import com.ibm.evolution.display.*;
import com.ibm.cis.business.client.*;
import java.util.*;
import
com.ibm.cis.sessions.PartnerSearchSession;
import com.ibm.cis.processes.*;
import com.ibm.cis.displayobjects.*;
import com.ibm.cis.helpers.*;
import com.ibm.cis.querybuilder.*;
import com.ibm.cis.data.*;

// user code end {IMPORT-STATEMENTS}
public class SearchPartner extends
com.ibm.evolution.control.Activity {


       // user code begin {INSTANCES AND
CLASSVARIABLES}
       public DisplayObject displayObject;
       // user code end {INSTANCES AND
CLASSVARIABLES}

       /**
         * Save the current state of the
object
         */
       private java.lang.String state;

       /**
         * Allowed output state of the
activity
         */
       public static final
java.lang.String START = "START";

       /**
         * Allowed input state of the
activity
         */
       public static final
java.lang.String SEARCH_PARTNER =
"search_partner";

       /**
         * Allowed input state of the
activity
         */
       public static final
java.lang.String SHOW_SUMMARY =
"show_summary";
/**
  * Der Standardconstructor der Class.
  */
public SearchPartner() {
       super();

       // user code begin {USERDEFINIERTER
KONSTRUKTOR}
       // user code end {USERDEFINIERTER
KONSTRUKTOR}
}
/**
  * Dies ist der Getter für die
Instanzvariable state.
  * @return Den Value from state
  */
public java.lang.String getState() {
       return state;
}
/**
  * Diese Methode legt fest welche
Eingabedaten zwingend erforderlich sind.
  * Fehlen Daten, wird false
zurückgegeben oder eine
NotPreparedException
  * geworfen.
  * @return true, wenn alle
erforderlichen Daten vorhanden sind,
ansonsten false
  * @throws NotPreparedException
  */
protected boolean isPrepared() throws
NotPreparedException {

       // Kontollvariable, false wenn
nicht alle erforderlichen Daten
vorhanden
       boolean controlFlag = true;

       // user code begin {isPrepared
USER-BLOCK}
       // user code end {isPrepared
USER-BLOCK}
       return controlFlag;
}
/**
  * Diese Methode legt fest,ob die
Eingabedaten plausibel sind.
  * Sind Daten nicht plausibel, wird
false zurückgegeben oder eine
InvalidException geworfen
  * @return true, wenn alle Daten
plausibel sind, ansonsten false
  * @throws CommandException
  */
protected boolean isValid() throws
CommandException {

       // Kontollvariable, false wenn
nicht alle Daten plausibel sind
       boolean controlFlag = true;

       // user code begin {isValid USER-
BLOCK}
       // user code end {isValid USER-
BLOCK}
       return controlFlag;
}
/**
  * Diese Methode Enthält die
eigentliche fachliche Logik und den
Workflow.
  * Zusätzlich werden hier die Error
abgefangen oder weitergeleitet.
  * @throws CommandException
  */
protected void performExecute() throws
CommandException {


       // user code begin
{performExecute USER-BLOCK}
       PartnerSearchSession session =
(PartnerSearchSession) getSession();
    DISearchMask inputMask = new
DISearchMask();

    if (session.getUserInput()
instanceof DISearchMask) {
```

```
        inputMask = (DISearchMask)
session.getUserInput();
    }

    try {

        Configuration config = new
Configuration();

        DISearch search = new DISearch();

search.setServiceId(getServiceId());

        session.setContractId(-1);
        session.setClientId(-1);

search.setMenu(MenuBuilder.buildMenu(sessio
n));

        DISearchMask searchMask = new
DISearchMask();

        searchMask.setLastName(
            inputMask.getLastName() != null
? inputMask.getLastName() : "");
        searchMask.setFirstName(
            inputMask.getFirstName() !=
null ? inputMask.getFirstName() : "");

searchMask.setZip(inputMask.getZip() !=
null ? inputMask.getZip() : "");

searchMask.setCity(inputMask.getCity() !=
null ? inputMask.getCity() : "");
        searchMask.setProposalId(
            inputMask.getProposalId() !=
null ? inputMask.getProposalId() : "");
        searchMask.setInsuranceId(
            inputMask.getInsuranceId() !=
null ? inputMask.getInsuranceId() : "");
        searchMask.setClaimId(
            inputMask.getClaimId() != null
? inputMask.getClaimId() : "");
        searchMask.setLicensePlateId(
            inputMask.getLicensePlateId()
!= null ? inputMask.getLicensePlateId() :
"");

System.out.println(inputMask.getLastName())
;

        search.setInput(searchMask);

        DISearchResult searchResult = new
DISearchResult();

        if
(getState().equals(SEARCH_PARTNER) &&
!inputMask.isEmpty()) {

            InputFields fields = new
InputFields();
            FieldDataMapping maps = new
FieldDataMapping();
            QueryBuilder builder = new
QueryBuilder();


fields.addField(Constants.SEARCH_LASTNAME,
inputMask.getLastName());

fields.addField(Constants.SEARCH_FIRSTNAME,
inputMask.getFirstName());

fields.addField(Constants.SEARCH_ZIP,
inputMask.getZip());

fields.addField(Constants.SEARCH_CITY,
inputMask.getCity());

       fields.addField(Constants.SEARCH_IS
NO, inputMask.getInsuranceId());

       fields.addField(Constants.SEARCH_PR
OPOSAL, inputMask.getProposalId());

       fields.addField(Constants.SEARCH_CL
AIM, inputMask.getClaimId());

fields.addField(Constants.SEARCH_PLATE,
inputMask.getLicensePlateId());

            maps.addMappingEntry(

Constants.SEARCH_LASTNAME,

config.readAttribute(getServiceId(),
"database&search&lastName"));
            maps.addMappingEntry(
                Constants.SEARCH_ZIP,

config.readAttribute(getServiceId(),
"database&search&zip"));
            maps.addMappingEntry(
                Constants.SEARCH_CITY,

config.readAttribute(getServiceId(),
"database&search&city"));

            Client dataObj = new
Client();
            dataObj.initialise();

            String query =
builder.buildQuery(fields, maps,
session.getBrokerId());

            Vector v =
dataObj.searchClient(query);

            Hashtable result = new
Hashtable();

            for (Enumeration e =
v.elements(); e.hasMoreElements();) {
                Client re = new
Client();
                re = (Client)
e.nextElement();
                ClientBO clientBO = new
ClientBO(re.getClientid());
                result.put(new
Long(re.getClientid()).toString(),
clientBO);
            }

            if (!result.isEmpty()) {

                for (Enumeration e =
result.elements(); e.hasMoreElements();)
{
                    ClientBO clientBO =
(ClientBO) e.nextElement();
                    DISearchResultLine
line = new DISearchResultLine();

line.setLastName(clientBO.getClientData(
).getSurname());

line.setFirstName(clientBO.getClientData
().getFirstname());

com.ibm.cis.business.communication.Addre
ss postalAddress =

clientBO.getCommunications().getPostalAd
dress();
                    if (postalAddress !=
null) {

line.setZip(postalAddress.getZip());

line.setCity(postalAddress.getCity());

line.setStreet(postalAddress.getStreet()
);
                    }


                     line.setLink(
                        "?serviceId=" +
session.getServiceId() +
"&nameDO=com.ibm.cis.displayobjects.DISu
mmaryRequest&actionName=" +
PartnerSearch.SHOW_SUMMARY +
"&customerId="
```

```
                                +
clientBO.getClientData().getClientid());


searchResult.addLines(line);
                } //for end
            } else {

searchResult.setErrorString("Keine Treffer
gefunden!");
            }

        }

        session.setSearchMask(inputMask);

        search.setOutput(searchResult);

        displayObject = search;
        setState(SHOW_SUMMARY);

    } catch (Exception e) {
            e.printStackTrace();
        throw new
FatalExecutionException();
    }
```

```
        // user code end {performExecute
USER-BLOCK}
}
/**
  * Diese Methode enthält den Code, der
für ein manuelles Rollback benötigt
wird.
  * Sie wird automatisch aufegrufen,
wenn die Methode reset() aufgerufen
  * wird.
  */
protected void rollback() {

        // user code begin {ROLLBACK
USER-BLOCK}
        // user code end {ROLLBACK USER-
BLOCK}
}
/**
  * Dies ist der Setter für die
Instanzvariable state.
  * @param state
  */
public void setState (java.lang.String
state) {
        this.state = state;
}
}
```

8.6 Beispiel Anzeigeobjekt in XML

Folgend wird das aus der Java-Klasse com.ibm.cis.displayobjects.DISearch überführte XML Dokument vorgestellt. Das äußere Element enthält den voll-qualifizierenden Java-Klassennamen.Im Wesentlichen besteht (in diesem Fall) das Anzeigeobjekt DISearch aus einem Menu (mit zwei Menüeinträgen), einem leeren Ausgabe (Element output), sowie den Parametern von der Aufrufenden Seite (Es wurden keine Parameter in DISearchMask eingetragen)

```
<com.ibm.cis.displayobjects.DISearch>
<menu>
        <com.ibm.cis.displayobjects.DIMenu>
<outputFormat>html</outputFormat>
<nameDO>com.ibm.cis.displayobjects.DIMenu</nameDO>
        <items POS="0">
<com.ibm.cis.displayobjects.DIMenuItem>
<menuActionName>search_partner</menuActionName>
<nextDO>com.ibm.cis.displayobjects.DISearchMask</nextDO>
<outputFormat>html</outputFormat>
<nameDO>com.ibm.cis.displayobjects.DIMenuItem</nameDO>
<label>images/but_search_p.gif</label>
</com.ibm.cis.displayobjects.DIMenuItem>
</items>
        <items POS="1">
<com.ibm.cis.displayobjects.DIMenuItem>
        <menuActionName>logout</menuActionName>
        <nextDO102>com.ibm.cis.displayobjects.DIExitRequest</nextDO>
        <outputFormat>html</outputFormat>
        <nameDO>com.ibm.cis.displayobjects.DIMenuItem</nameDO>
        <label>images/but_exit.gif</label>
        </com.ibm.cis.displayobjects.DIMenuItem>
        </items>
</com.ibm.cis.displayobjects.DIMenu>
</menu>
<output>
        <com.ibm.cis.displayobjects.DISearchResult>
        <outputFormat>html</outputFormat>
<nameDO>com.ibm.cis.displayobjects.DISearchResult</nameDO>
        </com.ibm.cis.displayobjects.DISearchResult>
</output>
<input>
<com.ibm.cis.displayobjects.DISearchMask>
```

```
<city />
<insuranceId />
<proposalId />
<licensePlateId />
<firstName />
<outputFormat>html</outputFormat> <claimId />
<nameDO>com.ibm.cis.displayobjects.DISearchMask</nameDO>
                <zip />
                <lastName />
        </com.ibm.cis.displayobjects.DISearchMask>
</input><xslPath>search/search.xsl</xslPath>
<xslRoot>c:/kundenspiegel/app/web/</xslRoot>
<outputFormat>html</outputFormat>
<baseURL>http://localhost:8080/internet_customer_search/</baseURL>
<serviceId>internet_customer_search</serviceId> </com.ibm.cis.displayobjects.DISearch>
```

Die Elemente xslPath sowie xslRoot werden über die Konfiguration des Frameworks bestimmt. Die Verbindung aus Klassennamen und dem Ausgangsstatus des Workflows ist der Eintrag „xslPath“ zugewiesen. Diese beiden Parameter bestimmen die zur Transformation zur verwendende XSL-Datei.

Das Element „outputFormat“ gibt das zu erzeugende Ausgabeformat (HTML oder PDF) an.

„ServiceID“ bezeichnet den aktuellen Mandanten.

8.7 Beispiel XSL-Seite

Als Beispiel einer XSL-Datei zur Transformation der Daten aus XML zu HTML folgt der Inhalt der Datei Search.xsl

```
<?xml version="1.0" encoding="ISO-8859-
1"?>


<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Tr
ansform">
<!DOCTYPE xsl:stylesheet [
        <!ENTITY % HTMLlat1 SYSTEM
"c:/kundenspiegel/app/Rhythmyx/DTD/HTMLl
at1x.ent">
                %HTMLlat1;
        <!ENTITY % HTMLsymbol SYSTEM
"c:/kundenspiegel/app/Rhythmyx/DTD/HTMLs
ymbolx.ent">
                %HTMLsymbol;
        <!ENTITY % HTMLspecial SYSTEM
"c:/kundenspiegel/app/Rhythmyx/DTD/HTMLs
pecialx.ent">
                %HTMLspecial;
]>

<xsl:template match="/">
        <html>
        <head>
        <link rel="stylesheet"
href="styles/styles.css"
type="text/css"/>
        </head>
                <body>
                <table width="100%"
height="100%" cellpadding="0"
cellspacing="0">
                <col span="1"
width="150"/>
                <tbody>
                        <tr>
                                <td align="center"
height="45" width="127">
                                        <img
src="images/ibm.gif" width="86"
height="30" border="0" align="middle"/>
                                </td>
                                <td align="right"
height="50" width="871">
                                        <img
src="images/emakler.gif" width="257"
height="36" border="0" align="middle"/>
                                </td>
                        </tr>
                                <xsl:apply-
templates select="*" mode="mode26"/>
                </tbody>
                </table>
                </body>
        </html>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DIMenu
Item" mode="mode1">
        <xsl:for-each select=".">
        <input type="hidden"
name="actionName">
                <xsl:attribute
name="value"><xsl:value-of
select="menuActionName"/>
                </xsl:attribute>
        </input>
        </xsl:for-each>
</xsl:template>
```

```
<xsl:template
match="com.ibm.cis.displayobjects.DIMenu
Item" mode="mode2">
        <xsl:for-each select=".">
                <input type="image"
name="MenuItem">
                        <xsl:attribute
name="src">
                                <xsl:value-
of select="label"/>
                        </xsl:attribute>

        <xsl:attribute name="alt">
                                <xsl:value-
of select="label"/>
                        </xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DISubM
enuItem" mode="mode3">
        <xsl:for-each select=".">
                <input type="hidden"
name="nameDO">
                        <xsl:attribute
name="value"><xsl:value-of
select="subNextDO"/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>
<xsl:template
match="com.ibm.cis.displayobjects.DISubM
enuItem" mode="mode4">
        <xsl:for-each select=".">
                <input type="hidden"
name="actionName">
                        <xsl:attribute
name="value"><xsl:value-of
select="subActionName"/>
                        </xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DISubM
enuItem" mode="mode5">
        <xsl:for-each select=".">
                <input type="image"
name="Label">
                        <xsl:attribute
name="src"><xsl:value-of
select="subLabel"/>
                        </xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template match="subMenus"
mode="mode6">
        <xsl:for-each select=".">
                <tr>
                        <td align="right">
                                <form>

        <input type="hidden"
name="serviceId"
value="internet_customer_search"/>

        <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISub
MenuItem" mode="mode3"/>

        <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISub
MenuItem" mode="mode4"/>

        <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISub
MenuItem" mode="mode5"/>
                                </form>
                        </td>
                </tr>
        </xsl:for-each>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DIMenu
Item" mode="mode7">
        <xsl:for-each select=".">
                <tr>
                        <td width="100%">
                                <table
cellpadding="0" cellspacing="0"
width="100%">

        <tbody>

        <xsl:apply-templates
select="subMenus" mode="mode6"/>

        </tbody>
                                </table>
                        </td>
                </tr>
        </xsl:for-each>
</xsl:template>

<xsl:template match="items"
mode="mode8">
        <xsl:for-each select=".">
                <tr>
                <td width="130"
align="left">
                <form>
                        <input
type="hidden" name="serviceId"
value="internet_customer_search"/>
                        <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DIMen
uItem" mode="mode0"/>
                        <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DIMen
uItem" mode="mode1"/>
                        <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DIMen
uItem" mode="mode2"/>
                </form>
                <table cellpadding="0"
cellspacing="2" width="100%">
                        <tbody>
                        <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DIMen
uItem" mode="mode7"/>
                        </tbody>
                </table>
                </td></tr>
        </xsl:for-each>
</xsl:template>


<xsl:template
match="com.ibm.cis.displayobjects.DIMenu
" mode="mode9">
        <xsl:for-each select=".">
                <tr><td width="114"
align="center">
                                <table
cellspacing="0" cellpadding="5">

