DIPLOMARBEIT

Enterprise JavaBeans für die Entwicklung Web-basierter Anwendungen – am Beispiel eines Bürgerprozessportals basierend auf Serviceflow Management (SfM)

Vorgelegt von

NOL SHALA

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Mai 2002

Erstbetreuer:
Dr. Ralf Klischewski

Zweitbetreuer:
Prof. Dr. Heinz Züllighoven

Erklärung:

Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den 27.05.2002

Nol Shala
Lippmannstr. 7
22769 Hamburg
Matr. Nr.: 4675108
Tel.: (040) 56 61 61
Email: 3shala@informatik.uni-hamburg.de

Betreuung:
Dr. Ralf Klischewski (Erstbetreuung)
Prof. Dr. Heinz Züllighoven (Zweitbetreuung)

Dr. Ralf Klischewski
Arbeitsbereich Softwaretechnik (SWT)
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22175 Hamburg

Prof. Dr. Heinz Züllighoven
Arbeitsbereich Softwaretechnik (SWT)
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22175 Hamburg

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

Das Web als Kommunikationsmedium im Internet/Intranet bietet dem Nutzer eine unüberschaubare Fülle von Informationen und Diensten. Um diese strukturiert und verständlich darzustellen werden Web-basierte Anwendungen entwickelt.

Web-basierte Anwendungen sind von den Anfängen des Internets bis heute durch ganz verschiedene Entwicklungen und Techniken geprägt worden. Neben der sogenannten Ein-Schicht-Architektur, einfaches HTML, sind heutzutage vielfältige Lösungen mit mehr als einer Architektur-Schicht im Internet zu finden. Zahlreiche Client- sowie Server-seitige Programmier- und Skriptsprachen bereichern die Web-basierte Anwendungsentwicklung. Der Komplexitätsgrad der Web-basierten Anwendungen steigt hierbei zudem noch durch die Verwendung von unterschiedlichen Hardware- und Softwarelösungen.

Da die Bedeutung des Internets einen immer größeren Einfluss auf die Softwareindustrie nimmt, werden an Web-basierte Anwendungsentwicklungen laufend neue Anforderungen gestellt. Im Vordergrund stehen vor allem Skalierbarkeit, Robustheit, Portabilität, Sicherheit, Wiederverwendung und Integration. Einige dieser softwaretechnischen Qualitätskriterien sind in der bisherigen Entwicklung Web-basierter Anwendungen nur zum Teil berücksichtigt worden.

Neue Web-Technologien, wie z.B. Enterprise JavaBeans (EJB), Teil der Java 2 Enterprise Edition (J2EE), erheben den Anspruch die Entwicklung Web-basierter Anwendungen durch die oben erwähnten softwaretechnischen Qualitätskriterien zu ergänzen. Aufgrund der robusten server-seitigen Architektur der Enterprise JavaBeans bietet sich deren Einsatz für die Realisierung von komponentenbasierten unternehmenskritischen Web-Anwendungen im Internet an.

Nun aber viele Unternehmen stehen vor der Frage, ob der Einsatz von EJB tatsächlich eine Verbesserung gegenüber anderen Technologien in der Entwicklung der Web-basierten Anwendungen mit sich bringt. Dabei stellen die Besonderheiten der Client-Server Architektur der Web-basierten Anwendungen sowie die vorhandenen Systeme und Infrastrukturen, aber auch die Qualifikation der Entwickler, eine zusätzliche Herausforderung für eine Entscheidungsfindung dar.

1.2 Fragestellung

Welche Vorteile bietet der Einsatz der Enterprise JavaBeans (EJB) in der Praxis der Web-Entwicklung? (z.B. In welchem Kontext bzw. Projektumfang empfiehlt sich der Einsatz der EJB?)

Dies soll an dem Fallbeispiel des Bürgerprozessportals „Prozessportal für Serviceflow Management/eGovernment Hamburg“ untersucht werden. Die konkrete Frage der Arbeit lautet:

Inwiefern verbessert eine Re-Implementierung auf Basis von EJB die Qualität der Anwendungslogik des Web-basierten Serviceflow Management Prototyps im Hinblick auf ihren robusten und skalierbaren Einsatz?

