Universität Paderborn

Seminararbeit

Konzeption und Implementierung einer auf Web Services basierenden Portal-Kopplung auf Portlet-Ebene

vorgelegt bei

Prof. Dr. Ludwig Nastansky

Oktober 2002

betreut durch

Dipl.-Wirt.-Inf. Olaf Hahnl

vorgelegt von

Holger Ploch

Pontanusstraße 24

33102 Paderborn

Studiengang Wirtschaftsinformatik

Hendrik Voigt

Westernstraße 43

33098 Paderborn

Studiengang Wirtschaftsinformatik

Inhaltsverzeichnis

Abbildungsverzeichnis:

Abkürzungsverzeichnis:

ACL	Access Control List
API	Application Programming Interface
ASF	Apache Software Foundation
AXIS	Apache eXtensible Interaction System
CORBA	Common Object Request Broker Architecture
DB	Datenbank
DCOM	Distributed Component Object Model
E-Business	Electronic Business
EMNID	Erforschung der öffentlichen Meinung, Marktforschung, Nachrichten, Informationen und Dienstleistungen
GCC	Groupware Competence Center
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IAO	Frauenhofer Institut Arbeitswirtschaft und Organisation
IBM	Industrial Business Machines
PDA	Personal Digital Assistant
RMI	Remote Method Invocation
RPC	Remote Procedure Calls
SMTP	Simple Mail Transfer Protocol
SOA	Service-orientierte Architektur
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
TNS	Taylor Nelson Sofres
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WSDL	Web Services Definition Language
WSRP	Web Services for Remote Portlets
WS-Security	Web Services Security
XML	Extensible Markup Language

1 Einleitung

1.1 Motivation

Personalisierbare Intranet-Portale sollen den Mitarbeitern schnell und effizient Zugriff auf die für sie wichtigen Informationen bieten. In diesem Kontext ermöglichen Portale Organisationen einen strukturierten Zugriff auf große, verteilte und zum Teil völlig verschiedene Informationssysteme, die einzubinden sind. Da Portale zentrale Informationssysteme darstellen, müssen sie über Schnittstellen verfügen, um Informationen aus unterschiedlichen Quellen möglichst einfach und schnell einbinden zu können. Dadurch können diese Informationen zentral verwaltet und verteilt werden.

Virtuelle Unternehmen, also Unternehmenszusammenschlüsse auf beschränkte Zeit, sind in diesem Zusammenhang besonders an Schnittstellen zur organisationsübergreifenden Kooperation interessiert. Diese entwickeln sich momentan im Bereich des Informationsmanagements. Biermann konstatiert, dass „für eine Kopplung auf organisatorischer und technischer Ebene verschiedene Gesichtspunkte berücksichtigt werden (müssen). Wichtigster organisatorischer Aspekt sind die Unterschiede in den Einfluss- und Kontrollmöglichkeiten“ (vgl. [Biermann 2001] S.21). In anderen Unternehmensgebieten wie z.B. im Handel kann auf einheitliche Austauschsprachen wie EDIFACT zurückgegriffen werden.

Laut einer in der Reihe media vision expert des Fraunhofer Institut Arbeitswirtschaft und Organisation (IAO) und TNS EMNID[1] veröffentlichten Marktübersicht setzen „viele Anbieter von Portal Software (..) auf offene Schnittstellen zu Drittsystemen und standardisierte Datenaustauschformate. Portale werden damit immer mehr zum Zentrum von E-Business Aktivitäten und zur zentralen unternehmensinternen Plattform“ [Fraunhofer Institut 2002].

Web Services sind eine hoch aktuelle, noch in der Entwicklung befindliche Technologie, die es Applikationen ermöglicht, über ein standardisiertes Protokoll auf entfernte Applikationen zuzugreifen und mit ihnen zu kommunizieren. Web Services sind konzeptionell unabhängig von Technologien und Architekturen. Die hier vorgestellte Lösung gründet hauptsächlich auf dem Protokoll Simple Object Access Protocol (SOAP), der Beschreibungssprache Web Services Definition Language (WSDL) und der Registrierungstechnologie Universal Description, Discovery and Integration (UDDI), die im Kapitel 2.3 noch genauer betrachtet werden. Eine Entscheidung für Web Services ist ein Schritt hin zu offenen Architekturansätzen, da damit keine Wahl der Programmiersprache oder der Plattform getroffen wird.

[1] Verbund der Marktforschungsgruppe Taylor Nelson Sofres (TNS) und dem Markt-, Media- und Meinungsforschungsunternehmen „Erforschung der öffentlichen Meinung, Marktforschung, Nachrichten, Informationen und Dienstleistungen“ (EMNID)

Das folgende Szenario soll den Mehrwert für das G8-Portal, das im Kapitel 2.2 betrachtet wird, im Kontext einer Erweiterung der Lehre in Form einer Kooperation mit anderen Universitäten verdeutlichen.
Der Lehrstuhl Wirtschaftsinformatik 2 der Universität Paderborn schließt einen Kooperationsvertrag mit der Universität Bielefeld. Darin wird festgelegt, dass bei beiden Parteien ein auf der G8-Portal Engine basierendes Portal implementiert und ein Web Service Interface zum Austausch von Vorlesungsmaterialien verwendet wird.
Seitens der Universität Paderborn wird das G8-Portal bereits eingesetzt. Für die einzelnen Vorlesungen werden einerseits speziell angepasste Portlets und andererseits spezifische Templates definiert. Studenten können sich mittels ihres „Notes-HTTP Passwortes“ anmelden. Die Universität Bielefeld wird gemäß des Vertrags ein auf der G8-Portal Engine basierendes Portal implementieren und die Lehre auf diese Weise unterstützen. Mittels des geplanten Web Service Interfaces können die Portlets und Templates ohne weiteren Aufwand ausgetauscht und direkt verfügbar gemacht werden, ohne das jeweilige Authentifizierungsmanagement des anbietenden Portals auf Portlet-Ebene verwenden zu müssen. Der Mehrwert für die Studenten liegt in der gut strukturierten und zentral zugänglichen Informations- und Werkzeugsammlung. Gezielt können Vorlesungen nachbereitet und konkrete Wissensgebiete vertieft werden. Die Lehre nutzt für ihren Teil ebenfalls diese Vorteile. Ihr bleibt aber ein Mehraufwand durch langwierige Integration erspart.
Indem Entwickler keine proprietären Lösungen implementieren und der Administrator Inhalte leichter einbinden kann, lässt sich allgemein festhalten, dass diese generische, standardisierte Lösung den Beteiligten Arbeit erspart.
Das dargestellte Szenario lässt sich modifiziert auf kooperierende Organisationen übertragen.

1.2 Ziele der Arbeit

Den Rahmen dieser Seminararbeit bildet das G8-Portal. Das Hauptziel ist die Entwicklung einer Schnittstelle zur Kopplung der auf der G8-Portal Engine basierenden Portale. Die zugrundeliegende Technik für eine Implementierung des Interfaces sind Web Services. Eine weitere Evaluierung von Alternativen findet deshalb nicht statt.
Die Portal-Kopplung soll auf Portlet-Ebene stattfinden. Zudem ist eine Einbindung von Portlets in Templates des Consumer-Portals (siehe Kapitel 2.1.2) wünschenswert, aber kein primäres Ziel. Durch eine generische Lösung wird sichergestellt, dass der Administrator keine Programmierarbeit leisten muss und eine weitgehende Administrierbarkeit der Portal-Kommunikation bzgl. Initiierung, Manipulation und Beendigung vorhanden ist. Eine Trennung der Formatierungen und der Daten wird nicht gefordert. Zugriffsrestriktionen durch agierende Firewalls werden vernachlässigt.

Zur Realisierung dieser Ziele wird die existierende Technologielandschaft im Umfeld von Web Services evaluiert. Der Fokus der Arbeit ist in der Entwicklung einer offenen Architektur zu sehen, die möglichst generisch gestaltet ist. An die Konzeption schließt sich die Einbindung und Implementierung in das G8-Portal an.

1.3 Aufbau der Arbeit

Im Kapitel 2 der vorliegenden Seminararbeit werden grundlegende Konzepte, Begriffe, Technologien und Architekturen im Kontext von Web Services vorgestellt. Sie dienen als Basis für spätere Kapitel.
Im Kapitel 3 werden Designentscheidungen genannt und begründet. Es umfasst eine technische sowie konzeptionelle Komponente.
Darauf aufbauend folgt im Kapitel 4 die Präsentation der Lösung. Entwickelte Konzepte werden vorgestellt und ergänzen den kommentierten Quellcode. Komponentenmodell, Statechart-Diagramm und Zustandsautomat visualisieren die Portal-Kopplung. Konkrete Instanzen von Nachrichten sind aufgearbeitet, um eine bessere Lesbarkeit zu erreichen.
Kapitel 5 umfasst sinnvolle Weiterentwicklungen hinsichtlich Web Services Security und einer Places-Komponente.
Den Abschluss bildet Kapitel 6 mit einem Fazit.