        <col span="1" valign="middle"
align="center"/>

        <tbody>

<xsl:apply-templates select="items"
mode="mode8"/>

        </tbody>
                                </table>
                        </td>
                </tr>
        </xsl:for-each>
</xsl:template>

<xsl:template match="menu"
mode="mode10">
        <xsl:for-each select=".">
                <tr valign="top"
align="center">
```

```
                        <td valign="top"
width="114" align="center">
                                <table
cellspacing="0" cellpadding="0">

        <col span="1" valign="middle"
align="center"/>

        <tbody>

        <xsl:apply-templates
select="com.ibm.cis.displayobjects.DIMen
u" mode="mode9"/>

        </tbody>
                                </table>
                        </td>
                </tr>
        </xsl:for-each>
</xsl:template>

<xsl:template match="firstName"
mode="mode11">
        <xsl:for-each select=".">
                <input align="left"
maxlength="36" size="10"
name="firstName" type="text"
tabindex="1">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template match="lastName"
mode="mode12">
        <xsl:for-each select=".">
                <input align="left"
maxlength="36" size="19" name="lastName"
type="text" tabindex="1">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template match="proposalId"
mode="mode13">
        <xsl:for-each select=".">
                <input align="left"
maxlength="25" size="25"
name="proposalId" type="text"
tabindex="2">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template match="insuranceId"
mode="mode14">
        <xsl:for-each select=".">
                <input align="left"
maxlength="25" size="25"
name="insuranceId" type="text"
tabindex="3">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DISear
chMask" mode="mode15">
        <xsl:for-each select=".">
                <tr><td>
        <xsl:apply-templates
select="firstName" mode="mode11"/>
                                <xsl:apply-
templates select="lastName"
mode="mode12"/>
                        </td><td>
                                <xsl:apply-
templates select="proposalId"
mode="mode13"/>
                        </td><td
align="right" valign="top">
        <xsl:apply-templates
select="insuranceId" mode="mode14"/>
                        </td></tr>
        </xsl:for-each>
</xsl:template>

<xsl:template match="zip" mode="mode16">
        <xsl:for-each select=".">
                <input align="left"
maxlength="5" size="5" name="zip"
type="text" tabindex="4">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>

<xsl:template match="city"
mode="mode17">
        <xsl:for-each select=".">
                <input align="left"
maxlength="24" size="24" name="city"
type="text" tabindex="5">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>
<xsl:template match="licensePlateId"
mode="mode18">
        <xsl:for-each select=".">
                <input align="left"
maxlength="25" name="licensePlateId"
type="text" tabindex="6" size="25">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>
<xsl:template match="claimId"
mode="mode19">
        <xsl:for-each select=".">
                <input align="left"
maxlength="25" name="claimId"
type="text" tabindex="7" size="25">
                        <xsl:attribute
name="value"><xsl:value-of
select="."/></xsl:attribute>
                </input>
        </xsl:for-each>
</xsl:template>
<xsl:template
match="com.ibm.cis.displayobjects.DISear
chMask" mode="mode20">
        <xsl:for-each select=".">
                <tr><td>
                                <xsl:apply-
templates select="zip" mode="mode16"/>
                                <xsl:apply-
templates select="city" mode="mode17"/>
                        </td>
                        <td>
                                <xsl:apply-
templates select="licensePlateId"
mode="mode18"/>
                        </td>
                        <td valign="top"
align="right">
                                <xsl:apply-
templates select="claimId"
mode="mode19"/>
                        </td>
                </tr>
        </xsl:for-each>
</xsl:template>
<xsl:template match="input"
mode="mode21">
        <xsl:for-each select=".">
        <tr>
        <td align="center">
                <form>
```

```
                        <table
width="100%">
                        <tbody>
                        <tr>

        <th>Vorname, Name</th><th
align="center">Angebotsnummer</th><th>Ve
rsicherungsnummer</th>
                        </tr>
                                <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DISea
rchMask" mode="mode15"/>
                        <tr>

        <th>PLZ/Ort</th><th>KFZ-
Kennzeichen</th><th>Schadennummer</th>
                        </tr>
                                <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DISea
rchMask" mode="mode20"/>
                        </tbody>
                </table><br /><input
type="hidden" name="serviceId"
value="internet_customer_search"/>
                        <input
type="hidden" name="nameDO"
value="com.ibm.cis.displayobjects.DISear
chMask"/>
                        <input
type="hidden" name="actionName"
value="search_partner"/>
                        <input
type="image" name="actionName"
value="search_partner"
src="images/but_search_a.gif" />
                </form>
        </td>
        </tr>
        </xsl:for-each>
</xsl:template>

<xsl:template match="lines"
mode="mode22">
        <xsl:for-each select=".">
        <tr><td>        <a>
                <xsl:attribute
name="href"><xsl:value-of
select="com.ibm.cis.displayobjects.DISea
rchResultLine/link"/></xsl:attribute>
                <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISea
rchResultLine/lastName"/>
                </a></td>
        <td><xsl:apply-templates
select="com.ibm.cis.displayobjects.DISea
rchResultLine/firstName"/>
        </td><td>
                <xsl:apply-templates
select="com.ibm.cis.displayobjects.DISea
rchResultLine/zip"/>
        </td><td><xsl:apply-templates
select="com.ibm.cis.displayobjects.DISea
rchResultLine/city"/>
        </td>
        <td><xsl:apply-templates
select="com.ibm.cis.displayobjects.DISea
rchResultLine/street"/>
        </td></tr>
        </xsl:for-each>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DISear
chResult" mode="mode23">
        <xsl:for-each select=".">
                <tr>
                <td align="center">
                <table width="100%"
cellpadding="0" cellspacing="0"
border="1">
                <tbody>
                <tr>
        <th>Name</th><th>Vorname</th><th>
PLZ</th><th>Ort</th><th>Strasse</th></tr
>
                <xsl:apply-templates
select="lines" mode="mode22"/>
                </tbody>
                </table></td></tr>
        </xsl:for-each>
</xsl:template>

<xsl:template
match="com.ibm.cis.displayobjects.DISear
chResult" mode="mode24">
        <xsl:for-each select=".">
        <tr>
        <td class="error">
                <xsl:apply-templates
select="errorString"/>
        </td></tr></xsl:for-each>
</xsl:template>

<xsl:template match="output"
mode="mode25">
        <xsl:for-each select=".">
        <tr>
        <td align="center">
        <h1>Ergebnis</h1>
        <table width="100%"
cellpadding="1" cellspacing="1">
                <tbody>
                        <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DISea
rchResult" mode="mode23"/>
                        <xsl:apply-
templates
select="com.ibm.cis.displayobjects.DISea
rchResult" mode="mode24"/>
                </tbody>
        </table>
        </td></tr></xsl:for-each>
</xsl:template>

<xsl:template match="*" mode="mode26">
        <xsl:for-each select=".">
                <tr><td  height="498"
valign="top" align="center" width="127">
                <h1>Menu</h1>
                <br/>
                <table height="100%">
                <col span="1"
width="150"/>
                        <tbody>
                        <xsl:apply-
templates select="menu" mode="mode10"/>
                        </tbody>
                </table>
                </td>
                <td height="498"
valign="top"
width="871"><h1>Kundensuche</h1>
                <table width="100%">
                        <tbody>
                        <xsl:apply-
templates select="input" mode="mode21"/>
                        <tr><td><hr
size="5"/></td></tr>
                        <xsl:apply-
templates select="output"
mode="mode25"/>
                        </tbody></table>
                </td></tr>
        </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

8.8 Screenshots

Abbildung Anhang A-0-4 Startseite Kundenspiegel

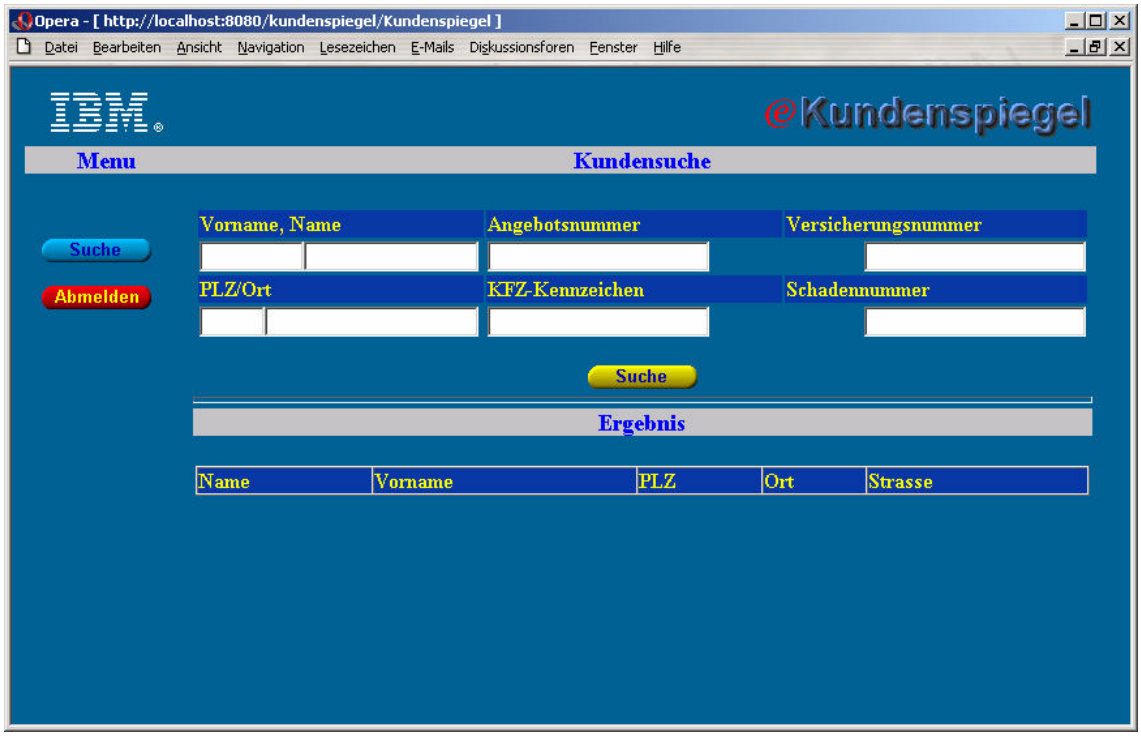


Abbildung Anhang A-0-5 UC 1 Kundensuche durchführen - Start Suche

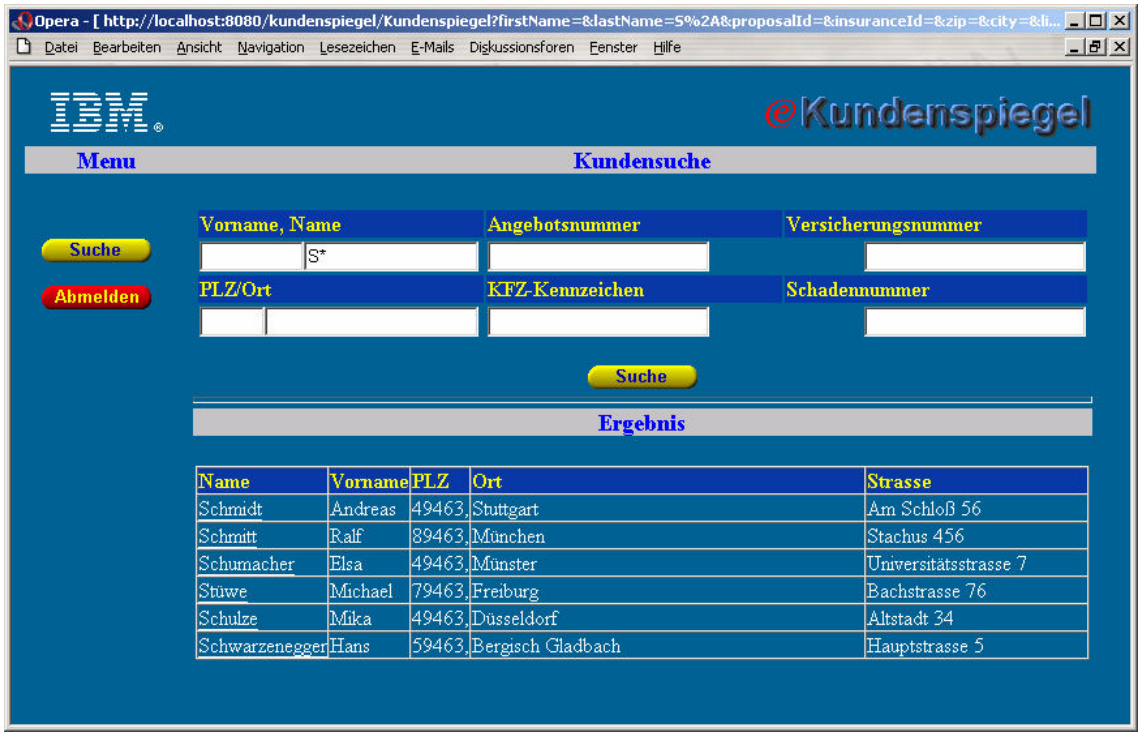


Abbildung Anhang A-0-6 UC 1 Kundensuche durchführen - Ergebnisliste einsehen

Über den Namen gelangt der Makler in den UC 2 Kundenübersicht einsehen.

Abbildung Anhang A-0-7 UC 2 Kundenübersicht einsehen

Die Vertragsübersicht bietet die Möglichkeit, die gesamten Verträge eines Versicherungskunden einzusehen.Über den Link VS Nr – der Vertragsnummer gelangt der Makler in den UC 3 Vertragsdetails einsehen.

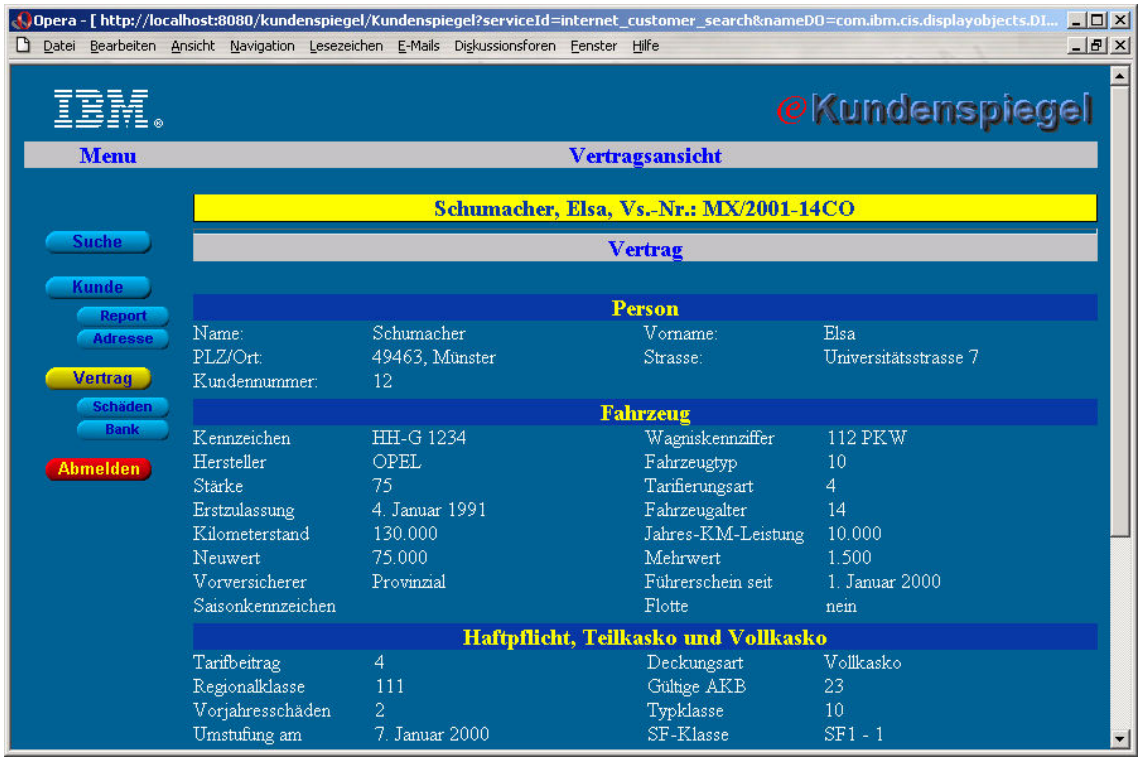


Abbildung Anhang A-0-8 UC 3 Vertragsdetails bearbeiten-Vertragsdetails einsehen

Der Abbildung ist zu entnehmen, dass die Menuleiste dynamisch erzeugt wird.