1.3 Vorgehensweise

Um die oben genannten Fragen zu beantworten werden als erstes die Techniken zur Entwicklung von Web-basierten Anwendungen zusammenfassend dargestellt. Dabei werden die Technologien und Protokolle, die die Entwicklung von Web-basierten Anwendungen geprägt haben, mit all ihren Vor- und Nachteilen erörtert. Die Zusammenhänge und Anhaltspunkte der Entwicklung von Web-basierten Anwendungen bezüglich der klassischen traditionellen Vorgehensweise der Softwareentwicklung werden kurz diskutiert. Welche softwaretechnischen Entwurfsprinzipien und Qualitätskriterien als Anforderungen und Kriterien für die Entwicklung Web-basierter Anwendungen in den Vordergrund gestellt werden sollten, wird erörtert.

Heutzutage werden viele Plattformen und Technologien für die Entwicklung von Web-basierten Anwendungen angeboten. Als Teil der Java 2 Enterprise Edition (J2EE) versprechen die Enterprise JavaBeans (EJB) eine server-seitige Architektur mit besserer Skalierbarkeit und Portabilität sowie ein sicheres und robustes Transaktions- und Persistenzkonzept. Zur Beantwortung der zentralen Fragestellung der Arbeit wird zunächst die EJB-Architektur detailliert besprochen. Die Vorteile des EJB-Komponentenmodells hinsichtlich der Skalierbarkeit, Robustheit, Portabilität, Sicherheit, Wiederverwendung und Integration wird untersucht. Für den Gesamtkontext der Fragestellung wird der Projektverlauf gemeinsam mit dem Serviceflow Management Leitbild und der Entwurfsmetapher der Fachlichen Services sowie dem Ist-Zustand der Web-basierten SfM Architektur, dabei insbesondere der Anwendungslogik, näher besprochen und untersucht. Der Web-basierte Serviceflow Management (SfM) Prototyp aus dem Projektseminar *Service als Leitbild: Softwareunterstützung für Dienstleister „die elektronische Beantragung von Briefwahlunterlagen bei der Stadt Hamburg über http://www.hamburg.de“* dient als Fallbeispiel für die Schwachstellenanalyse mit dem Schwerpunkt auf der Anwendungslogik.

Eine Verbesserung und Optimierung des alten Web-basierten SfM Prototyps bezüglich seiner Schwachstellen wird durch den Einsatz von EJB durchgeführt. Darauf folgend wird eine Bewertung nach softwaretechnischen Entwurfsprinzipien und Qualitätskriterien der EJB Implementierung vorgenommen. Aus den Erkenntnissen der Re-Implementierung nach der EJB-Vorgehensweise des Web-basierten SfM Prototyps werden die Vorteile und Nachteile der EJB in der Praxis, besonders im Bezug auf das Fallbeispiel aus dem eGovernment-Bereich, aufgezeigt. Abschließend werden die Ergebnisse, die bei der Entscheidungsfindung hinsichtlich der Anwendbarkeit der EJB-

Architektur in einem Unternehmen bzw. einem Projektumfang in Betracht gezogen werden sollten, präsentiert.

1.4 Gliederung

Kapitel 2, ***Einsatz von EJB in der Entwicklung Web-basierter Anwendungen***, behandelt die Domäne dieser Arbeit. Im Vordergrund steht das Konzept der EJB sowie dessen Möglichkeiten für den Einsatz in der Entwicklung Web-basierter Anwendungen.

> Kapitel 2.1, ***Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen***, stellt allgemeine Merkmale, Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen vor. Die Besonderheiten und Techniken der Entwicklung Web-basierter Anwendungen werden kurz umrissen. Des weiteren wird die Entwicklung Web-basierter Anwendungen mit Anforderungen und Kriterien sowie softwaretechnischen Problemen konfrontiert.
>
> Kapitel 2.2, ***EJB-Konzepte***, gibt einen detaillierten Überblick der EJB-Architektur. Die Merkmale der EJB-Architektur werden als erstes kurz vorgestellt. Anschließend wird der Anwendungskontext der EJB präsentiert und auf die einzelnen Bestandteile der Architektur: EJB-Server, EJB-Container, EJBean und EJB-Client näher eingegangen. Die Typen von EJB: Session, Entity und Message-Driven Beans und deren wesentliche Eigenschaften werden erläutert. Das Transaktions- und Persistenzmanagement-Konzept sowie die EJB QL der EJB-Spezifikation werden ausführlich beschrieben. Das Kapitel schließt mit den Einschränkungen der EJB-Spezifikation ab.
>
> Kapitel 2.3, ***Web-Komponenten und EJB***, gibt das Zusammenspiel der EJB und der Web-Komponenten im Bezug auf die Erstellung von Web-basierten Unternehmensanwendungen wieder.
>
> Kapitel 2.4, ***EJB in der Praxis?***, stellt am Schluss des zweiten Kapitels zusammenfassend einige Vorteile und Nachteile der EJB für die Praxis vor.