2 Grundlagen

2.1 Begriffe

2.1.1 Portal

Das G8-Portal bildet die Basis dieser Seminararbeit, so dass Portalen und dem damit verbundenen Portal-Gedanken eine grundlegende Rolle zukommt.
Eine einheitliche und befriedigende Definition des Begriffes Portal sucht man vergebens. Die unterschiedlichen Versuche diesbezüglich skizzieren Teilaspekte, ohne eine umfassende Definition geben zu können. Die folgende Liste enthält typische Merkmale für Internet-Portale:

- Funktion einer Einstiegsseite ins Internet als single point of access
- Verfügbarkeit von Such- und Navigationstools
- Strukturierung der Internetinhalte durch Hierarchisierung (Auto ⇨ Werkstatt ⇨ VW Vertragswerkstatt ⇨ Werkstatt xyz) und Priorisierung (z.B. verweisende Links auf eine Seite als Indiz für Relevanz)
- Interaktivität wie Email und Chat gegeben
- Vorhandensein von Personalisierungsmöglichkeiten

Ein weiteres Indiz für Internet-Portale ist, dass sie zur Erreichung wirtschaftlicher Ziele dienen. Informationen werden mit kommerziellen Werbeeinblendungen verbunden und vermischt. Der Besucher soll möglichst lange an die Seite gebunden werden.
Im Gegensatz dazu wird bei personalisierbaren Intranet-Portalen versucht, dem Mitarbeiter schnell und effizient Zugriff auf für ihn wichtige Informationen zu ermöglichen. In diesem Kontext bieten Portale den Organisationen einen strukturierten Zugriff auf große, verteilte und z. T. sehr verschiedene Informationssysteme. Dem einzelnen Benutzer wird eine personalisierte Sicht auf das Unternehmen geboten.
Innerhalb von virtuellen Unternehmen können Portale als zentrale Informationssysteme dienen. Sie bieten Schnittstellen, so dass die einzelnen beteiligten Unternehmen Zugang zu ihren Informationen bereitstellen können. Diese Informationen werden von der geschaffenen virtuellen Unternehmung verwaltet und verteilt. Die Fähigkeit, in allen Abteilungen einen signifikanten Datenfluss zu ermöglichen und zu integrieren und auf dieser Basis fundierte Entscheidungen zu fällen, bestimmt den Erfolg.
Für weitergehende Betrachtungen wird auf [Biermann 2001], [Hahnl 2000] und [Schmidt 1999] verwiesen.

2.1.2 Portale als Endknoten einer Web Service Architektur

Bei der Kopplung von Portalen mit Hilfe von Web Services nehmen diese aus konzeptioneller Sicht unterschiedliche Funktionen wahr. Zuerst wird das Portal, das einen Service an-

bietet, betrachtet. Es füllt die Rolle eines Providers aus. Im folgenden wird dieser Portal-Knoten als Service-Portal bezeichnet. Der Nutzer konsumiert den Dienst des Service-Portals und wird daher als Consumer-Portal definiert.

2.2 Architektur des G8-Portals

Im folgenden beschränkt sich die Betrachtung auf die für dieses Projekt relevanten Bestandteile der Portal-Architektur. Für eine detaillierte Beschreibung wird auf die Diplomarbeit von Olaf Hahnl [Hahnl 2000] verwiesen. Die Portal-Architektur orientiert sich an den Gegebenheiten, die am Groupware Competence Center (GCC) von Prof. Dr. Nastansky an der Universität Paderborn vorhanden sind. Das impliziert die Ausrichtung auf eine prozessgetriebene Lotus Domino Infrastruktur, die aus der Groupware-Plattform Lotus Domino und den darauf aufbauenden Applikationen besteht. Diese Plattform ermöglicht das Ablegen, Verwalten und Verteilen von Informationen.

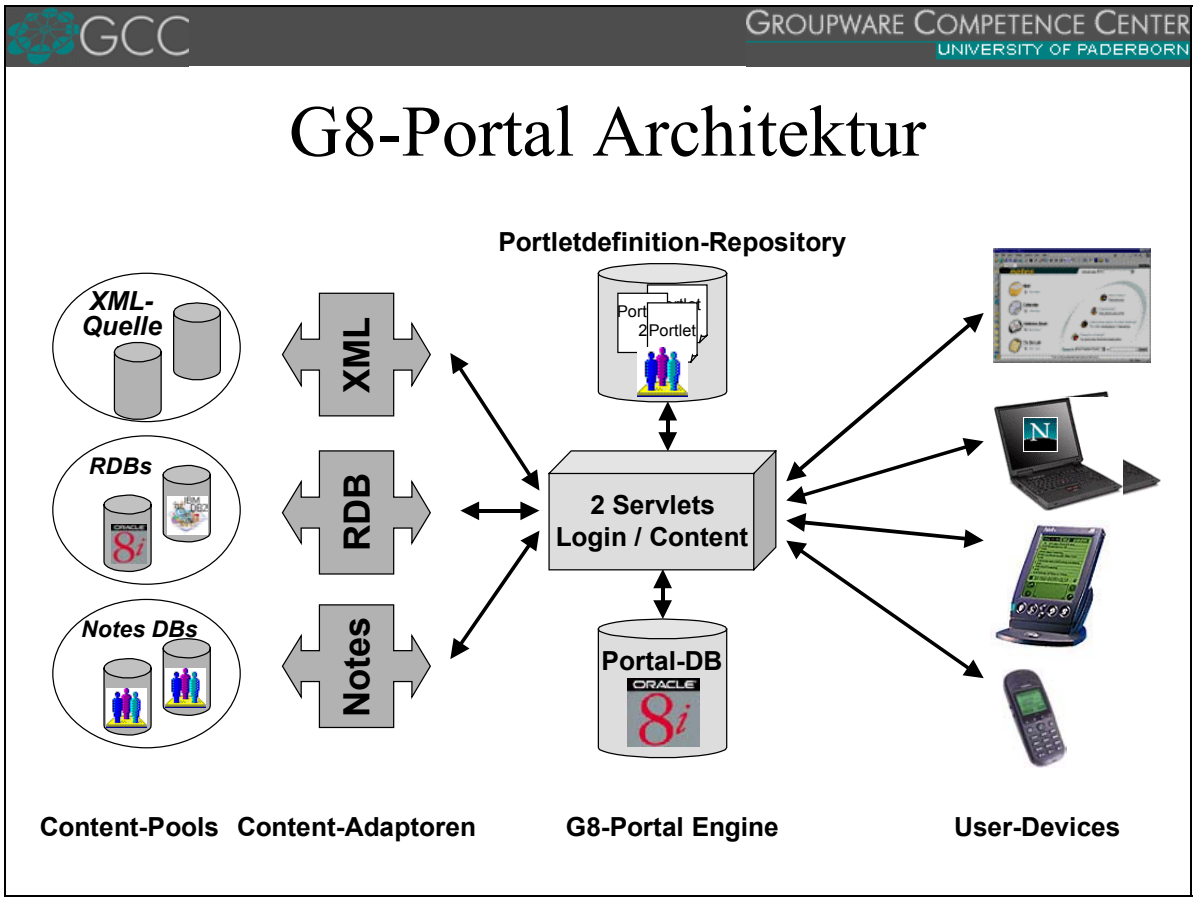

Abbildung 1: G8-Portal Architektur in Anlehnung an [Bruse 2001]

Die Content-Adaptoren dienen als Zwischenschicht. Sie integrieren sogenannte Content-Pools wie z.B. Lotus Domino Datenbanken, relationale Datenbanken etc. in das Portal. Sie realisieren die physische Anbindung an die Basisdienste der G8-Portal Engine. Der Content-

Adapter stellt Portlets, also die kleinste zusammenhängende Informationseinheit des Portals, dar. Dazu greift er auf entsprechende Konfigurationsinformationen aus dem Portletdefinition-Repository zu. Er erhält von der G8-Portal Engine die Daten über die vorgenommene Benutzerpersonalisierungen aus dem Portletdefinition-Repository.
Jede Anbindung eines Content-Pools wird mittels dem Portletdefinition-Repository angelegt, verwaltet und ggf. gelöscht. Die auf diese Weise entstehenden Informationsmodule, im folgenden Portlets genannt, sind Träger der eigentlichen Funktionalität. Der Benutzer kann diese in Form von Bausteinen der Oberfläche hinzufügen und personalisiert somit Bereiche des G8-Portals. Folglich gruppieren die so entstehenden Seiten Informationen.
Neben den durch die Nutzer personalisierten Bereichen lassen sich Angebote zentral erstellen und Nutzergruppen zuordnen. Diese werden als Templates bezeichnet.
Die G8-Engine basiert auf zwei Servlets. Zur Ausführung benötigen diese einen Servlet-Container, der z.B. in WebSphere, in einem Domino- oder auch in einem Tomcat-Server, integriert ist. Die jetzige Realisation des G8-Portals am GCC beruht auf einer Tomcat Server-Umgebung, die in den Lotus Domino Server R5 eingebettet ist. Die Personalisierungsinformationen sowie die personalisierten Meta-Daten werden in einer relationalen Datenbank gehalten, um auch bei großen Benutzerzahlen eine gute Performance zu gewährleisten. Aktuell wird eine Oracle8i Datenbank eingesetzt. Andere Datenbanken sind problemlos implementierbar.