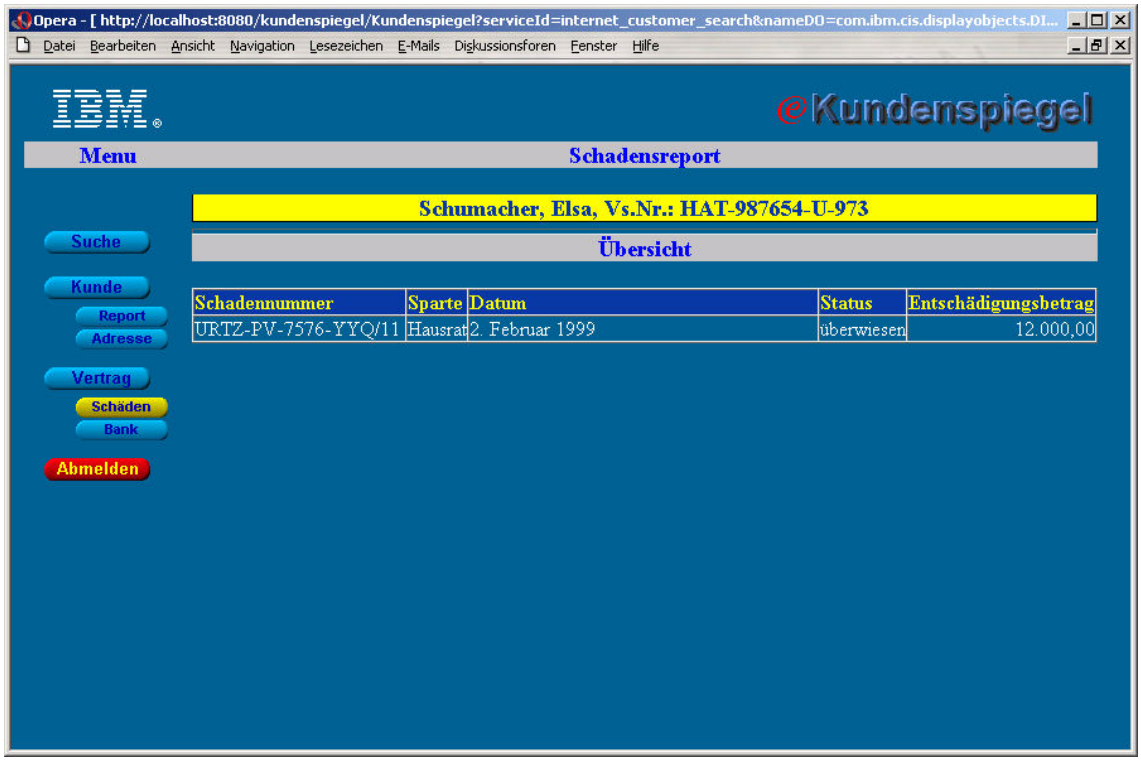


Abbildung Anhang A-0-9 UC 3 Vertragsdetails bearbeiten-Schadenliste einsehen

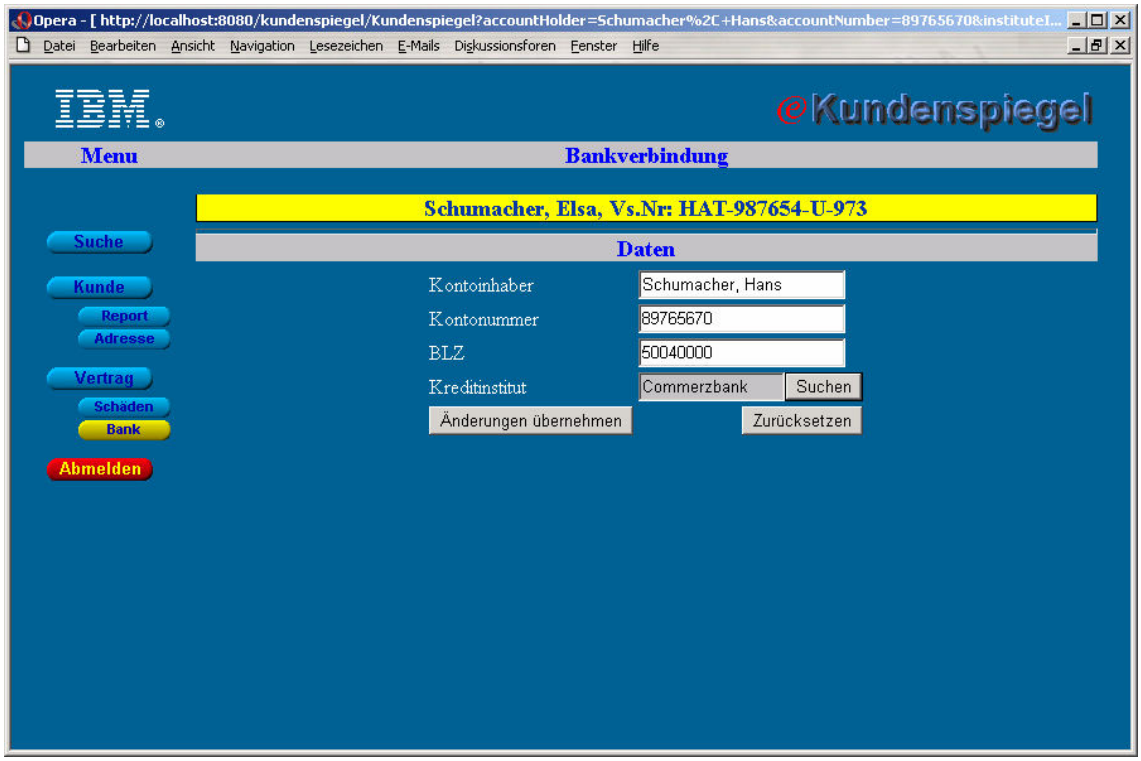


Abbildung Anhang A-0-10 UC 3 Vertragsdetails bearbeiten - Bankdaten bearbeiten

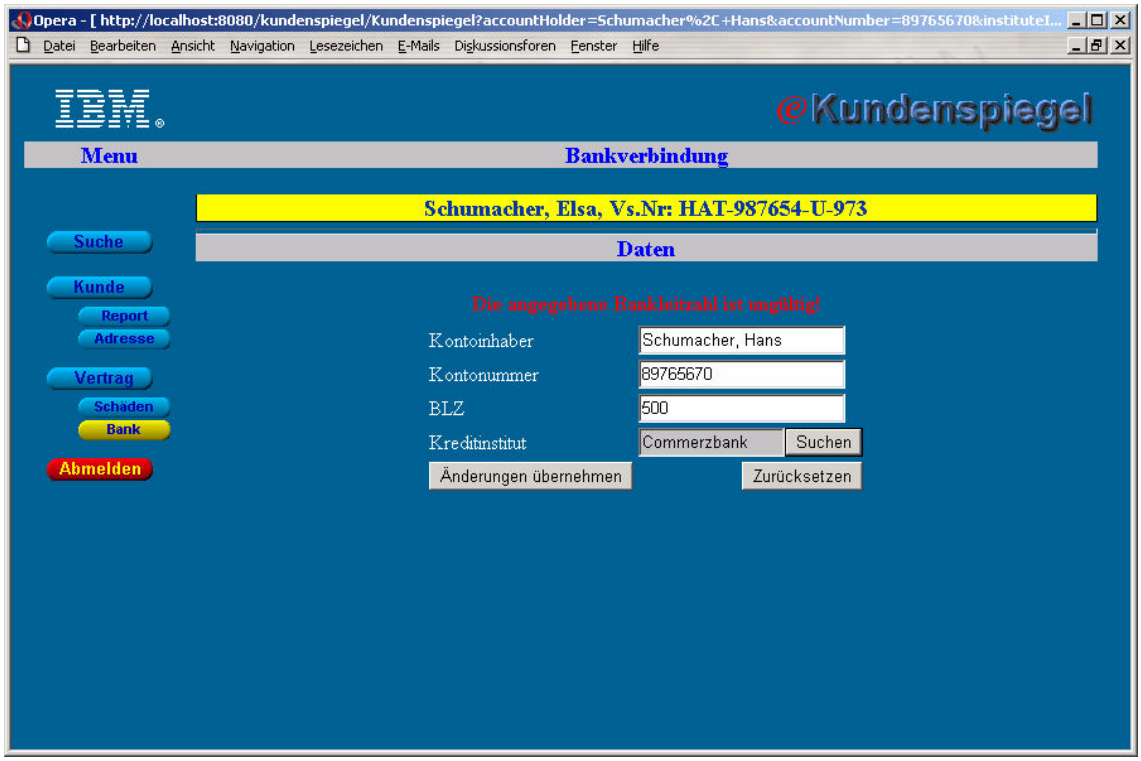


Abbildung Anhang A-0-11 UC 3 Vertragsdetails bearbeiten-Bankdaten bearbeiten, Fehlerfall

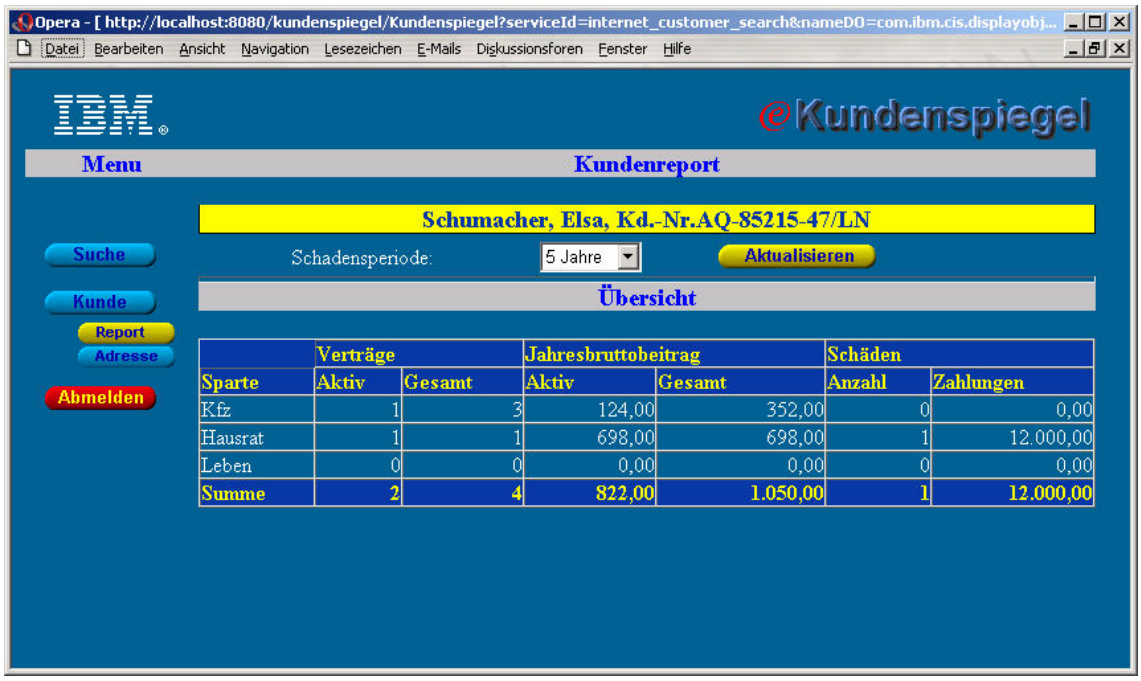


Abbildung Anhang A-0-12 UC 4 Kundenreport einsehen

Die Schadensperiode kann begrenzt bzw. erweitert werden.

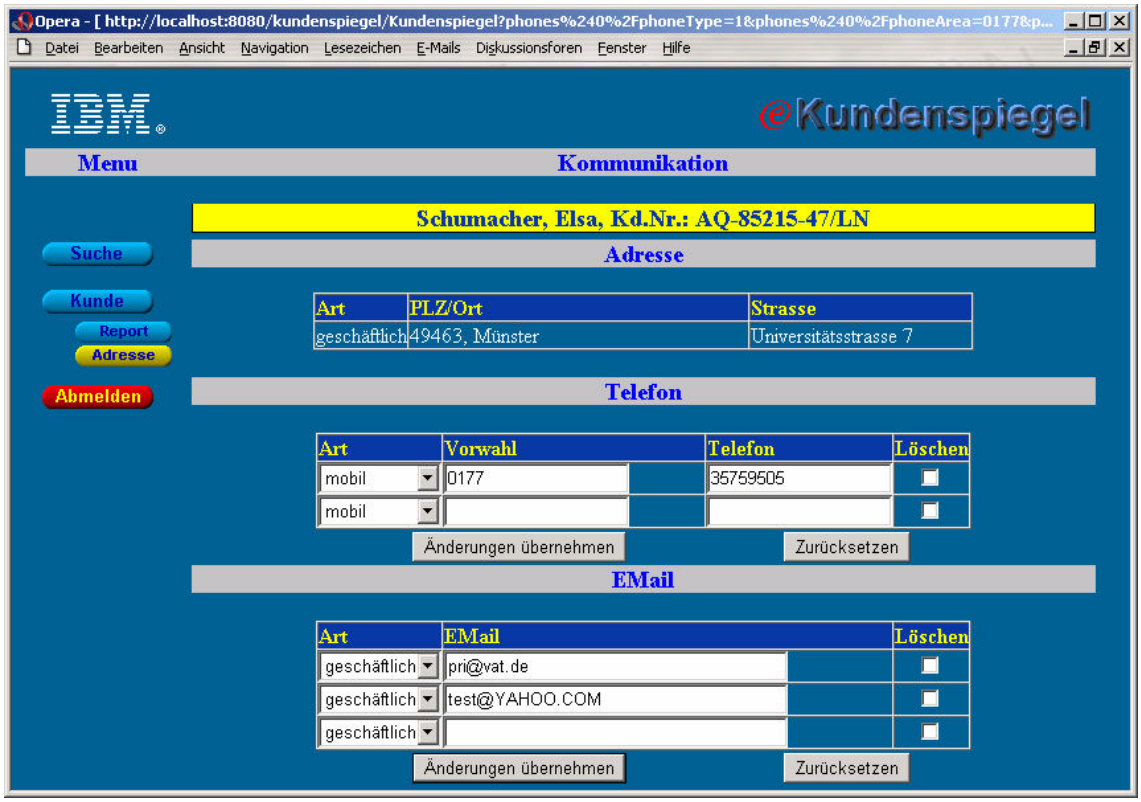


Abbildung Anhang A-0-13 UC 5 Kommunikationsdaten bearbeiten

Anhang B Protoyp im Portalserver

8.9 Erläuterung der Lösungsbewertung

Variante	Komplexität	Wiederver-wendung	Wartbarkeit	Konfiguration	Nutzen der Portlet-API
1	5	5	1(a)	1 (b)	1 (c)
2	1(d)	3(e)	1(a)	1 (b)	5
3	5	5	5	5	1(f)
4	5	5	5	5	1(c)
5	3(g)	5	5	5	5

Abbildung 0-1 Erläuterung der Lösungsbewertung

(a) Es existieren mindestens zwei Frameworks – eins im Applikationserver und eins je Portlet

(b) Das Administrationsservlet wird nicht unterstützt

(c) Die Portlet-API wird nicht verwendet.

(d) In zahlreichen Klassen des Frameworks wird die HTTP-Session gesichert. Die Session der Portlet-API entspricht nicht die der Servlet-API, dementsprechend müssen die Klassen des Frameworks angepasst werden und Methoden die in der Portlet-API nicht zur Verfügung stehen (z.B. PortletSession.isNew()) umgangen werden

(e) Die projektabhängigen Klassen, Teile der Servlet-API kapseln müssen wie unter (d) angepasst werden.

(f) Die Portlet-API wird nicht verwendet – es findet keine direkte Interaktion zwischen Portal und Anwendung im Applikationserver statt.

(g) geringe Teile des Frameworks müssen geändert werden, damit die Ausgabe entsprechend des Portalservers aufbereitet werden.

8.10 Quellcode Portlet

```
import java.io.*;
import java.net.*;

import java.util.Enumeration;
import java.util.StringTokenizer;
import javax.servlet.http.*;
import org.apache.jetspeed.portlet.*;
import
org.apache.jetspeed.portletcontainer.Por
tletRequestImpl;
import
org.apache.jetspeed.portletcontainer.Por
tletResponseImpl;
import
org.apache.jetspeed.portlets.AbstractPor
tlet;
import
org.apache.jetspeed.portlet.service.*;
// import com.ibm.wps.sso.*;
// import
com.ibm.wps.portletservice.credentialvau
lt.CredentialVaultService;

public class EvolutionController extends
AbstractPortlet
{

public void init(PortletConfig
portletconfig)    throws
UnavailableException {

               serverURL =
portletconfig.getAttribute("URL");
        super.init(portletconfig);
}


public void setParameter(String key,
String value){
               if (this.parameter
==null)
                       this.parameter =
new java.util.Hashtable();
               this.parameter.put(key,
value);
}

public void
setDefaultValues(PortletSession
session){

        setParameter(USER_NAME, "aaa");
        setParameter(PASSWORD, "aaa");


        // Zugangsdaten in der Datenbank
müssen noch dem Portal angeglichen
werden.
        // setParameter(USERNAME,
session.getAttribute("uid"));
        // setParameter(PASSWORT,
session.getAttribute("userpassword"));

        setParameter(NAME_DO,
        getConfig().getAttribute("NAME_DI
SPLAY_OBJECT"));
        setParameter(SERVICE_ID,
        getConfig().getAttribute("SERVICE
_ID"));
        setParameter(ACTION_NAME,
        getConfig().getAttribute("ACTON_N
AME"));


}

public String getParameterAsString(){
        String APPEND = "&";
        String CONNECT = "=";

        StringBuffer sBuffer = new
StringBuffer("?");
        boolean first = true;

        Enumeration e =
this.parameter.keys();

        while (e.hasMoreElements()){
                String alias = (String)
e.nextElement();

                if (first)
                        first = false;
                else

        sBuffer.append(APPEND);

                sBuffer.append(alias);
                sBuffer.append(CONNECT);

        sBuffer.append(this.parameter.get
(alias));
        }
         return sBuffer.toString();
}



public String
getInputStream(HttpURLConnection con){
        StringBuffer tmp = new
StringBuffer();
        String inputLine;
        try {
                BufferedReader in = new
BufferedReader(
                                    new
InputStreamReader(

con.getInputStream()));

                 while ((inputLine =
in.readLine()) != null)
                tmp.append(inputLine);

        in.close();
        }
                catch (IOException io){

                return null;
        }
        //
portletlog.info(this.getClass().getName(
) + ".getInputStream:" +
tmp.toString());
        return tmp.toString();

 }

public void login(PortletRequest
request){
}

public void logout(PortletSession
session){

}

public void service(PortletRequest
portletrequest, PortletResponse
portletresponse)
        throws IOException,
PortletException {
        try
        {
            portletlog =
getPortletLog();


         PortletWindow window  =
portletrequest.getWindow();
                String windowState =
"default";
                if (window.isDetached())
windowState = "detached";
                if (window.isMaximized())
windowState = "max";
                if (window.isMinimized())
windowState = "min";
```

```
          pSession =
portletrequest.getSession(true);

        String returnURI =
portletresponse.createURI().toString();


               request =
((PortletRequestImpl)portletrequest).get
ServletRequest();
               response =
(HttpServletResponse)((PortletResponseIm
pl)portletresponse).getServletResponse()
;

            if
(pSession.getAttribute("cookie") ==
null){

       portletlog.info("pSession.getAttr
ibute cookie" +
pSession.getAttribute("cookie") + " ist
= null");
                        // default values
für ersten aufruf

       //out.print("<b>setze
defaultParameter um wieder auf
kundensuche zu kommen<b>")  ;


       setDefaultValues(pSession);


            }


            String key;
            // StringBuffer tmp = new
StringBuffer();

            PrintWriter out =
response.getWriter();
            //
**************************************
***
            // EIGENTLICHER Durchlauf

//**************************************
*******
       serverSessionID = (String)
pSession.getAttribute("cookie");



       portletlog.info(this.getClass().g
etName() + "######### ....Auslesen der
Requests ##############");

       String dencodeNamespace =
portletresponse.encodeNamespace("");

       portletlog.error("DecodeParameter
: " +dencodeNamespace);

        for(Enumeration e =
request.getParameterNames();
e.hasMoreElements();
portletlog.info(this.getClass().getName(
) + "Request-Parameter: " + key + "=" +
request.getParameter(key)))
            {
                 key =
(String)e.nextElement();
               // tmp.append("Request-
Parameter: " + key + "=" +
request.getParameter(key));
                // Werte für die
weiteren Durchläufe
                if (serverSessionID !=
null){

                        // mappen des
Portlet auf HTTP Request
                /*    if
(key.startsWith(dencodeNamespace)){
                         // key ohne den
WPSParameter aufbereiten

       setParameter( (new
StringBuffer(key).delete(0,dencodeNamesp
ace.length())).toString(),
request.getParameter(key));
                        }else{
                         */
       setParameter( key,
request.getParameter(key));
                        //}
                }
            }
                // URL erzeugen

       URL url = new URL (serverURL +
getParameterAsString());



        con = (HttpURLConnection)
url.openConnection();

        // Parameter für jeden
Frameworkaufruf setzen

        con.setRequestProperty("user-
agent","wps2.1");

con.setRequestProperty("wps_imagePath",
portletresponse.encodeURI("/"));

con.setRequestProperty("wps_inputPath",
portletresponse.encodeNamespace(""));

con.setRequestProperty("wps_windowState"
, windowState);


       // falls cookie parameter
gesetzt, dem Request hinzufügen,
       // um wieder in die richtige
Session des Servers zu gelangen

            if
(pSession.getAttribute("cookie")!=
null){

       con.setRequestProperty("cookie",
(String)
pSession.getAttribute("cookie"));
            }


        // server auslesen
         String serverOutput =
getInputStream(con);


       // Falls Angrage nicht
erfolgreich
       if (con.getResponseCode() !=
HttpURLConnection.HTTP_OK){


       pSession.removeAttribute("cookie"
);
               throw new
UnavailableException( "Anwendung ist zur
Zeit nicht erreichbar! - mach doch mal
ne Pause" );
       }

       // falls Anfrage erfolgreich
bearbeitet.
        if (con.getResponseCode() ==
HttpURLConnection.HTTP_OK &&
serverOutput !=null){

              // falls erster aufruf
              // nach erstem
Verbindungsaufbau nutzen der Session
ermöglichen

       if(pSession.getAttribute("cookie"
) ==null){

                       //zurück kommt
set-cookie
```

```
        pSession.setAttribute("cookie",
con.getHeaderField(SESSION_ID));

        serverSessionID=con.getHeaderFiel
d(SESSION_ID);
                }

                 // Falls neue Sessionid
vom Server kommt
                // sicherstellen, das
beim nächsten Durchlauf mit der
Startseite begonnen wird.
                // dazu wird das cookie
Attribut aus der Session entfernt
                // und dadurch beim
nächsten durchlauf die defaultvalues
verwendet
                if
(con.getHeaderField(SESSION_ID) !=null
&& !(serverSessionID.equals(
con.getHeaderField(SESSION_ID)))){

        pSession.removeAttribute("cookie"
);
                 }

                // Ausgabe der Daten

out.println(serverOutput);

        }else{

                // Anfrage wurde nicht
erfolgreich beantwortet
                // cookie-Attribut aus
der Session nehmen, um nächsten
Durchlauf mit dem defaultvalues zu
starten

        pSession.setAttribute("cookie",
null);

        }

          response.flushBuffer();
         }
         catch (Exception e){

            portletlog.error("Exception
wurde geworfen ... controller wird
beendet " + e.getMessage());

        pSession.removeAttribute("cookie"
);
                //
        portletlog.error(getPortletSessio
nInfo(pSession, "\n"));
                 throw new
UnavailableException( "It is time to say
goodby..");
         }
    }
/* private String[]
getPrincipalsFromSubject(Subject
subject, String className)

                throws PortletException {
        try {
                // Get the Set of
Principals
                Object[] principals =
        subject.getPrincipals(Class.forNa
me(className)).toArray();

                // Do we have any?
                if ((null == principals)
|| (0 == principals.length)) {
                // No principals of this
class name
                        return(null);
                }

                // Create the String
Array to hold the values
                String[] values = new
String[principals.length];

                // Populate the values
Array
                for (int current = 0;
current < values.length; current++) {
                        values[current] =
((Principal)
principals[current]).getName();
                }
                return(values);
        } catch (ClassNotFoundException
cnfe) {
                throw(new
PortletException("Class " + className +
" could not be found",cnfe));
                }


                }

                 */

        private HttpServletRequest
request;
        private HttpServletResponse
response;
        private
java.net.HttpURLConnection con;
        private PortletLog portletlog;

        private String serverURL = null;
         // Hashtable zur
Parameterübergabe ans Framework
        private java.util.Hashtable
parameter;
        private String serverSessionID =
null;

        private PortletSession pSession;

        // Konstanten
        public static final String
SESSION_ID = "Set-Cookie";
        public static final String
DEFAULT_URL =
"http://localhost/test.html";
        public static final String
NAME_DO = "nameDO";
        public static final String
ACTION_NAME = "actionName";
        public static final String
SERVICE_ID = "serviceId";
        public static final String
USER_NAME = "userName";
        public static final String
PASSWORD = "password";

}
```

8.11 Codeausschnitte Frameworkänderungen

8.11.1 Anpassung Anzeigeobjekt

```
public abstract class DisplayObject {
        …
        private java.lang.String wpsImagePath;
```

```
	public void setWpsImagePath(String newWpsImagePath) {
		wpsImagePath = newWpsImagePath;
	}
}
```

8.11.2 Anpassungen des Steuerungsservlet

1. **wps_ImagePath den Anzeigeobjekten**

```
	if (request.getHeader("user-agent").equalsIgnoreCase("WPS2.1")){
	   process.getDisplayObject().setWpsImagePath(
	                        request.getHeader("wps_imagepath"));
	}
```

2.) **Dem wpsWindowState vor der XSLT-Transformation mitteilen, um den Pfad zum XSL-Dokument festzulegen**

```
if (request.getHeader("user-agent").equalsIgnoreCase("WPS2.1")){
   htmlGenerator.setXslDocument(
                process.getDisplayObject().getXslRoot()
                + request.getHeader("wps_windowstate")
                + "/"
                + process.getDisplayObject().getXslPath());
}else{ //vorheriger Code
   htmlGenerator.setXslDocument(
                 process.getDisplayObject().getXslRoot()
               + process.getDisplayObject().getXslPath());
}
```

3.) **Mitteilen eines Fehlerfalls**

```
private void createErrorPage(HttpServletRequest req, HttpServletResponse
res, String serviceId) throws java.lang.Exception {

   StringBuffer errorPage = new StringBuffer();
   errorPage.append("<html xmlns=\"http://www.w3.org/1999/xhtml\">");
   …
   errorPage.append("</html>");

   res.getWriter().print(errorPage.toString());


if (request.getHeader("agent").equalsIgnoreCase("WPS2.1")){
   res.sendError(res.SC_CONFLICT);
}

}
```