2.3 Web Services

In Anlehnung an die von IBM veröffentlichte Definition von Web Services wird zuerst eine Arbeitsdefinition gegeben [IBM 2001].

2.3.1 Definition

Ein Web Service stellt eine Schnittstelle dar. Sie erfüllt mehrere Aufgaben. Die Anwendungslogik ist in Form von Operationen gekapselt, auf die über ein Netzwerk mit Hilfe standardisierter XML-Nachrichten zugegriffen werden kann. Eine zentrale Eigenschaft von Web Services besteht darin, dass durch die standardisierte und formalisierte XML-Notation, die den Web Service beschreibt, die konkrete Implementierung weitgehend verborgen bleibt. Dies ermöglicht sowohl hardware-, software- als auch programmiersprachenunabhängige Implementierungen.

Es gibt keine allgemein akzeptierte Definition von Web Services. Microsoft hat bereits zwei unterschiedliche Definitionen gegeben (vgl. [Microsoft 2002 a] und [Microsoft 2002 b]).

2.3.2 Service-orientierte Architektur (SOA)

In Anlehnung an Graham (vgl. [Graham et al. 2002] S.23 f.) lässt sich die Rolle der unten aufgeführten Technologien mit Hilfe einer service-orientierten Architektur beschreiben.
Service-Nachfrager, Service-Provider und Service-Registrierung sind die drei zu identifizierenden Rollen.

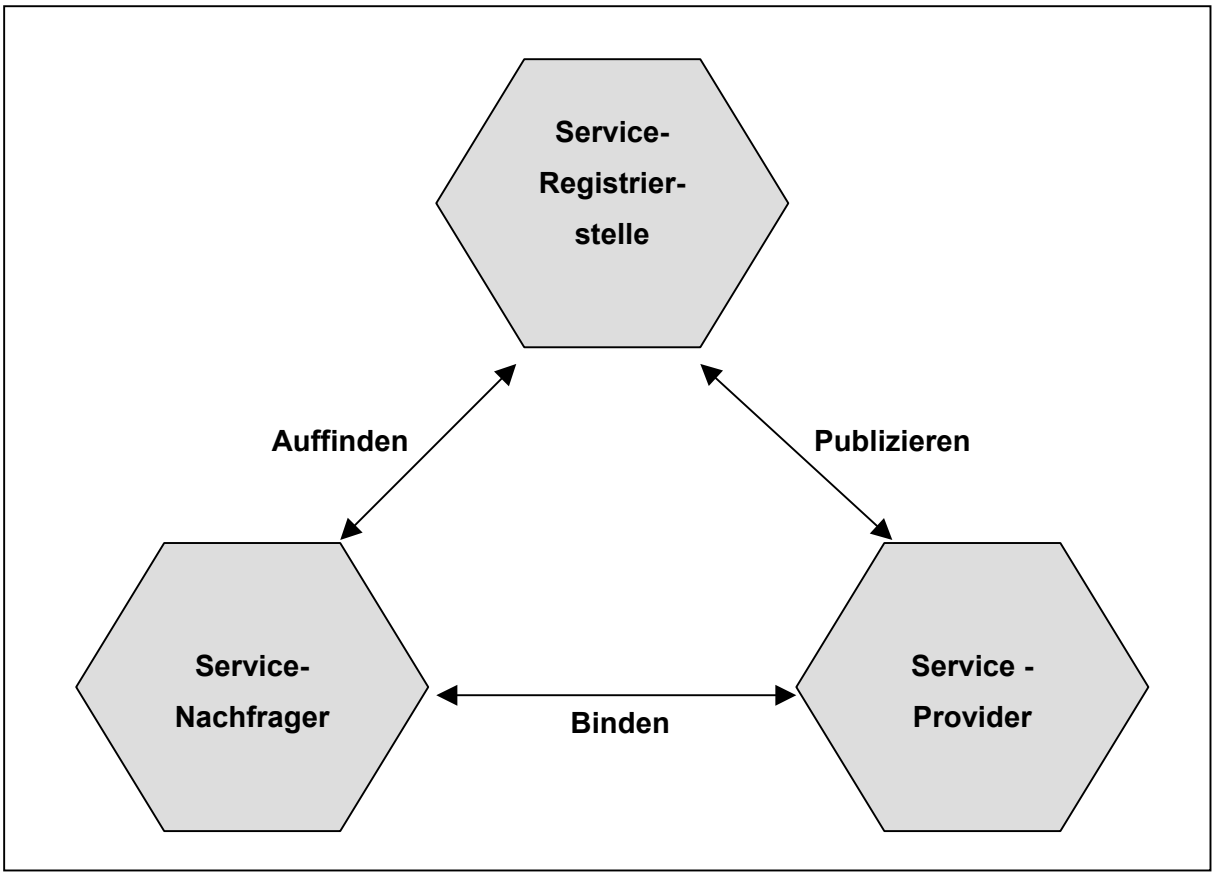

Abbildung 2: SOA-Architektur

Der Service-Provider ist für das Erstellen der Service Beschreibung zuständig, die in einer oder mehreren Service-Registrierstellen publiziert wird. Der Service-Provider stellt die Web Service Funktionalitäten bereit und agiert als Server.
Die Service-Registrierstelle veröffentlicht die Web Service Beschreibung und stellt eine Suchfunktion zur Verfügung. Sie übernimmt die Rolle des Bindeglieds zwischen dem Service-Nachfrager und dem Service-Provider.
Als Konsument des Web Services tritt der Service-Nachfrager auf. Er greift auf die Suchfunktionalitäten der Registrierungsstelle zurück, die unterschiedlich komplex gestaltet sein können. Um den Kontrakt zwischen Service-Nachfrager und Service-Provider zu etablieren, wird die Service Beschreibung in dem Web Services Definition Language Format (WSDL) übermittelt. Der Web Service kann nun an die Applikationen gebunden und schließlich mittels des Simple Object Access Protocols (SOAP) aufgerufen werden.

2.3.3 Simple Object Access Protocol (SOAP)

SOAP[2] ist zu einem Synonym für Web Services geworden (vgl. [Graham et al. 2002] S.114). SOAP stellt in diesem Kontext den de facto Standard für den Nachrichtenaustausch und für das Aufrufen von Prozeduren dar. Die SOAP 1.1 Spezifikation wurde von namhaften Unternehmen wie IBM und Microsoft unter der Federführung des World Wide Web Consortium (W3C) verfasst. Es besteht weder eine feste Bindung an eine Programmiersprache noch an eine Plattform oder an spezielle Hardware. Es basiert auf XML 1.0, XML Schema und XML Namespaces. Das Protokoll wird für Kommunikation, Verbindungsmanagement, Sicherheit, Übertragungssupport, Serialisieren und Deserialisieren[3] von Daten, Protokollevolution, Versionsmanagement, Fehlerbehandlung und weitere Aufgaben verwendet. Für eine Vertiefung sei auf [Cauldwell et al. 2001] und [Graham et al. 2002] verwiesen.

2.3.4 Web Services Definition Language (WSDL)

WSDL[4] ist eine technische Beschreibung des Web Services. Eine derartige Beschreibung wird z.B. für die Festlegung eines Nachrichtenprotokolls, des Nachrichtenformats und auch zur Mitteilung der Netzwerkadresse benötigt. Im Gegensatz zu einer textuellen Beschreibung ist WSDL formalisiert und standardisiert. Der Service-Nachfrager kann eindeutig feststellen, wie er den Web Service aufruft. Es existiert eine umfangreiche Toolunterstützung für eine Reihe von Programmiersprachen. Serverseitig werden WSDL Dokumente ausgehend von der bestehenden Informationstechnologie automatisch erstellt. Clientseitig lassen sich mit Hilfe dieser Tools Hilfsklassen zum Web Service Zugriff erstellen, so dass eine Kapselung erfolgt und der Service-Nachfrager mit Objekten kommuniziert.