8.12 Screnshots

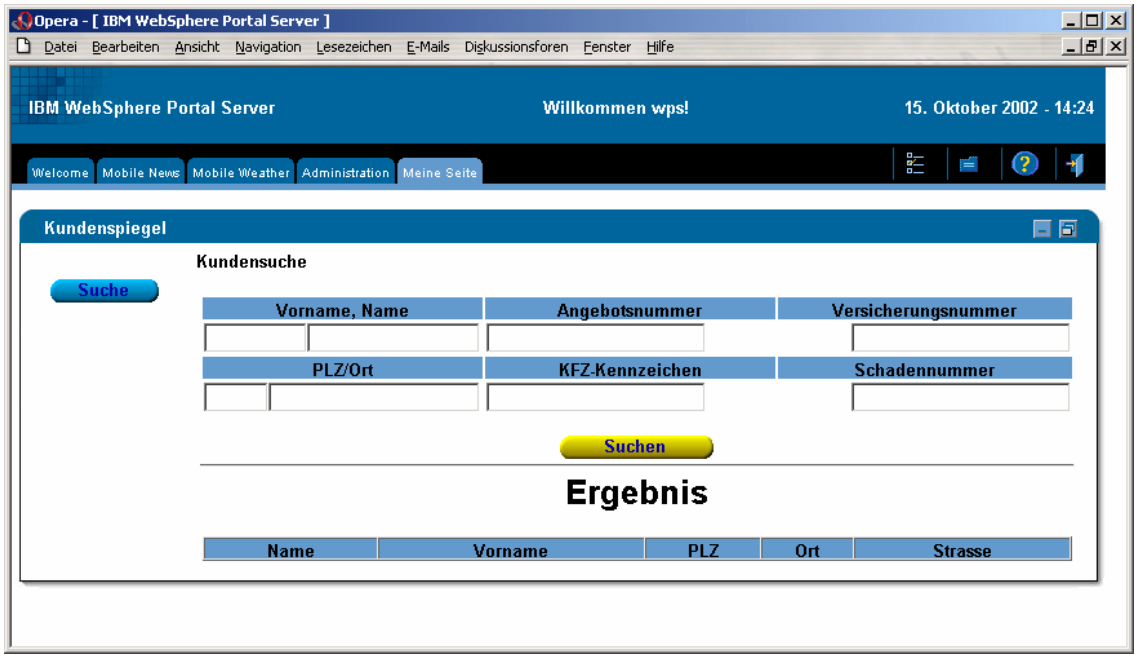


Abbildung Anhang B-0-2 UC 1 Kundensuche durchführen -Start Suche

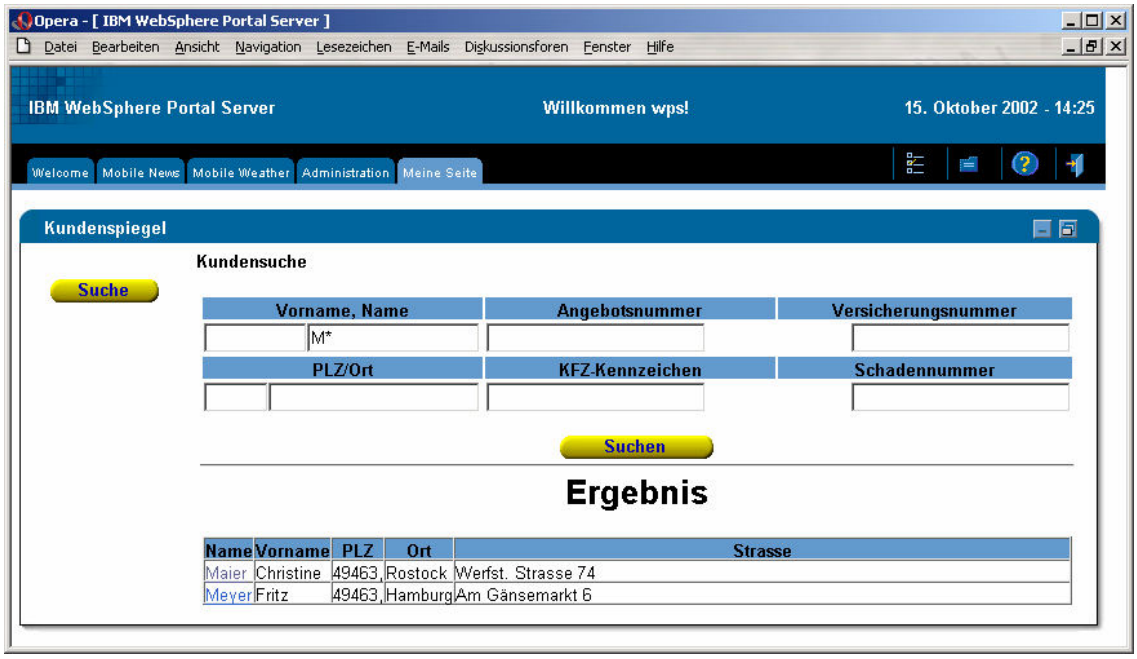


Abbildung Anhang B-0-3 UC 1 Kundensuche druchführen -Ergebnisliste

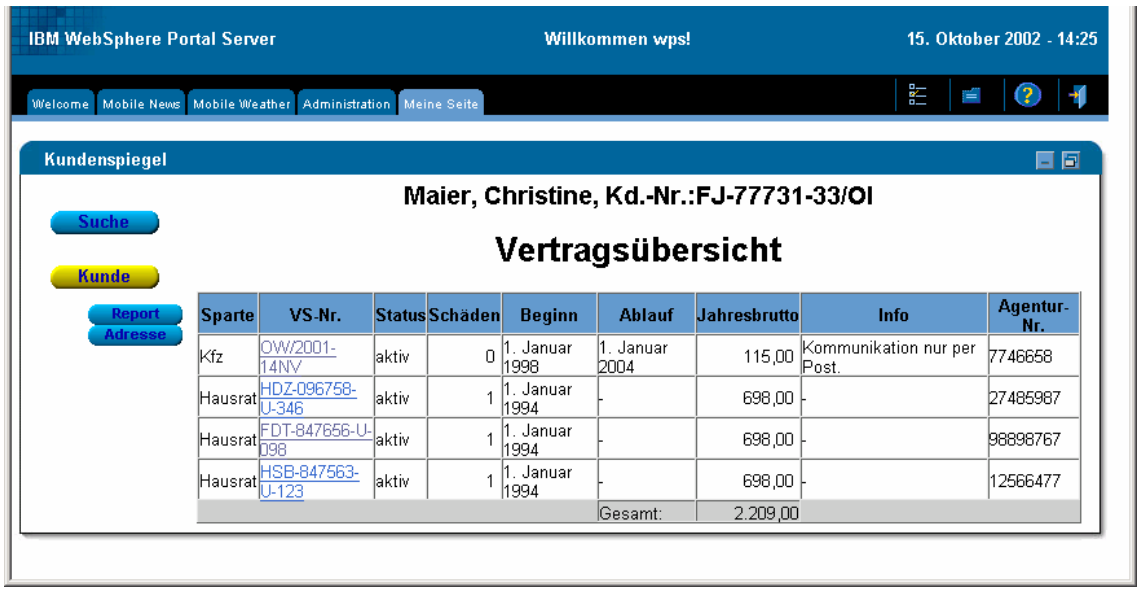


Abbildung Anhang B-0-4 UC 2 Kundenübersicht einsehen

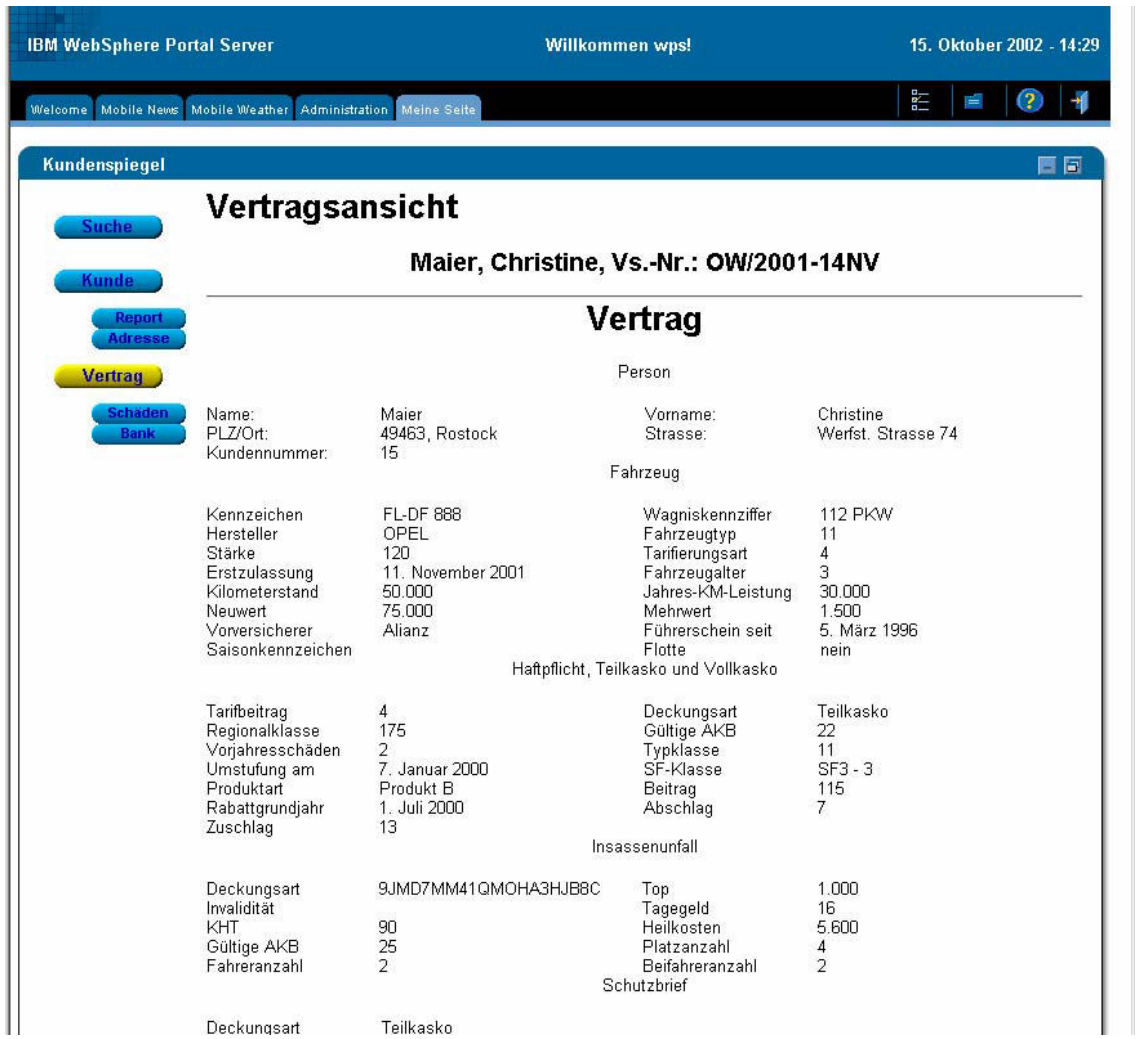


Abbildung Anhang B-0-5 UC 3 Vertragsdetails bearbeiten

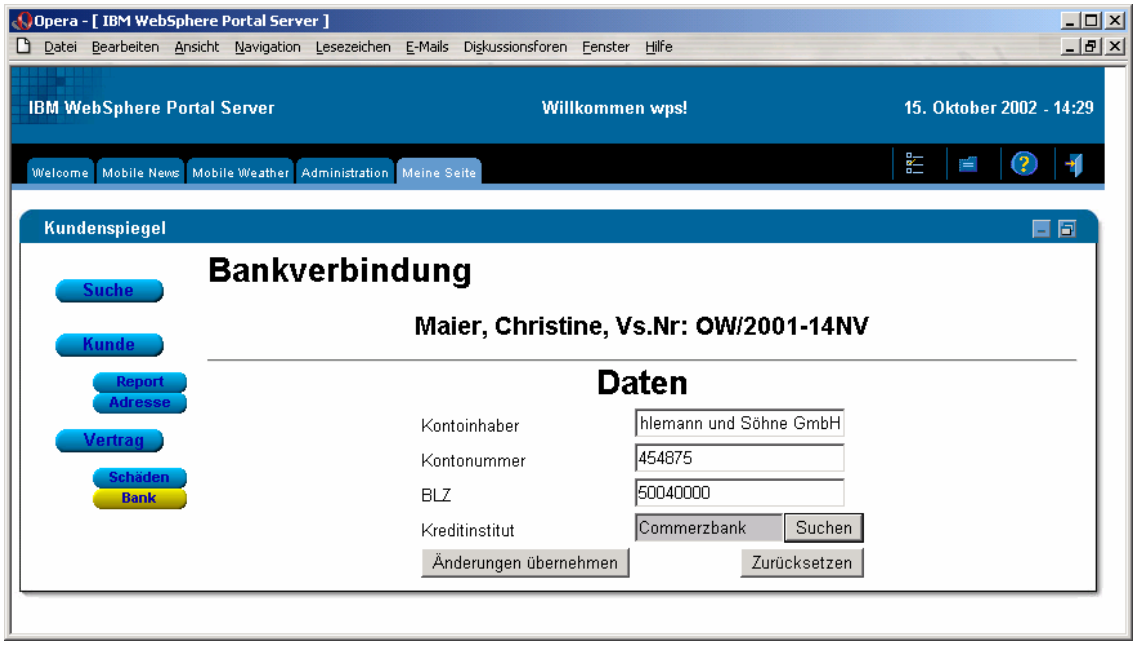


Abbildung Anhang B-0-6 UC 3 Vertragsdetails bearbeiten – Bankdaten

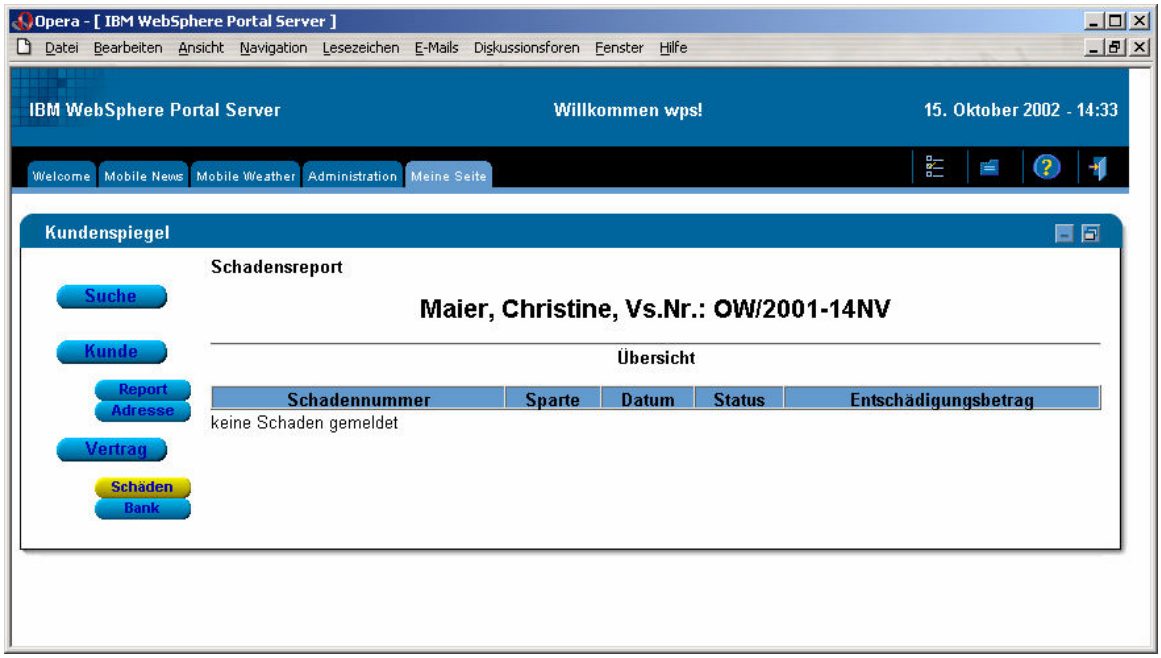


Abbildung Anhang B-0-7 UC 3 Vertragsdetails bearbeiten – Schadenliste

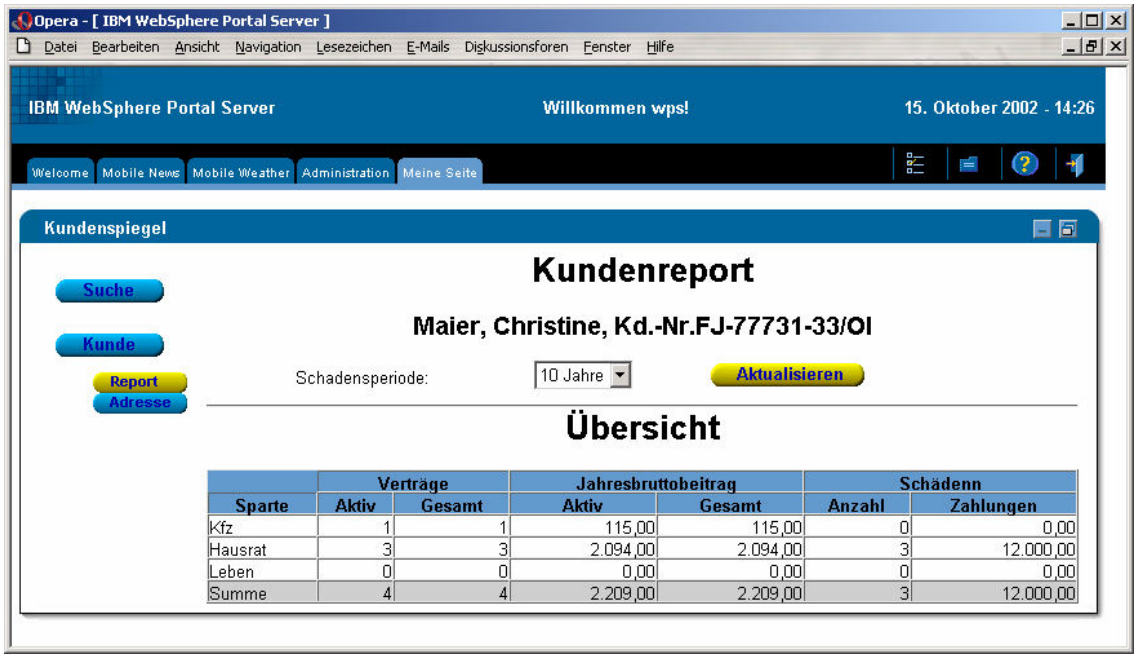


Abbildung Anhang B-0-8 UC 4 Kundenreport einsehen

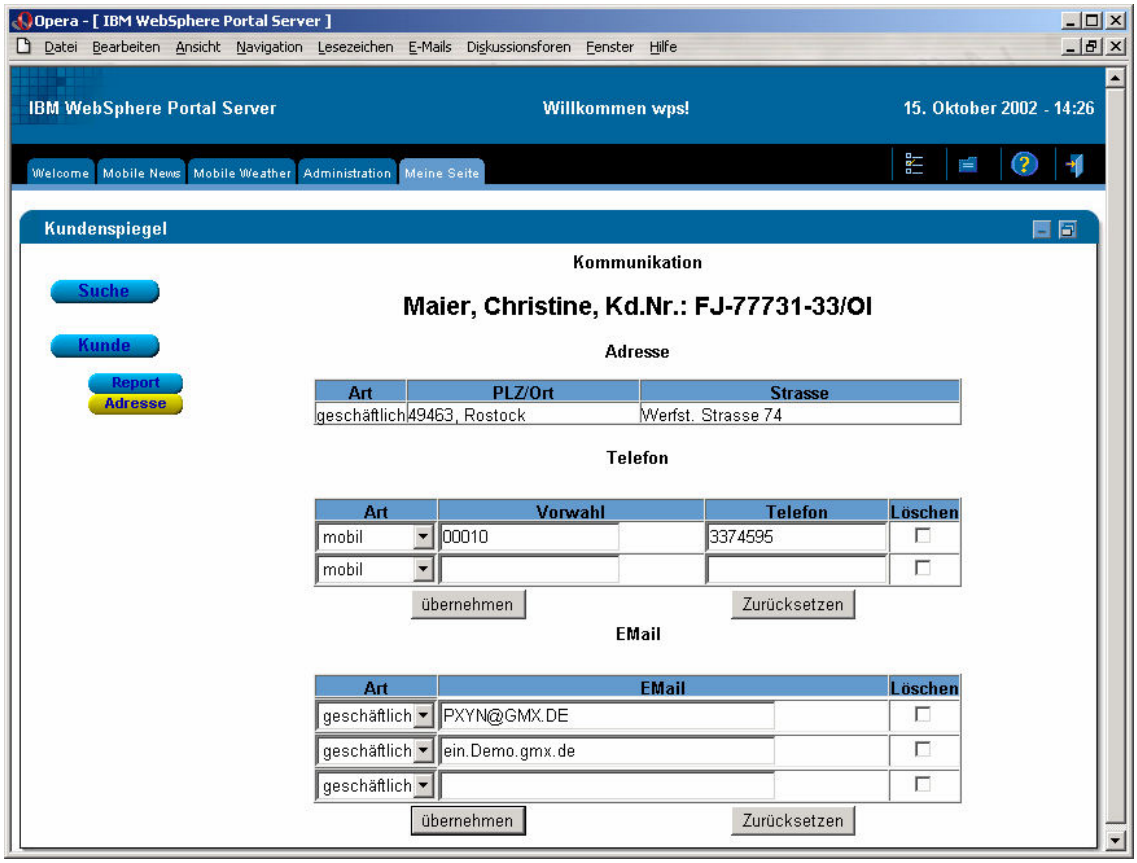


Abbildung Anhang B-UC 5 Kommunikationsdaten bearbeiten

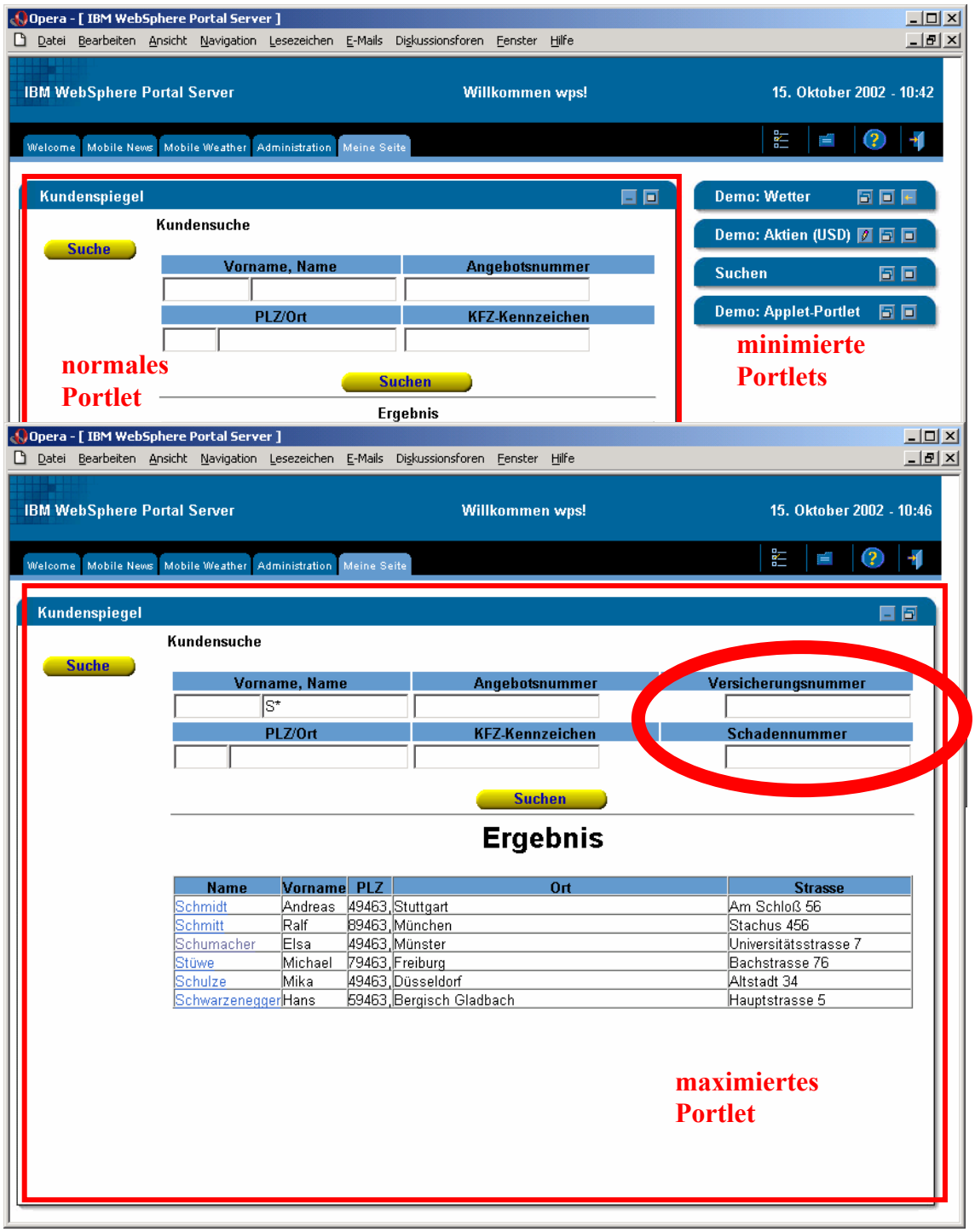


Abbildung Anhang B-0-9 Verwendung Portletmodus im Prototyp

Anhang C Portalserver

8.13 Personalisierung einer Seite

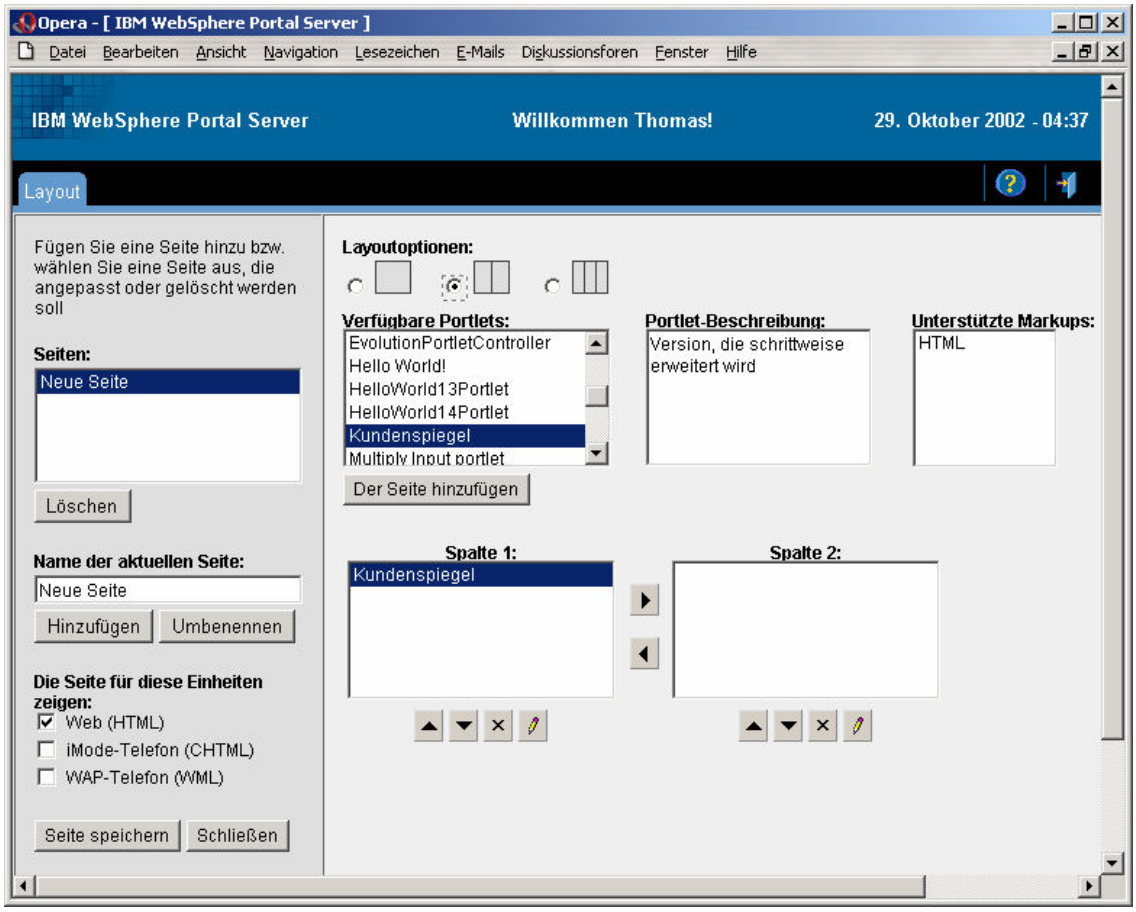


Abbildung Anhang C-0-1 Personalisierung einer Seite

„Thomas“ personalisiert sich eine Seite „Neue Seite“ im Portal.