2.3.5 Universal Description, Discovery and Integration (UDDI)

Der Zweck von UDDI liegt in der Unterstützung beim Entdecken und Veröffentlichen von Web Services. Dazu definiert es einen Standard für Service-Registrierungsstellen.

UDDI definiert Datenstrukturen und eine Application Programming Interface (API) Spezifikation. Auf diese Weise wird eine programmierbare Registrierung und das Auffinden von Meta-Informationen wie des Service-Typs oder des Bindings realisiert. Außerdem sind die UDDI Registrierungsoperationen zur Registrierung, Verwaltung und Auffindung in einer web-basierten Benutzerschnittstelle eingebunden.

UDDI umfasst zusätzlich eine nicht-funktionale Beschreibung. Diese gibt Aufschluss darüber, wer der Service-Provider ist und welche Funktion der Web Service z.B. im Rahmen eines

---

[2] http://xml.apache.org/soap/index.html

[3] Beim Serialisieren werden Objekte in Sequenzen gekapselt. Deserialisieren ist der entgegengesetzte Vorgang.

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

Geschäftsprozesses erfüllt. Das UDDI Konzept ist eng mit dem von WSDL verzahnt. Eine öffentliche und offizielle Geschäftsregistrierung befindet sich unter www.uddi.org. Es lassen sich allerdings auch private UDDI Registrierungsstellen schaffen, um beispielsweise sicherzustellen, dass sich nur ausgewählte Geschäftspartner registrieren können.
Aufgrund der Komplexität von Portlets und dem Anwendungsgebiet kann bei der Kopplung von G8-Portalen vorausgesetzt werden, dass die Gegenstellen einander bekannt sind. Somit kann auf UDDI in dieser Seminararbeit verzichtet werden.

2.3.6 Betriebswirtschaftlicher Nutzen

Der betriebswirtschaftliche Nutzen lässt sich prägnant durch das Wort „Integration" beschreiben. Einerseits können Applikationsfunktionalitäten eingebunden werden, andererseits werden Geschäftsbeziehungen durch die Integration von Applikationen gefördert. Unterstützend wirkt dabei der Aspekt der komponentenbasierten Entwicklung. In einem Komponentenmodell werden Schnittstellenbeschreibungen, Protokolle und ausdrückliche Kontextabhängigkeiten[5] standardisiert. Softwaresysteme können auf Basis von Komponenten zusammengesetzt werden. Dadurch steigt die Wiederverwendbarkeit und der zeitliche und damit finanzielle Aufwand für die Erstellung sinkt. Das volle Einsparungspotential der Anwendungsinfrastruktur wird jedoch erst bei organisationsübergreifender Geschäftsprozessautomatisierung wirksam (vgl. [Nußdorfer 2002]).

2.3.7 Web Services als Komponentenmodell

Aus technischer Sicht ist der Grundgedanke der Integration nicht neu, dafür aber die Verwendung offener Standards und Technologien.
Ein Web Service bietet mehr als ein Programm im Internet, das mit standardisierten Protokollen aufzurufen ist. Für diese Verfahrensweise existieren die Common Object Request Broker Architecture (CORBA), die Remote Method Invocation (RMI) und das Distributed Component Object Model (DCOM) bzw. deren Implementierungen. Diese Komponentenmodelle gewährleisten keine vollständige Unabhängigkeit von Plattformen und Programmiersprachen. Zusätzlich sind die verwendeten Übertragungsprotokolle nicht immer völlig unabhängig vom jeweiligen Verfahren.
Web Services stellen Softwarekomponenten dar, die zusätzlich zu dem in Kapitel 2.3.6. beschriebenen betriebswirtschaftlichen Nutzen Plattformunabhängigkeit verwirklichen. Neben Remote Procedure Calls (RPC), also Aufrufen entfernter Methoden und Funktionen, verarbeiten Web Services dokumentenbasierte Nachrichten. Die auszutauschenden Daten werden in XML verpackt. Dieses Vorgehen ermöglicht ein Höchstmaß an Interoperabilität (vgl. [Röwekamp 2002]).

[5] Funktionen, die eine Softwarekomponente von anderen verwendet

3 Designentscheidungen

3.1 Konzeptionelle Entscheidungen

3.1.1 Verwaltung der Portal-Dienste

Das Service-Portal bietet als Portal-Dienst vordefinierte Portlets an. Die G8-Architektur sieht eine Verwaltung der Portlets im Portletdefinition-Repository vor. Prinzipiell sind drei mögliche Varianten der portalübergreifenden Portlet-Kopplung denkbar. Die Konfigurationsdokumente liegen nur auf einem der beiden Portale oder auf beiden gleichzeitig.

Der Fall, dass die Konfigurationsdokumente nur auf dem Consumer-Portal liegen, ist auszuschließen. Es ist auf diese Weise nicht möglich, ein Portlet als Portal-Dienst anzubieten, da es auf dem Service-Portal nur mit einem Konfigurationsdokument existieren kann.

Ein erster intuitiver Ansatz zur Verwaltung der Portlets besteht darin, sämtliche Konfigurationsintelligenz auf Seiten des Service-Portals zu implementieren. Bei einer Anfrage vom Consumer-Portal wird eine Liste der Portlets zurückgegeben, auf die zugegriffen werden kann. Aber bei der jetzigen Portal-Architektur wird ein Content-Adapter als kleinste Informationseinheit benötigt und dessen Instanz muss auf dem Consumer-Portal existieren. Deshalb können die Konfigurationsdokumente nicht ausschließlich auf dem Service-Portal vorgehalten werden. Somit folgt für die Konzeption, dass jedes Konfigurationsdokument sowohl auf dem Service- als auch auf dem Consumer-Portal vorhanden sein muss.

Die Konfigurationsdokumente in dem Portletdefinition-Repository des Service-Portals werden um Einträge ergänzt, die anzeigen, ob ein entfernter Zugriff möglich ist. Für jeden Portal-Dienst wird eine weitere Instanz des neuen Konfigurationsdokuments des Remote Portlets im Portletdefinition-Repository des Consumer-Portals erstellt. In dieser Instanz werden Zugriffsinformationen gespeichert. Somit besteht für jede Portletdefinition auf dem Consumer-Portal ein Pendant auf dem Service-Portal. Dieses Paar von Konfigurationsdokumenten wird über einen Synchronisationsmechanismus abgeglichen. Werden Portlets gelöscht, erstellt oder für das Consumer-Portal bedeutende Eigenschaften verändert, werden diese Informationen automatisch verteilt.

3.1.2 Transportprotokoll

Aufgrund der Unabhängigkeit des SOAP-Protokolls vom Transportprotokoll, muss eine Wahl zwischen den universal verwendeten Protokollen Simple Mail Transfer Protocol (SMTP) und Hypertext Transfer Protocol (HTTP) getroffen werden.

SMTP unterstützt das Push-Prinzip. Die SOAP Nachrichten werden an die Gegenstelle gesendet, jedoch ohne eine direkte Antwort zu erhalten. Ein Echtzeitverhalten wird nicht erreicht.

Das HTTP-Protokoll realisiert einen Nachrichtenaustausch via Request/Response-Mechanismus. Eine Response beinhaltet neben einer SOAP-Nachricht auch HTTP-spezifische Statusinformationen. Eine erfolgreiche HTTP-Transaktion enthält den Statuscode „200 (OK)“. Dieser zeigt an, dass der Client-Request erfolgreich war und die Server-Response die angeforderten Daten enthält. Bei Auftreten eines Serverfehlers kann die Ursache beispielsweise eine angeforderte Aktion vom Client sein, die vom Server nicht ausgeführt werden kann (Code 501 (Not Implemented)). Einige Frameworks wie das Apache eXtensible Interaction System (AXIS, siehe Kapitel 3.2) unterstützen die Auswertung dieser Rückgabecodes. Dies ist ein Vorteil, da sofort und anhand qualitativ hochwertiger Informationen entschieden werden kann, ob eine Anfrage erneut zu senden ist. Das Echtzeitverhalten erleichtert die Kommunikation und entspricht eher einem RPC. Die vorgestellte Lösung basiert auf dem HTTP-Protokoll, da für eine Portal-Kommunikation der zuletzt genannte Aspekt von Bedeutung ist.

3.1.3 Benutzerverwaltung

Ein erster naheliegender Ansatz der Benutzerverwaltung ist, die Benutzer des Consumer-Portals auch im Service-Portal anzulegen. Dies erfordert aber zwischen den beiden Anmeldeinformationen eine eindeutige Zuordnung, die wiederum in der Benutzerverwaltung des Consumer-Portals hinterlegt sein muss. Dieses Konzept ist im Hinblick auf Änderungen wenig robust, so dass aufwändige Synchronisationsmechanismen notwendig sind, um Datenkonsistenz zu erreichen.

Für das dargestellte Szenario genügt eine einfachere Benutzerverwaltung. Eine einheitliche Kennung für jedes Consumer-Portal sorgt für eine Autorisierung am Service-Portal. Dieses verwaltet Zugriffsdaten und Rechte für eine geringe Anzahl von Consumer-Portalen.

3.2 Wahl der SOAP Implementation

Es sind derzeit diverse SOAP Implementationen verfügbar und ihre Zahl steigt stetig. Auf eine detaillierte Sichtung der einzelnen Lösungen muss verzichtet werden, da diese den Schwerpunkt der Seminararbeit bedeutend verschieben würde.

Prinzipiell ist bei kleineren Projekten und im universitären Kontext von kommerziellen Produkten abzusehen. Open-Source-Produkte sind kostenlos und durchaus konkurrenzfähig. Sie haben den Vorteil, dass sie sich nicht engen Zeitrestriktionen und der Notwendigkeit einer Refinanzierung unterwerfen müssen. Es entsteht der Eindruck einer evolutionären Entwicklung.

Aufgrund der Tatsache, dass Microsoft den Anstoß für die Entwicklung zu SOAP gab und im weiteren Verlauf stark an der Formulierung der Spezifikation beteiligt war, hätte sich

Microsoft´s SOAP Toolkit 2.0[6] angeboten. Allerdings wurde es für die Programmiersprache Visual Basic entwickelt. Eine Verwendung setzt eine Einarbeitungsphase in die Programmiersprache und entsprechende Adaptionen im Portal voraus. Das Risiko, die Umgebung nicht integrieren zu können, ist als hoch einzuschätzen.

Für Java wurde erst das Apache SOAP Toolkit 2.0 empfohlen, das ursprünglich IBM´s alphaWorks Abteilung entwickelte (Vgl. [Cauldwell et al. 2001] S.34). Aufgrund von Standardisierungsbestrebungen übergab IBM es an die Apache Software Foundation (ASF). Anfänglich kritisierten Entwickler die nicht besonders umfangreiche und unpräzise Dokumentation. Derweil ist das Toolkit samt überarbeiteter Dokumentation in das Apache Projekt sorgfältig eingepflegt. Die Versionsnummer liegt aktuell bei 2.3.1. Der Webseite der Apache Organisation[7] zufolge kann allerdings von einer Weiterentwicklung nicht ausgegangen werden.

Apache AXIS stellt das Nachfolgeprojekt dar, das von der Apache Software Foundation ins Leben gerufen wurde. Aus den Erfahrungen mit SOAP 2.2 wird ein komplett neuer Ansatz angestrebt. Sowohl in der Architektur, die Offenheit und Erweiterbarkeit fördert, sowie komplexe Prozessmodelle unterstützt, als auch in der Performance sind deutliche Verbesserungen zu Konkurrenzprodukten zu erkennen (vgl. [Graham et al. 2002] S. 125 f.). Ein weiterer bedeutender Aspekt ist, dass AXIS sowohl client- als auch serverseitig genutzt werden kann. AXIS gilt als eine der vielversprechendsten Java-basierten Web Service Engines.

In der im Kapitel 4 vorgestellten Lösung wird Apache AXIS verwendet.

---

[6] Das Toolkit ist kostenfrei und lässt sich unter http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/001/580/MsdnCompositeDoc.xml herunterladen.

[7] http://xml.apache.org/axis/index.html

4 Technische Einbindung

4.1 Verteilte Portal-Architektur

Die bestehende Portal-Architektur wird um Komponenten erweitert, die eine Portal-Kopplung realisieren.

Die Funktionsweise des G8-Portals wird um das G8 Portal SOAP Gateway erweitert. Es bildet die Schnittstelle des Service-Portals. Eingehende SOAP-Requests werden empfangen und analysiert. Darauf aufbauend wird eine SOAP-Response erstellt.

Das G8-Portal SOAP Gateway Proxy des Consumer-Portals stellt das Pendant zum G8 Portal SOAP Gateway dar. Der Remote SOAP Adapter realisiert eine Harmonisierung der Portal-Kopplung, da Portlets des Service-Portals und lokale Portlets gleichbehandelt werden.

Die Kooperation der unterschiedlichen Komponenten zweier auf der G8-Portal Engine basierenden Portale ist in der Abbildung 3 visualisiert.

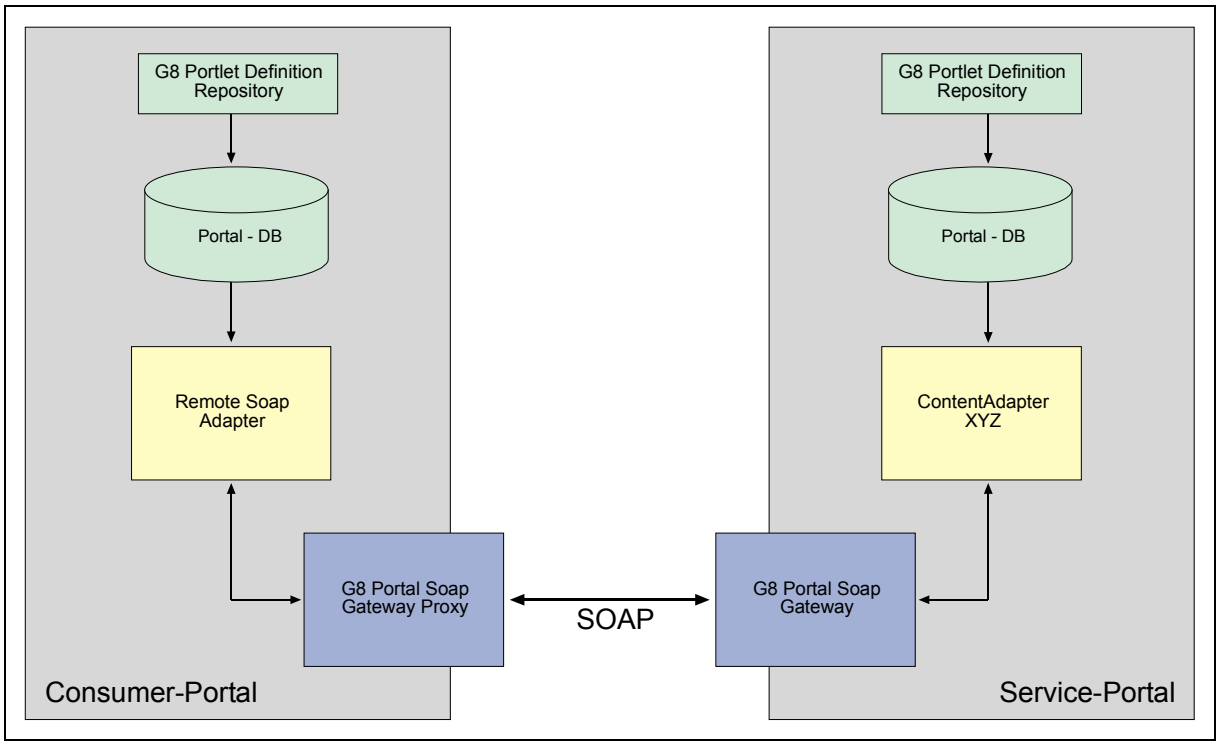

Abbildung 3: Komponentenkooperation bei einer Portal-Kopplung

4.2 Application Programming Interface (API) des Service-Portals

Innerhalb der Portletdefinition-Repository Umgebung sind Remote Service Connections integriert. Auf diese Weise erhält das Consumer-Portal die nötigen Informationen für den Zugriff auf das Service-Portal. Die Voraussetzungen für das Binden an den Web Service des Service-Portals und somit für den Zugriff auf dessen API sind damit vorhanden.

4.2.1 Abruf der verfügbaren Portlets

Die verfügbaren Portlets werden bei der Konfiguration einer Seite im G8-Portal aufgelistet. Diese Aufstellung der verfügbaren Portlets setzt sich aus den lokalen und den Remote Portlets der Service-Portale zusammen. Das Consumer-Portal kann sich eine Auflistung der zuletzt genannten Remote Portlets durch die Funktion „getAvailablePortlets()“ generieren lassen und erreicht auf dieser Ebene eine Synchronisation mit dem jeweiligen Service-Portal. Der untere Zustandsautomat kennzeichnet die Kommunikationsstruktur.