Er wählt die Layoutoptionen aus (Anzahl der Spalten) und wählt ein Portlet (Kundenspiegel) aus seinen verfügbaren aus.

Die Portlet-Beschreibung sowie die unterstützen Markups werden aus der Datei Portlet.xml verwendet. (Hierbei handelt es sich für die Einstellungen, die für jedes Portlet der Porlet-Applikation vorgenommen werden).

8.14 Installation eines Portlets

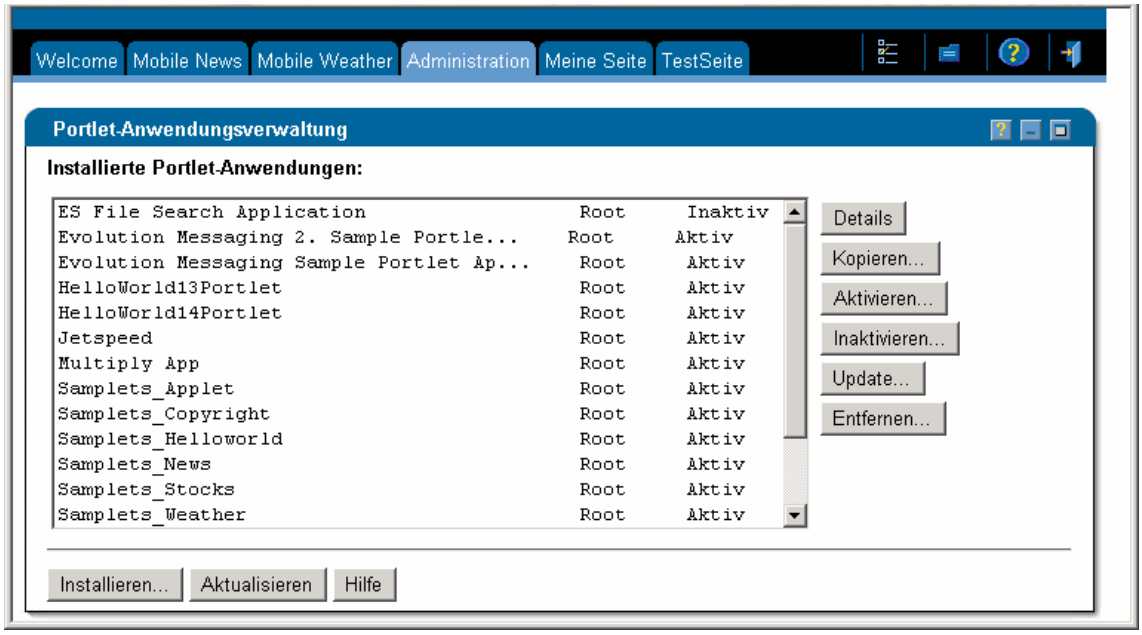


Abbildung Anhang C-0-2 Portletinstallation

Um ein neues Portlet einem Portal anzuzeigen sind folgende Aktionen notwendig:

Akteur	Aktion
Administrator(A)	Ruft Administrationsseite auf.(siehe oben)
System (S)	Zeigt Administrationsseite an
A	Wählt „Installieren…“
S	Zeigt Maske zur Eingabe des Par-Files
A	Wählt Par-File aus
S	Erzeugt Sicherheitsabfrage, ob das Portlet installiert werden soll,
A	Wählt „Fortsetzen“
S	Zeigt Ergebnis der Installation
A	Wählt „Beenden“
S	Zeigt Portlet (siehe oben)
A	selektiert das Portlet in der Liste (durch anklicken) und wählt „Aktivieren“
S	Erzeugt Sicherheitsseite, ob das Portlet aktiviert wird.
A	Wählt „Fortsetzen“
A	Vergibt einem Benutzer bzw. einer Benutzergruppe View Rechte. (vgl. Abbildung 3-11 Zugriffsverwaltung)