Abbildung 4: Abruf der verfügbaren Portlets

4.2.2 Kommunikation auf der Basis von Portlet-Instanzen

Die grundlegenden Konzepte der API werden anhand eines beispielhaften Lebenszyklus eines Portlets erläutert.

Die Remote Portlets des Service-Portals können in eine Seite des Consumer-Portals integriert werden. Typischerweise werden bei der Auswahl dieser Portlets durch einen Benutzer Instanzen auf dem Service-Portal erzeugt. Der Lebenszyklus eines Portlets beginnt.

Nachfolgende SOAP-Nachrichten sind durch Kommentare angereichert und erläutern die Kommunikation der Portale von der Initiierung bis hin zur Löschung der jeweiligen Portlet Instanz. Aufgrund des Umfangs der auszutauschenden Daten werden nicht alle SOAP-Nachrichten aufgeführt und ausführlich erklärt.

Erzeugen einer neuen Portlet-Instanz:

Nach der obligatorischen Erzeugung einer Session kann auf den Web Service des Service-Portals zugegriffen werden. Durch createPortletInstance() wird eine neue Instanz eines Portlets erzeugt.

Request des Consumer-Portals an das Service-Portal: createPortletInstance()

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
	<soapenv:Body>
	<!--Anwendungsspezifische Elemente beinhalten eigentlichen Request mit
	Inputparametern. Aufruf der Funktion createPortletInstance(Remote
	Portal,geheim,ip,GER,[SESSIONID],50,1)-->
		<g8:createPortletInstance>
			<user xsi:type="xsd:string">Consumer Portal</user>
			<pwd xsi:type="xsd:string">geheim</pwd>
			<ip xsi:type="xsd:string">ip</ip>
			<lang xsi:type="xsd:string">GER</lang>
			<session xsi:type="xsd:string">[SESSIONID]</session>
			<portletID xsi:type="xsd:int">50</portletID>
			<device xsi:type="xsd:int">1</device>
		</g8:createPortletInstance>
	</soapenv:Body>
</soapenv:Envelope>
```

Response des Service-Portals auf den Request des Consumer-Portals

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
	<soapenv:Body>
	<!--Anwendungsspezifische Elemente beinhalten eigentlichen Response mit Rückgabewerten
	RemotePageKey und RemotePortletKey.-->
		<g8:createPortletInstanceResponse>
			<createPortletInstanceReturn xsi:type="soapenc:Array"
				soapenc:arrayType="xsd:string[2]">
				<item>[RemotePageKey]</item>
				<item>[RemotePortletKey]</item>
			</createPortletInstanceReturn>
		</g8:createPortletInstanceResponse>
	</soapenv:Body>
</soapenv:Envelope>
```

Übermittlung des Portlet-Markups:

Eine essentielle Funktionalität eines Portlets des Service-Portals liegt in der Bereitstellung des eigentlichen Inhalts. Dieser wird im Allgemeinen als „Markup“ bezeichnet. Das Service-Portal stellt den Markup über die Methode „getPortletMarkup()“ zur Verfügung.

Request des Consumer-Portals an das Service-Portal : getPortletMarkup()

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
	<soapenv:Body>
		<!--Anwendungsspezifische Elemente beinhalten eigentlichen Request mit
		Inputparametern. Aufruf der Funktion getPortletMarkup(PortletRequestData:
		requestData). Ein PorletRequestData Objekt besteht aus einer Vielzahl von
		Feldern, die in einer serialisierter Form der unteren Auflistung zu
		entnehmen sind.-->
		<g8:getPortletMarkup>
			<requestData xsi:type="g8:PortletRequestData">
				<portletKey xsi:type="xsd:string">
					[RemotePortletKey]
				</portletKey>
				<IP xsi:type="xsd:string" xsi:nil="true"/>
				<!--Die lokale Adresse des Consumer-Portals dient dem Service-
				Portal als Basis-URL zur Erzeugung von Verweisen, so dass diese
				im Consumer-Portal verarbeitet werden können-->
				<baseURL xsi:type="xsd:string">HTTP://CONSUMERPORTAL</baseURL>
				<!--Ausgabegerät, hier immer Wert 1 := HTML -->
				<device xsi:type="xsd:int">1</device>
				<!--Darstellungsmodus:
				1 -> normale Größe; 2 -> maximierte Größe -->
				<windowMode xsi:type="xsd:int">1</windowMode>
				<!--Benutzername des Consumer-Portals-->
				<user xsi:type="xsd:string">Consumer Portal</user>
				<!--Jeweiliger Status, in dem sich das Portlet befindet. Es
				werden ContentMode(1), CustomizeMode(2) und HelpMode(3)
				unterschieden. -->
				<currentMode xsi:type="xsd:int">1</currentMode>
				<!--Sprache-->
				<language xsi:type="xsd:string">GER</language>
				<!--Anzahl Spalten auf einer Seite -->
				<windowLayout xsi:type="xsd:int">1</windowLayout>
				<pageKey xsi:type="xsd:string">[RemotePageKey]</pageKey>
				<!--Default Wert für Spracheinstellung-->
				<defaultLanguage xsi:type="xsd:string">GER</defaultLanguage>
				<!--Session-->
				<session xsi:type="xsd:string">[SESSION]</session>
				<!--PortletKey des RemoteSOAPAdapters auf dem Consumer-Portal-->
				<urlPortletKey xsi:type="xsd:string">
					[RemoteSOAPKey]
				</urlPortletKey>
				<!--Passwort des Consumer-Portals -->
				<password xsi:type="xsd:string">geheim</password>
				<!--Vorheriger Modus, siehe auch currentMode-->
				<previousMode xsi:type="xsd:int">1</previousMode>
			</requestData>
		</g8:getPortletMarkup>
	</soapenv:Body>
</soapenv:Envelope>
```

Response des Service-Portals auf den Request des Consumer-Portals

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
	<soapenv:Body>
		<!--Anwendungsspezifische Elemente beinhalten eigentlichen Response.-->
		<g8:getPortletMarkupResponse xsi:type="g8:PortletResponseData">
			<!--Adaptoren können spezielle Aktionen in die Kopfzeile hinzufügen,
			z.B. Home beim WebApplicationAdapter-->
			<specialActions xsi:type="RemoteContainer"/>
				<keys xsi:type="soapenc:Array"
				soapenc:arrayType="xsd:string[1]">
					<item>HOME</item>
				</keys>
				<values xsi:type="soapenc:Array"
					soapenc:arrayType="xsd:string[1,2]">
					<item xsi:type="xsd:string">
						HTTP://CONSUMERPORTAL?GETACTION!
						PortletKey=[PortletKey]!DEV=1!HOME=1
					</item>
					<item xsi:type="xsd:string">web_home.gif</item>
				</values>
			</specialActions>
			<!--Eigentliche Funktionalität, die durch Porlet bereitgestellt werden
			soll -->
			<markup xsi:type="xsd:string">
				&lt;p&gt;In diesem Abschnitt ist ein Markup definiert!&lt;/p&gt;
			</markup>
			<session xsi:type="xsd:string">[SESSION]</session>
		</multiRef>
		</g8:getPortletMarkupResponse>
	</soapenv:Body>
</soapenv:Envelope>
```

Löschen einer Portlet-Instanz:

Der Lebenszyklus eines Portlets endet mit seiner Löschung. Die API sieht dafür folgende Kommunikation vor.

Request des Consumer-Portals an das Service-Portal: destroyInstance()

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
	<soapenv:Body>
		<!--Anwendungsspezifische Elemente beinhalten eigentlichen Request mit
		Inputparametern. Aufruf der Methode
		destroyInstance([remotePortletKey],[remotePageKey],Consumer Portal,geheim,1)-->
		<g8:destroyInstance>
			<remotePortletKey xsi:type="xsd:string">
				[remotePortletKey]
			</remotePortletKey>
			<remotePageKey xsi:type="xsd:string">[remotePageKey]</remotePageKey>
			<user xsi:type="xsd:string">Consumer Portal</user>
			<pwd xsi:type="xsd:string">geheim</pwd>
			<device xsi:type="xsd:int">1</device>
		</g8:destroyInstance>
	</soapenv:Body>
</soapenv:Envelope>
```

Response des Service-Portals auf den Request des Consumer-Portals

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
	<soapenv:Body>
		<!--Anwendungsspezifische Elemente beinhalten eigentliche Response mit
		Rückgabewert destroyInstanceReturn.-->
		<g8:destroyInstanceResponse>
			<!--Flag zeigt an, ob Portlet-Instanz erfolgreich gelöscht wurde.-->
			<destroyInstanceReturn xsi:type="xsd:boolean">
				true
			</destroyInstanceReturn>
		</g8:destroyInstanceResponse>
	</soapenv:Body>
</soapenv:Envelope>
```

Zur besseren Einordnung der Nachrichten und deren Zusammenhänge werden Zustände und erlaubte Aufrufreihenfolgen durch ein Statechart-Diagramm visualisiert. Es umfasst eine vollständige Aufführung der implementierten Operationen, die im Zusammenhang mit einer Portlet-Verwaltung bedeutend sind. Die vorher nicht erklärten Methoden werden an dieser Stelle erläutert.

Die Erzeugung einer neuen Portlet-Instanz auf Basis eines bestehenden Portlets wird durch „createPortletClone()“ unterstützt. Die Operation „invokePortletAction()“ reicht Portlet-Events weiter. Die An- bzw. Abmeldung am Service-Portal wird durch „createSession()“ respektive „destroySession()“ implementiert.

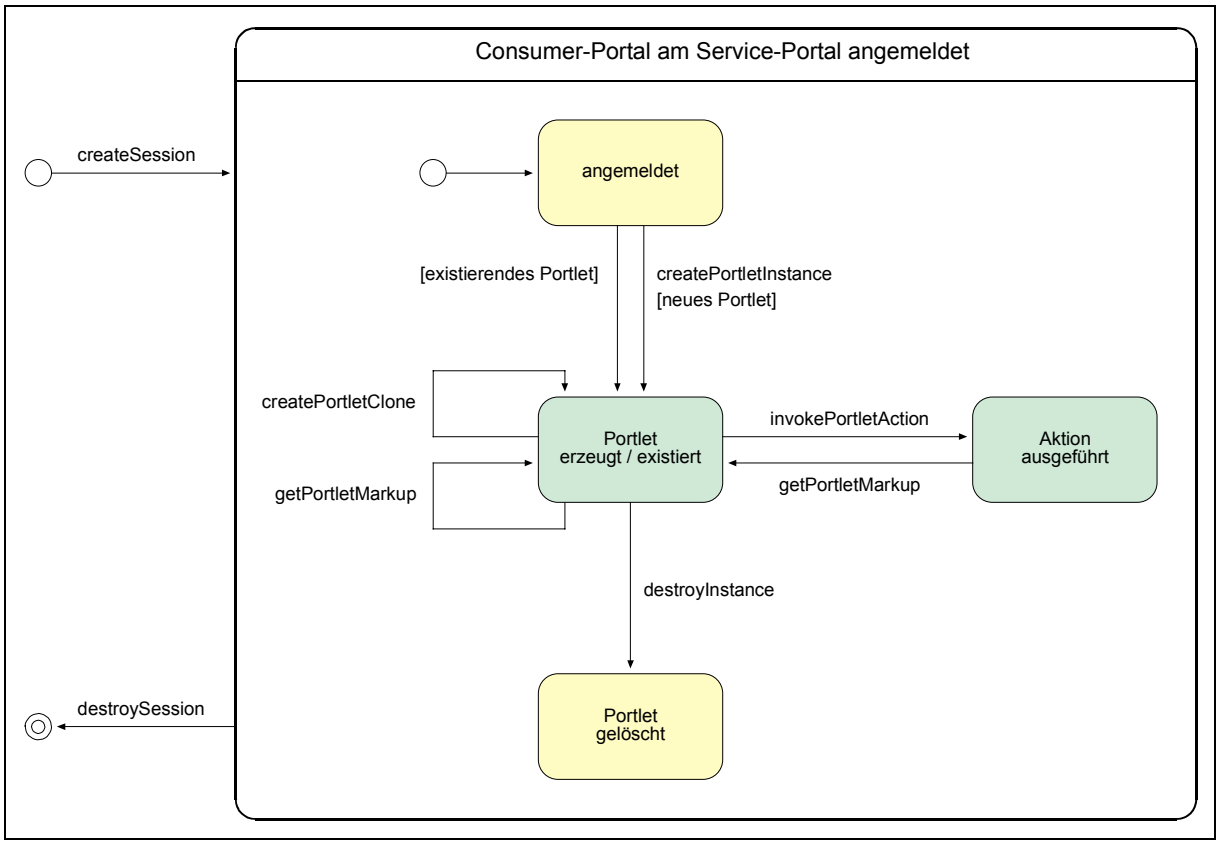

Abbildung 5: Statechart API-Aufrufe

5 Ausblick

5.1 Web Services Security (WS-Security)

Bei der Entwicklung von Standalone-Applikationen kann die Sicherheitskomponente vernachlässigt werden. Jedoch spielt der Aspekt der Vernetzung in der heutigen Informationstechnologielandschaft eine entscheidende Rolle. Moderne Architekturen für komplexe Aufgaben basieren auf mehreren Schichten, die untereinander über Netzwerke kommunizieren. Beispielsweise werden Präsentations-, Anwendungs- und Datenhaltungsschichten gekapselt. Die verschiedenen Schichten bieten Schnittstellen an, um ihre Funktionen den jeweils anderen Schichten bereitzustellen. Die Vorteile liegen in der leichteren Wartbarkeit und Erweiterbarkeit sowie in der Steigerung der Performance der Systeme. Der Datenaustausch erfolgt über Inter-, Intra- und/oder Extranet. Web Services gehen per Definition einher mit zunehmender Vernetzung, so dass die Bedeutung von Sicherheit exponentiell zunimmt.

Durch Vernetzung wird Personen die Möglichkeit gegeben, auf Systeme einzuwirken und Schaden anzurichten. Einer materiellen bzw. immateriellen Schädigung[8] muss entgegengewirkt werden. Somit darf Sicherheit nicht erst bei Missbrauch des Zugriffs nachgerüstet werden und sollte im Prozess der Entwicklung der Applikationen berücksichtigt und integriert werden. Davon hängt auch die Zuverlässigkeit des Systems ab, also die Qualität des Services.
Web Services, die auf dem SOAP-Protokoll aufbauen, verlangen nicht zwangsläufig nach Sicherheitsmechanismen. Es ist vorgesehen, diese über Erweiterungen dem Protokoll hinzuzufügen.

Eine zentrale Anforderung an WS-Security kann wie folgt definiert werden.
„The Web service security language must support a (...) end-to-end message-level security and not just transport-level security" (vgl. [Atkinson et al. 2002] S.4).
Bei der Realisierung von WS-Security kann auf kein festes Schema zurückgegriffen werden.
„Understanding that security is not an "all-or-nothing" choice, but that it has to be carefully assessed for each particular situation, makes security a complex challenge." (vgl. [Cauldwell et al. 2001] S.560).
Sicherheit kann auf zwei unterschiedlichen Ebenen bei Applikationen, die auf Web Services basieren, verwirklicht werden. Dabei sind die Transportebene und die Applikationsebene zu betrachten. Die Ebenen ergänzen einander. Initiativen, die Sicherheitsmechanismen innerhalb von XML standardisieren, erweitern bestehende Konzepte.

[8] Stehlen von Gedankengut, Markennamen, Schädigung der Reputation

Wichtige Aspekte, die bei einer Implementierung von WS-Security beachtet werden müssen, werden kurz vorgestellt.
Beim Authentifizierungsprozess weist sich ein Benutzer, Computer oder eine Applikation gegenüber einem anderen Benutzer, Computer oder einer Applikation aus. Bei Web Services findet dies typischerweise zwischen Applikationen statt. Dazu werden Benutzername oder Rechnername, Passwort und Zertifikate verwendet.
Autorisierung schließt sich an die Authentifizierung an. Sie beinhaltet die Verwaltung und Überprüfung der Zugriffsrechte. Eine Realisierung erfolgt auf Applikationsebene.
Das Protokollieren der Zugriffe auf die verwalteten Ressourcen dient als Werkzeug, um Informationen zur Problembehebung in einem System zu erhalten. Analysen lassen sich auf diese Weise qualitativ hochwertiger gestalten.
Kryptographie hilft Integrität zu erreichen, indem verhindert wird, dass Daten von nicht befugten Personen manipuliert oder gelesen werden.