Tabelle 10Interaktionsschritte Portletinstallation

8.15 Applikationszugriff im Portal

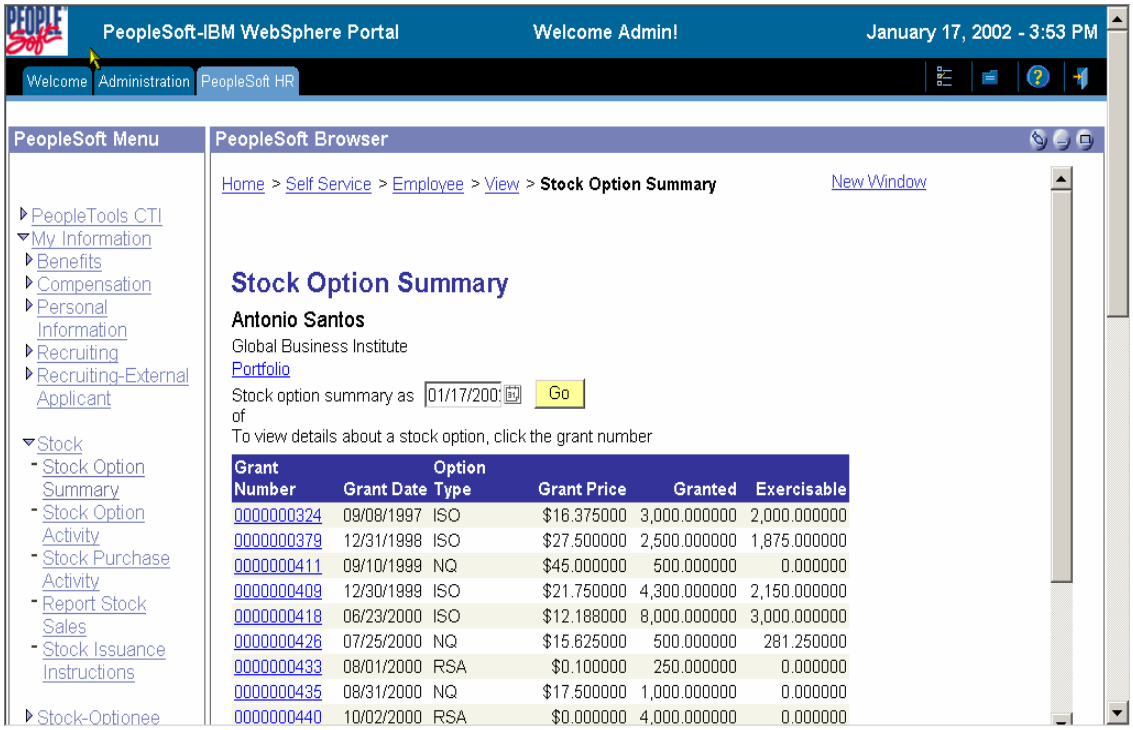


Abbildung Anhang C-0-3 PeopleSoft

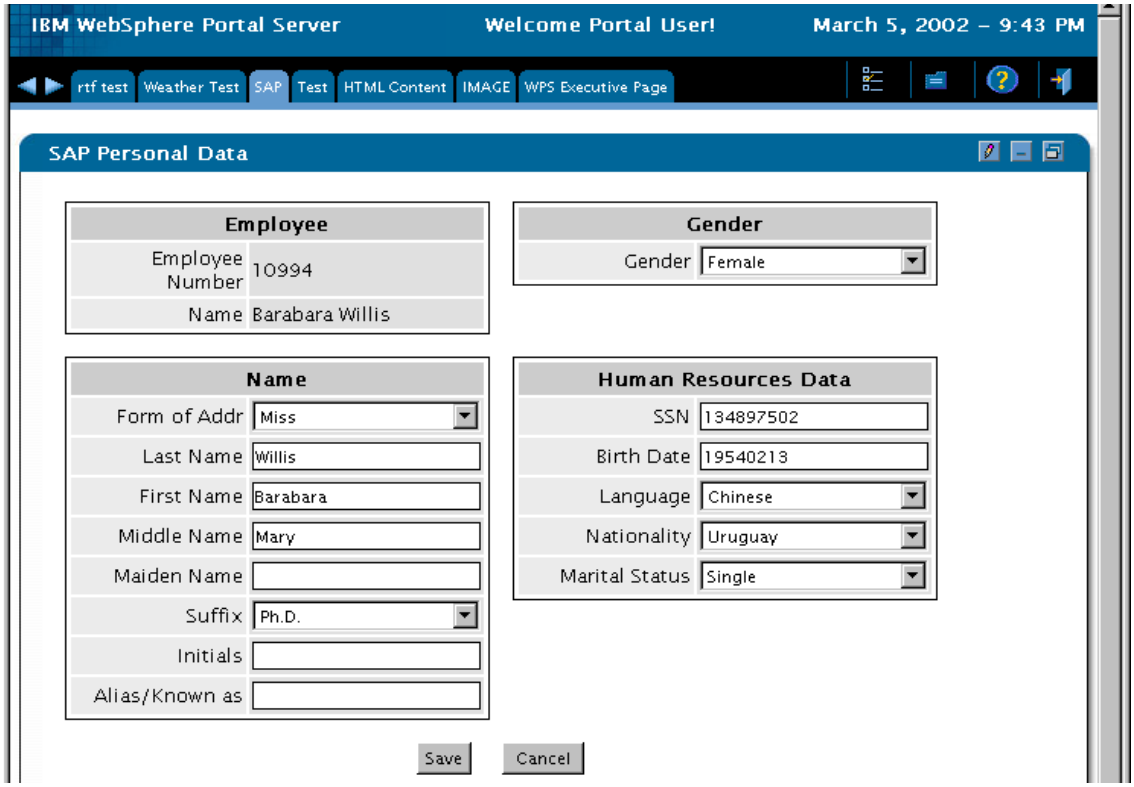


Abbildung Anhang C-0-4 SAP Personal Data

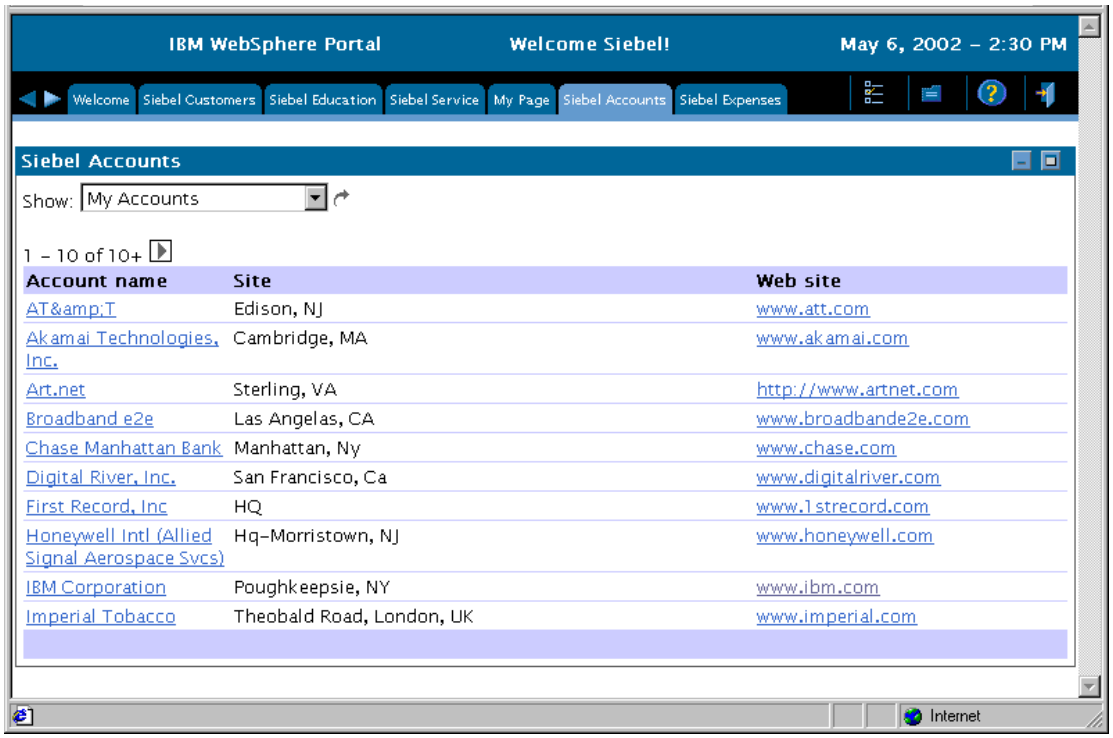


Abbildung Anhang C-0-5 Siebel Account

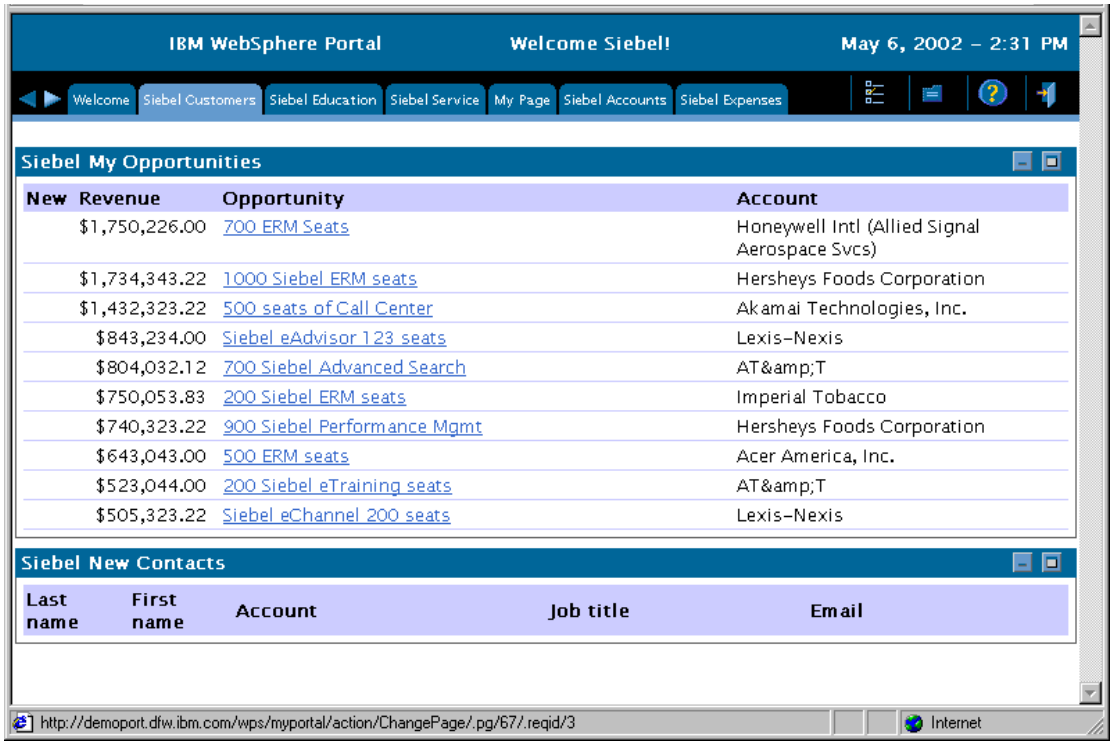


Abbildung Anhang C-0-6 Siebel My Opportunities

Anhang D

Glossar

Abstrakte Klasse (abstract class)	Von einer abstrakten Klasse können keine Objekte erzeugt werden.. Die abstrakte Klasse spielt eine wichtige Rolle in Vererbungsstrukturen, wo sie die Gemeinsamkeiten einer Gruppe von Unterklassen definiert. Damit eine abstrakte Klasse verwendet werden kann, muss von ihr zunächst eine Unterklasse abgeleitet werden. Alle Operationen können – wie auch bei einer konkreten Klasse – vollständig spezifiziert bzw. implementiert werden. Es ist jedoch nicht beabsichtigt, von dieser Klasse Objekt zu erzeugen.
Aggregation (aggregation)	Eine Aggregation ist ein Sonderfall der Assoziation. Sie liegt dann vor, wenn zwischen den Objekten der beteiligten Klassen eine Beziehung vorliegt, die sich als „ist Teil von“ oder „besteht aus“ beschreiben lässt.
Akteur (actor)	Ein Akteur ist eine Rolle, die ein Benutzer des Systems spielt. Akteure befinden sich außerhalb des Systems. Akteure können Personen oder externe Systeme sein.
Aktivitätsdiagramm (activity diagram)	Ein Aktivitätsdiagramm ist der Sonderfall eines Zustandsdiagramms, bei dem – fast – alle Zustände mit einer Verarbeitung verbunden sind. Ein Zustand wird verlassen, wenn die Verarbeitung beendet. ist. Außerdem ist es möglich, eine Verzweigung des Kontrollflusses zu spezifizieren und zu beschreiben, ob die Verarbeitungsschritte in festgelegter oder beliebiger Reihenfolge ausgeführt werden können.
Analyse (analysis)	Aufgabe der Analyse ist die Ermittlung und Beschreibung der Anforderungen eines Auftraggebers an ein Softwaresystem. Das Ergebnis soll die Anforderungen vollständig, widerspruchsfrei, eindeutig, präzise und verständlich beschreiben.
Anzeigeobjekte(Evolution)	Anzeigeobjekte enhalten die Nutzdaten, die zwischen Client und Server ausgetauscht werden. Ein Anzeigeobjekt können weitere Anzeigeobjekte enthalten. Das Hauptanzeigeobjekte bezeichnet das Anzeigeobjekt, das alle weiteren Anzeigeobjekt enthalt. Es handelt sich um einen Container.
Assoziation (association)	Eine Assoziation modelliert Verbindungen zwischen Objekten einer oder mehrer Klassen. Binäre Assoziationen verbinden zwei Objekte. Eine Assoziation wird beschrieben durch Kardinalitäten und einen optionalen Assoziationsnamen Sie kann um Restriktionen ergänzt werden.

Attribut (attribute)	Attribute beschreiben Daten, die von den Objekten der Klasse angenommen werden können. Alle Objekte einer Klasse besitzen dieselben Attribute, jedoch im allgemeinen unterschiedliche Attributwerte. Jedes Attribut ist von einem bestimmten Typ und kann einen Anfangswert (default) besitzen. Bei der Implementierung muss jedes Objekt Speicherplatz für alle seine Attribute reservieren. Der Attributname ist innerhalb der Klasse eindeutig. Abgeleitete Attribute lassen sich aus anderen Attributen berechnen.
Attributlisten (XML)	Attributlisten definieren, welches Attribut mit welchen Elementen verbunden werden darf. Für jedes Attribut legen Sie einen Namen, einen Typ und eventuell einen Standardtyp fest und definieren, ob das Attribut optional (#IMPLIED), vorgeschrieben (#REQUIRED) oder festgelegt (#FIXED) ist.
cHTML	Compact Hypertext Markup Language, basiert auf HTML und ist spezialisiert auf Kleingeräte wie PDAs etc.
Container	Ein Container ist eine Klasse, deren Objekte Mengen von Objekten (anderer) Klassen sind.
CORBA (Common Reques Broker Architecture)	CORBA ist der OMG-Standard, der spezifiziert, wie Objekte in einer verteilten, heterogenen Umgebung kommunizieren. Er beschreibt den Aufbau des ORB, seine Bestanteile sowie deren Verhalten und Schnittstellen.
Datenmodell	Jedem Datenbanksystem liegt ein Datenmodell zugrunde, in dem festgelegt wird, welche Eigenschaften und Struktur die Datenelemente besitzen dürfen, welche Konsistenzbedingungen einzuhalten sind und welche Operationen zum Speichern, Suche, Ändern und Löschen von Datenelementen existieren. Es lassen sich relationale und objektorientierte Datenmodell unterscheiden.
DDL (Data Definition Language)	Die Datendefinitionssprache ist eine Sprache, die ein relationales Datenbanksystem zur Verfügung stellt und die zur formalen Definition des logischen Schemas – d.h. den leeren Tabellen der relationalen Datenbank dient.
Deployment-deskriptor	Der Deploymentdeskriptor ist eine Datei (i.d.R. XML) über die Programme bzw. Applikationen wie Java Webanwendungen, Enterprise Java Beans oder Portlets etc. konfiguriert werden
DML (Data Manipulation Language)	Die Datenmanipulationssprache (DML) dient dazu, die leeren Tabellen einer Datenbank mit Daten zu füllen und diese Daten zu ändern.
Drei-Schichten-Architektur (three-tier architecture)	Die Drei-Schichten-architektur besteht aus der GUI-Schicht (Schicht der Benutzungsoberfläche), der Fachkonzeptschicht und der Schicht der Datenhaltung. Es sind zwei Ausprägungen möglich: die strenge und die flexible Drei-Schichten-Architektur.
Entwurf (design)	Aufgabe des Entwurfs ist – aufbauend auf dem Ergebnis der Analyse – die Erstellung der Softwarearchitektur und die Spezifikation der Komponenten, d.h. die Festlegung von deren Schnittstellen, Funktion- und Leistungsumfang. Das Ergebnis soll die zu realisierenden Programme auf einem höheren Abstraktionsniveau widerspiegeln.

Framework	Ein Framework besteht aus einer Menge von zusammenarbeitenden Klassen, die einen wiederverwendbaren Entwurf für einen bestimmten Anwendungsbereich implementieren. Es besteht aus konkreten und insbesondere aus abstrakten Klassen, die Schnittstellen definieren. Die abstrakten Klassen enthalten sowohl abstrakte als auch konkrete Operationen. Im Allgemeinen wird vom Anwender (= Programmierer) des Framworks erwartet, dass er Unterklassen definiert, um das Framework zu verwenden und anzupassen.
Geschäftsopjekt	Ein Geschäftsobjekt (GO) kapselt in Evolution das Model gemäßt dem MVC-Muster. Ein GO kann mehrere Backendsysteme und Datenbanken zugreifen.
Geschäftsprozeß	Ein Geschäftsprozeß besteht aus mehreren zusammenhängenden Aufgaben, die von einem Akteur durchgeführt werden, um ein Ziel zu erreichen bzw. ein gewünschtes Ergebnis zu erstellen.
I-Mode	I-Mode ist eine Mobilfunktechnik, mit der Daten wie Bilder und Spiele übertragen werden können. Im Gegensatz zu UMTS arbeitet sie mit bestehenden Mobilfunknetzen. Es sind allerdings spezielle Handys dafür erforderlich.
Instanz	Der Begriff Instanz zur bezeichnung eines Exemplars einer Klasse wurde aus dem Englischen übernommen (instance) und eingedeutscht.Objekt.
JCP	Java Community Process. Eine Vereinigung von mehr als 500 Firmen und weiteren Mitgliedern, die sich Java Standards beschliessen
Kardinalität (multiplicity)	Die Kardinalität bezeichnet die Wertigkeit einer Assoziation, d. h. sie spezifiziert die Anzahl der an der Assoziation beteiligten Objekte.
Klasse (class)	Eine Klasse definiert für eine Kollektion von Objekten deren Struktur (Attribute), das Verhalten (Operationen) und Beziehungen (Assoziationen, Vererbungsstrukturen.). Klassen besitzen – mit Ausnahme von abstrakten Klassen – einen Mechanismus, um neue Objekte zu erzeugen.
Klassendiagramm (class diagram)	Das Klassendiagramm stellt die Klassen, die Vererbung und die Assoziationen zwischen Klassen dar. Zusätzlich können Pakete modelliert werden.
Komponente	Eine Komponente stellt ein ausführbares Stück Software dar.
Markup	Als Markup werden alle Zeichen betrachtet, die nicht für den menschlichen Betrachter in den Text gesetzt werden, sondern für die Auswertung durch Programme vorgesehen sind.