Eine Kombination des HTTP Basic Authentication Mechanismus mit Secure Socket Layer (SSL) bietet sich an. HTTP Basic Authentication wird auch als Passwort Authentifikation bezeichnet, da ein Benutzername mit zugehörigem Passwort gegenüber einer Access Control List (ACL) verifiziert wird. Indem ein Angreifer das Netzwerk abhört und somit Benutzername und Passwort erhält, kann diese Sicherheit umgangen werden. Deshalb wird SSL als Protokoll für die Datenübertragung hinzugenommen, um auf diesem Weg die Daten durch kryptographische Methoden zu schützen. SSL ist zwischen der Applikations- und der Transportebene angesiedelt. Die Kombination von HTTP mit SSL wird als HTTPS bezeichnet. Ein Nachteil von SSL ist, dass immer nur die ganze Nachricht verschlüsselt wird. In einigen Fällen ist es dagegen erwünscht, lediglich Teile der SOAP-Nachricht zu verschlüsseln. SOAP-Encryption wirkt diesem Problem entgegen. Die Standardisierungsbemühungen sind allerdings noch nicht abgeschlossen.

5.2 Places

Web Services haben das Potential, Portale auf unterschiedlichen Ebenen zu verknüpfen. An dieser Stelle sei beispielsweise auf die Places Architektur hingewiesen. Dabei handelt es sich um eine Aggregationsebene über der des bisherigen Seitenregisters. Das bisherige Seitenregister wird sich dann unter dem Home-Place des Benutzers wieder finden und soll in seiner jetzigen Form beibehalten werden. Places stellen eine flexible Arbeitsumgebung mit diversen Einbindungsmöglichkeiten dar. Dieser Ansatz könnte das G8-Portal um Groupware-Systemklassen wie den Shared-Informationsspace oder auch das Workflow Management erweitern. Im Allgemeinen kann festgehalten werden, dass vielfältige Funktionalitäten und Eigenschaften von Places vorstellbar sind. Der Websphere Portal Server bietet bereits eine

Places Architektur, allerdings ist der Code nicht zugänglich und eignet sich somit nicht für die Selbstentwicklung und Forschung. Das G8-Portal wird zukünftig um eine eigene leistungsfähige Places Struktur erweitert. Diese stellt die Basis für eine weitergehende Kopplung von Portalen auf der Ebene der Places oder auch Seiten dar. Weitere Bestrebungen werden auf eine vollständige Kopplung von Portalen abzielen.

6 Fazit

Das Fundament dieser Arbeit stellt das G8-Portal dar. Der Fokus der Projekte, die auf dieser Plattform aufbauten, lag in den Jahren 2000 und 2001 stärker auf der Integration von Inhalten, von Applikationen und der Anbindung an Geräte wie Personal Digital Assistants (PDA) oder Handy. Viele Grundfunktionalitäten wurden bereits realisiert. Um mehr Flexibilität und höhere Wiederverwendbarkeit zu erreichen, führte der nächste Schritt zur Portal-Verknüpfung.

Ziel dieser Seminararbeit war die Konzeption und Umsetzung der Kopplung von Portal-Diensten auf Portlet-Ebene mit Hilfe von Web Services. Dieses Ziel wurde sowohl konzeptionell als auch in einer prototypischen Implementierung erreicht.

Die Lösung bietet eine manuelle Kopplung über Konfigurationsdokumente. Sind die Portale verknüpft, werden die Inhalte automatisch aktualisiert. Es handelt sich um eine lose Kopplung in Form von Verweisen auf das jeweils beteiligte Portal, bei der kein abgestuftes Authentifizierungsmanagement auf Benutzerebene unterstützt wird. Die Kommunikation erfolgt auf Basis von Nachrichten, die zwischen den Portalen ausgetauscht werden. Dieses Konzept ist offen und bietet somit vielfältige Erweiterungsmöglichkeiten im Hinblick auf mögliche Einsatzgebiete.

Im Speziellen werden Portal-Dienste als Portlets von einem Service-Portal angeboten. Dies geschieht über ein Konfigurationsdokument, das analog zu denen von anderen Portlets aufgebaut ist und spezifische Informationen beinhaltet, die einen Remote-Zugriff ermöglichen. Diese Portlets werden beim Consumer-Portal über dieselbe Schnittstelle wie die internen Portlets eingefügt. Administratoren können ohne Einarbeitungszeit und genauer Kenntnis der Funktionsweise eine Portal-Kopplung herstellen.

Dem Benutzer bleibt die Quelle des Portlets weitgehend verborgen. Er muss sich nicht gegenüber dem Service-Portal identifizieren. Somit ist eine ausreichende Transparenz der Lösung gewährleistet.

Derzeit umfasst das G8-Portal noch keine Places. Diese Portal-Ebene fand folglich keine Berücksichtigung.

Das betriebswirtschaftliche Potential ist reichhaltig. Die Portal-Kopplung unterstützt zwei zentrale Ziele. Ein Ziel ist die Einbindung von externen Ressourcen, die ansonsten nicht zugänglich wären. Der konkrete Nutzen hängt sehr stark vom Einsatzgebiet ab. Ein weiteres Ziel ist die Schaffung einer offenen Architektur. Es lässt sich tendenziell festhalten, dass mit steigender Kooperation die Qualität der Informationen zunimmt und der Aufwand zur Bereitstellung abnimmt.

Diese Seminararbeit bietet eine gute Basis, um weitere Portal-Ebenen zu koppeln. Konzepte und Module können in weiterführenden Arbeiten komplett wiederverwendet oder adaptiert werden.

7 Literaturverzeichnis

[Atkinson et al. 2002]
Atkinson, B. et al. : Web Services Security (WS-Security), Version 1.0, 5. April;
(http://www-106.ibm.com/developerworks/library/ws-secure/, 26.09.2002)

[Biermann 2001]
Biermann, I.: Portale als Schnittstelle für organisationsübergreifende Kooperation - Konzept und Implementierung eines XML-basierten Wissenstransfers zwischen Intranetportalsystemen. Diplomarbeit, Universität Paderborn, Fachbereich Wirtschaftswissenschaften, Lehrstuhl für Wirtschaftsinformatik 2, April 2001.

[Bruse 2001]
Bruse, T.: Workplace Portal G8, 2001
(Knowledge-Pool: http://pbfb5www.uni-paderborn.de/www/WI/WI2/wi2_lit.nsf/663247270b635985c1256bc900519bef/2f2524f00e9d1f01c1256a16004583a8?OpenDocument, 16.10.2002)

[Cauldwell et al. 2001]
Cauldwell, P. et al.: Professional XML Web Services, Wrox Press Ltd., Birmingham, 2001

[Fraunhofer Institut 2002]
o.V.: Fraunhofer Institut Arbeitswirtschaft und Organisation (2002): Marktübersicht Portal Software, 2002;
(http://www.portale.iao.fhg.de/Marktuebersicht.cfm am 17.08.2002)

[Graham et al. 2002]
Graham, St. et al.: Building Web Services with Java – Making Sense of XML, SOAP, WSDL, and UDDI, Sams Publishing, Indianapolis, 2002

[Hahnl 2000]
Hahnl, O.: Personalisierte Portaltechnologien auf Basis einer prozessgetriebenen Groupware-Umgebung - Konzeption und Realisierung einer offenen Architektur unter Verwendung von Lotus Domino und Oracle8i. Diplomarbeit, Universität Paderborn, Fachbereich Wirtschaftswissenschaften, Lehrstuhl für Wirtschaftsinformatik 2, Oktober 2000.

[IBM 2001]
Kreger, H.: Web Services Conceptual Architecture (WSCA 1.0), 2001
(http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf am 20.09.02)

[Microsoft 2002 a]
o.V.: XML Web Services - The XML Web Services Developer Center Replaces the Web Services Node Page;
(http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000442, 26.09.2002)

[Microsoft 2002 b]
Kirtland, M.: A Platform for Web Services;
(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnWebsrv/html/Websvcs_platform.asp, 26.09.2002)

[Nußdorfer 2002].
Nußdorfer, R.: IT-Infrastruktur – Web-Services-Standards koppeln Geschäftsprozesse übergreifend, Computer Zeitung, Heft 035, Seite 16, 2002;
(http://www.computer-zeitung.de, 27.09.2002)

[Röwekamp 2002]
Röwekamp, L.: Dienste leisten: Java Web Service Tutorial –Teil 1: Grundlagen. In: Magazin für professionelle Informationstechnik (IX), Ausgabe 9, Seite 121 f., 2002

[Schmidt 1999]
Schmidt, C.: Konzeption eines Groupware-basierten „Enterprise Knowledge Portals“ - Entwicklung und Diskussion eines Lösungsvorschlags und Übertragung auf das Lotus Notes-basierte Intranet der Deutschen Bank AG. Diplomarbeit, Universität Paderborn, Fachbereich Wirtschaftswissenschaften, Lehrstuhl für Wirtschaftsinformatik 2, Dezember 1999.