MQSeries	MQSeries ist ein Softwareprodukt von IBM. MQSeries ermöglicht das Verteilen von Nachrichten zwischen verschiedenen Systemen und sorgt dafür, dass diese Nachrichten genau einmal an dem entsprechenden Zielsystem – selbst nach einem Stromausfall - ankommt. Die besondere Bedeutung für EAI liegt darin begründet, dass MQSeries auf vielen verschieden Plattformen verfügbar ist und dass viele Softwareprodukte eine MQSeries-Schnittstelle liefern. Typische Anwendungsgebiete von MQSeries sind Application Integration, e-business Integration, EDI, Integration von ERP in die Unternehmenslandschaft sowie die Integration über verschiedene Unternehmen. MQSeries setzt sich aus verschiedenen Produkten zusammen[103]. Sie enthalten Basisdienste und Tools zum Versenden der Nachricht und ermöglicht das Konvertieren der Nachrichten und stellen somit das gewünschte Zielformat zur Verfügung, beispielsweise die Konvertierung eines Proprietären Formates in das SAP-IDOC Format. Eine Anbindung an MQSeries bietet also die Möglichkeit viele Zielplattformen anzusprechen.
MVC-Muster (Model/View/ Controller)	MVCMuster besteht aus den drei Klassen, *Model*, *View* und *Controller.*Das Model-Objekt repräsentiert das Fachkonzeptobjekt. Oft gibt es mehrere Möglichkeiten, die fachlichen Daten zu präsentieren. Für jede Präsentation gibt es ein View-Objekt. Das Controller-Objekt bestimmt, wie die Benutzungsoberfläche auf Eingaben reagiert Jedes View-Objekt besitzt ein zugehöriges Controller-Objekt, das diese Darstellung mit der Eingabe verbindet Das impliziert, dass es zu jedem Model-Objekt eine beliebige Anzahl von Paaren (View, Controller) geben kann, jedoch mindestens eines.
Objektorientierte Analyse (object oriented analysis)	Ermittlung und Beschreibung des Anforderungen an ein Softwaresystem mittels objektorientierter Konzepte und Notationen. Das Ergebnis ist ein OOA-Modell.
Objektorientierter Entwurf (object oriented design)	Aufbauend auf dem OOA-Modell erfolgt die Erstellung der Softwarearchitektur und die Spezifikation der Klassen aus Sicht der Realisierung. Das Ergebnis ist das OOD-Modell, das ein Spiegelbild der objektorientierten Programme auf einem höheren Abstraktionsniveau bildet.
ODBC (Open Database Connectivity)	ODBC ist eine standartisierte Schnittstelle für den Datenzugriff. Sie wurde ursprünglich von Microsoft spezifiziert.
OMG (Object Management Group)	Systemanbieter und Anwender objektorientierter Technicken haben sich 1989 zur OMB zusammengeschlossen. Die OMG verfolgt das Ziel, Standards und Spezifikationen für verteilte objektorientierte Anwendungen zu schaffen.
ORB (Object Request Broker)	In verteilten Systemen wird die Kommunikation zwischen Klient und Server vom ORB durchgeführt.
Persistenz	Persistenz ist die Fähigkeit eines Objekts, über die Ausführungszeit eines Programms hinaus zu leben, d.h. die Daten diese Objekt bleiben auch nach Beendigung des Programms erhalten und stehen bei einem Neustart wieder zur Verfügung.

[103] MQSeries Messaging, Integrator und Worklow

Referenzimplementierung(RI)	Um sicher zu stellen, dass sich eine Spezifikation auch implemntieren läßt und die Kompatibilität der Implementation sichergestellt werden kann wird innerhalb des JCP u.a. die Referenzimplementierung erstellt
Request	Mit einem request fordert der Client ein Objekt auf dem Server zur Ausführung einer Operation auf.
Sequenzdiagramm (sequence diagram)	Ein Sequenzdiagramm besitzt zwei Dimensionen. Die Vertikale repräsentiert die Zeit und auf der Horizontalen werden die Objekte angetragen. In das Diagramm werden die Botschaften eingetragen, die zum Aktivieren der Operationen dienen.
UML	Unified Modeling Language, die von Booch, Rumbaugh und Jacobson bei der Rational Software Corporation entwickelt und 1997 von der OMG (Object management Group) als Standard akzeptiert wurde.
URN	Ein Uniform Ressource Name ist ein eindeutiger Name im Web-Umfeld, hinter dem sich im Gegensatz zu einer URL keine Website befinden muß.
W3C, World Wide Web Consortium	Ende 1994 gegründete Organisation, die es sich zur Aufgabe gemacht hat, die Standards für das Web zu entwickeln. Zu den Mitgliedern gehören hauptsächlich Universitäten und Firmen.
WAP, Wireless Application Protocol	Von Ericson, Motorola, Nokia und Uniwired Planet initierte Spezifikation für den Internet-Zugriff mit Handys oder PDAs. WAP definiert u.a. Eckwerte für sogenannte Micro-Browser, mit denen Web-Inhalte auf dem Handy-Display dargestellt werden.
Web Application Archive (WAR)	Ein Web Application Archive (WAR) ist eine Datei die mit einem jar-tool erstellte Datei. Der Aufbau dieser Datei ist seit der Java-Servlet-API 2.2 vorgegeben und enthält verschiedene Bestandteile von Webapplikationen (HTML, Servlets, Java Server Pages) in gepackter Form.
WML, Wireless Markup Language	Für die Darstellung von Internetseiten auf Handys oder PDAs entwickelte Auszeichnungssprache. WML ist durch eine XML-DTD definiert.
XML-Namespace	Um Namenskonflikte zwischen den eingesetzten DTDs zu vermeiden, spezifiziert der W3C Standard XML-Namespace.Namensräume werden in einem XML-Dokument durch Zuweisen einers Uniform Ressource Name (URN) an ein frei wählbares Kürzel definiert.
Xpath	Die XML Path Language ist eine Sprache um Teile eines XML-Dokumentes zu adressieren und ermöglicht die Selektion sowohl von einzelnen Knoten als auch von ganzen Bereichen.
Zustand (Evolution)	Innerhalb von Evolutioin dienen Zustände als Übergang zwischen Aktivitäten. Zusammen mit den Anzeigeobjekten bestimmen die Zustände die zu verwendende XSL-Datei zur Transformation der Daten

Quellenverzeichnis

Literaturquellen

[Arci01] F. Arciniegas: XML Developer’s Guide, Franzis, Poing, 2001

[Bach00] M. Bach: XSL und Xpath – verständlich und praxisnah, Addison Wesley, München, 2000

[Balz99] Heide Balzert: Lehrbuch der Objektmodellierung, Analyse und Entwurf, Spektrum, Berlin, 1999

[BKR00] J. Becker, M. Kugeler, M. Rosemann: Prozessmanagement – Ein Leitfaden zur prozessorientierten Organisationsgestaltung, Springer Verlag 2000, 2. Auflage

[Coad90] P.Coad, E. Yourdan, Object-Oriented Analysis, Yourdon Press, 1990

[DP00] S. Denninger, I. Peters: Enterprise Java Beans, Addison Wesley, München 2000

[Fowl99] M. Fowler: Analysemuser - Wiederverwendbare Objektmodelle, Addison Wesley, Bonn, 1999

[Gamm96] E. Gamma, R. Helmm R. Johnson, J. Vlissides: Entwurfsmuster Elemente Wiederverwendbarer objektorientierter Software, Addison Wesley, Bonn, 1996

[GT00] V. Gruhn, A. Thiel: Komponentenmodelle: DCOM, JavaBeans, Enterprise Java Beans, CORBA, Addison-Wesley, München, 2000

[HC94] M. Hammer, J. Champy : Business Reengineering – Die Radikalkur für das Unternehmen, Campus Verlag, Frankfurt, 1994

[JBR99] I. Jacobson, G.Booch, J.Rumaugh: The Unified Software Developement Process, Addison Wesley, Reading, MA, 1999

[Krue02] G. Krueger, Handbuch der Javaprogrammierung, Addison-Wesley, Bonn, 2002

[Lint00] D. S. Linthicum : Enterprise Application Integration, Addison-Wesley,2000, 3rd printing

[MP02] S. Mienert, K. Popp, Enterprise-Portale & Enterprise Application Integration, dpunkt.verlag, 2002

[NN00] E. und M Niedermair: Internetprogrammierung mit Java, Data Becker, Düsseldorf 2000

[Oest98] B. Oestereich: Objektorientierte Softwareentwicklung, Analyse und Design mit der Unified Modeling Language – 4 aktualisierte Auflage,Oldenbourg, München 1998

[Oest99] B. Oestereich, u.a. Erfolgreich mit Objektorientierung: Vorgehensmodelle und Managementpraktiken für die objektorientierte Softwareentwicklung,Oldenbourg, München, 1999

[ÖFA02] H. Österle, E. Fleisch, R. Alt: Business Networking in der Praxis-Beispiele und und Strategien zur Vernetzung mit Kunden und Lieferanten, Springer,. Berlin 2002

[SH02] P. Stahlknecht, U. Hasenkamp: Einführung in die Wirtschaftinformatik, Springer Verlag Berlin, Heidelberg New York 2002, 10. Auflage

Internetquellen

[I.1] http://www.edition-w3c.de/TR/1998/REC-xml-19980210.html
20.10.2002, Extensible Markup Language (XML)

[I.2] http://www.w3c.org/XML/Activity
20.10.2002, Extensible Markup Language (XML) Activity Statement

[I.3] http:// www.competence-site.de
20.10.2002, Anforderungen, Entwicklungen und Trends im Bereich EAI

[I.4] http://www.omg.org/technology/documents/formal/uml.htm
20.10.2002, Unified Modeling Language (UML)

[I.5] http://www7b.software.ibm.com/wsdd/zones/portal/portlet/2.1api/,

19.06.2002, Java-Doc der Portal-API 2.1

[I.6] http://www-3.ibm.com/software/webservers/portal/library/PortletDevelopmentGuide.pdf, 14.09.2002, PortletDevelopement Guide

[I.7] http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246539.html?Open
26.07.2002, DB2 UDB e-business Guide

[I.8] http://www7b.software.ibm.com/wsdd/zones/vajava/
30.07.2002, Visual Age for Java Zone

[I.9] http://www-106.ibm.com/developerworks/ibm/library/
15.08.2002, History-Making Components - Tracing the roots of components from OOP through WS

[I.10] http://www-106.ibm.com/developerworks/ibm/library/i-portal2
17.06.2002, Portlet Applikation Development: Give and take -- portlet messaging techniques

[I.11] http://www-106.ibm.com/developerworks/library/i-portal/?dwzone=ibm
17.06.2002, Introduction to portlet structure and programming

[I.12] http://www.w3.org/Protocols/
17.06.2002, HTTP - Hypertext Transfer Protocol

[I.13] http://www.gartner.com/reprints/ibm/106501.html
17.06.2002, Big Change Evident in 2H02 Horizontal Portal Product MQ

[I.14] http://www.isss-awareness.cenorm.be/de/e-commerce-leitfaden/Definitions_de.htm#Portal

[I.15] http://jakarta.apache.org/
20.10.2002, jakarta Projekt von apache

[I.16] http://www.w3.org/TR/xpath
20.10.2002, XML Path Language (XPath)Version 1.0

[I.17] http://www.w3.org/MarkUp/
20.10.2002, HyperText Markup Language (HTML) Home Page

[I.18] http://www.w3.org/Style/XSL/
20.10.2002, The Extensible Stylesheet Language (XSL)

[I.19] http://www.w3.org/TR/xslt
20.10.2002, XSL Transformations (XSLT) Version 1.0

[I.20] http://xml.apache.org/
20.10.2002, Apache XML Project

[I.21] http://www-5.ibm.com/de/software/enews/essay/2001-02-essay-portale.pdf
20.10.2002, Portal Total

[I.22] http://rfc.sunsite.dk/index2201.html#2251
20.10.2002, Lightweight Directory Access Protocol (v3)

[I.23] http://www-3.ibm.com/software/webservers/portal/portlet/catalog
20.10.2002, Portlet catalog

[I.24] http://www.openldap.org/doc/admin21/guide.html
20.10.2002, LDAP

[I.25] http://www.ovum.com/go/product/sample/0048531.htm#TopOfPage
20.10.2002, Ovum

[I.30] http://www.informatik.uni-stuttgart.de/ipvr/as/lehre/seminar/docss00/message-broker.pdf
20.10.2002, Message Broker

[I.31] http://techupdate.zdnet.de/cgi-bin/de/techupdate/printer_friendly.cgi?id=2117822&tid=419&pid=1

20.10.2002, Portalsoftware wird immer begehrter
[I.32] http://www.uni-koeln.de/rrzk/www/suche/portale.html
20.10.2002, Portale, die Startseiten für das Surfen im Internet
[I.32] http://www.enews24.de/artikelprint.asp?id=7499
20.10.2002, Der Markt für Portal-Software
[I.33] http://www.wissensmanagement.net/online/archiv/2000/06_0700/Unternehmensportale.htm#1
20.10.2002, Informationslogistische Dienste für Unternehmensportale
[I.34] http://www.ids-scheer.com/de
20.10.2002
[I.35] http://www.ibm.com/software/webservers/portal/library/InfoCenter/
20.10.2002, Infocenter IBM WebSphere Portal Server
I.36 http://www-3.ibm.com/software/webservers/portal/library/V41PortletMigrationGuide.pdf
[I.37] http://java.sun.com/webservices/docs/1.0/tutorial/doc/WebApp3.html
20.10.2002, Web Application Archives
[I.38] http://www-3.ibm.com/software/webservers/appserv/doc/v40/aee/wasa_content/0503.html
20.10.2002, WAS 4.0 Changes to security since Version
[I.39] http://www.percussion.com/downloads.htm
20.10.2002, Xsplit takes any standard HTML file and splits it automatically into two related XML files
[I.40] http://freespace.virgin.net/joe.carter/TableGen/
20.10.2002, The Tablegen Main Page
I.41 https://www2.skwbschoellerbank.at/023/wps/portal/action/ChangePage/.pg/65/.reqid/1
20.10.2002, SKW Schoellerbank
[I.42] http://www.competence-site.de/eaisysteme.nsf/D544A36C8611C335C1256A5A00472B5C/$File/text_sailer_sercon.pdf
20.10.2002, Anforderungen, Entwicklung und Trends im Bereich Enterprise Application Integration (EAI), Mario Sailer

sonstige Quellen

[Meta00] Meta Group: Der Markt für Portale, Marktplätze und Mobile Commerce in Deutschland, Analyse, 2000
[Comp00] Computerwoche, Softwareprojekte brauchen Navigationshilfen, Computerwoche Nr. 50 vom 15.12.2000
[Java02] Java Magazin, Ausgabe 3.2002
[Gart01] Gartner Group: 2H01 Portal Products Magic Quadrant, Research Note vom 23.07.2001
[Lind00] B. Lindner, Vergleichende Untersuchungen zur XML-Repräsentation von Verkehrstelematik-Daten in Client-Server Anwendungen und deren mulimedialer Aufbereitung, Diplomarbeit, 2000
[Meta01] Meta Group: e-Business und Enterprise Application Integration: Der Schlüssel zum Erfolg, EAI-Studie 2001
[KN99] Ring Katy; Neil, Ward-Dutton (Ring/Neil 1999): Enterprise Application Integration , Making the Right Connections, Ovum Report 1999.

Index

A

B

C

D

E

G

H

J

K

L

M

P

R

S

T

U

V

W

X

Z

Die beiliegende CD

Auf der Beiliegenden CD sind neben der Diplomarbeit folgende Inhalte:

Verzeichnis	Inhalt
Webanwendung	Bestandteile des Prototypen
Webanwendung\Anzeigeobjekt in XML	Beispiele, wie innerhalb einer Anfage aus den Anzeigeobjekten XML Generiert wird
Webanwendung\Database	Datenbankmodell sowie Installationsskripte um die Datenbank zu erzeugen, ferner sind die Beispieldaten in XML enthalten
Webanwendung\HTTPStarseiten	Statische Html Seiten um den Protoyp in der Visualage for Java Test Umgebung (WTE) zu starten
Webanwendung\src	Quelltext zum Prototypen darunter Java Klassen des Evolutioin Frameworks, der erstellte Prototyp und im Unterverzeichnis APP die Konfiguration sowie die XSL Seiten
Software	Tomcat und jetspeed – die Opensource Lösung, die Basis für den WebSphere Portal server ist.
Portlets	Das erstellte Portlet
Doku	Infocenter des WebSphere Portal Server und div. weiter Dokumentationen
Usecases	Ausführliche Beschreibung der Anfroderungen an den Prototypen

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder Sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, den

Unterschrift