Kapitel 3, ***EJB für ein Bürgerprozessportal***, erörtert die Eignung der EJB für den Einsatz in einem Bürgerprozessportal.

> Kapitel 3.1, ***eGovernment-Bürgerprozessportal: Einführung und allgemeine Anforderungen***, gibt die Möglichkeiten des Einsatzes der Web-basierten Anwendungen auf der Regierungs- und Verwaltungsebene wieder. Die allgemeinen Anforderungen und Besonderheiten eines eGovernment-Bürgerprozessportals werden dargestellt.
>
> Kapitel 3.2, ***Der Web-basierte SfM Prototyp***, schildert als erstes den Serviceflow Management (SfM) Ansatz, der als theoretische Grundlage für die Realisierung des Web-basierten Prototyps verwendet wurde. Der Projektverlauf und der Ist-Zustand des Web-basierten SfM Prototyps am Fallbeispiel „Prozessportal für

Serviceflow Management/eGovernment Hamburg“ wird dabei vorgestellt. Eine Schwachstellenanalyse hinsichtlich der Skalierbarkeit, Robustheit, Portabilität, Sicherheit, Wiederverwendung und Integration des Web-basierten SfM Prototyps, insbesondere der Anwendungslogik, wird durchgeführt.

Kapitel 3.3, ***Architektur der mit EJB realisierten SfM Anwendungslogik***, untersucht die Eignung von EJB für die Realisierung eines Prozessportals. Im Anschluss daran wird die Architektur des in EJB realisierten Web-basierten SfM Prototyps bzw. der Anwendungslogik demonstriert. Eine Bewertung des mit EJB realisierten Web-basierten SfM Prototyps nach softwaretechnischen Entwurfsprinzipien und Qualitätskriterien sowie eine Eignung von EJB hinsichtlich des Anwendungskontexts eGovernment-Bürgerprozessportals schließen das dritte Kapitel ab.

Kapitel 4, ***Resümee***, gibt die Ergebnisse und Antworten auf die Fragestellung zusammenfassend wieder. Der Ausblick beendet diese Arbeit.

1.5 Danksagung

Ich möchte mich an dieser Stelle ganz herzlich bei all denjenigen Personen bedanken, die mir bei der Erstellung dieser Arbeit behilflich waren.

Mein großer Dank geht zunächst an Herrn Dr. Ralf Klischewski für die Erstbetreuung, der mich während dieser Arbeit mit zahlreichen Hilfestellungen und Denkanstößen unterstützt hat. Bei Herrn Prof. Dr. Heinz Züllighoven bedanke ich mich für die Zweitbetreuung.

Des weiteren möchte ich mich bei Timmy Blank, mit dem ich im Rahmen des Projektseminars „*Service als Leitbild: Softwareunterstützung für Dienstleister*“, die Anwendungslogik des funktionalen Web-basierten SfM Prototyps realisiert habe, bedanken.

Martti Jeenicke und Wolf-Gideon Bleek vom Arbeitsbereich SWT danke ich für Diskussionen, Literaturhinweise, Feedback und „last but not least“ für den tollen Kaffee.

Ein ganz besonders großer Dank geht an meine Ehefrau Mirjam, die das Korrekturlesen dieser Arbeit gemacht hat. Außerdem war sie mir und unserer kleinen Familie während der ganzen Zeit eine große Stütze.

Bei meinem Sohn Elion bedanke ich mich für die seelische und geistige Entspannung und vor allem das köstliche Lachen, das er mir bescherte.

2 Einsatz von EJB in der Entwicklung Web-basierter Anwendungen

In diesem Kapitel wird als erstes das HTT-Protokoll, das als Kommunikationsprotokoll für die Client- und Server-seitigen Web-basierten Anwendungen genutzt wird, erläutert. Daneben wird auf die Vielfältigkeit der Techniken und Technologien für die Entwicklung Web-basierter Anwendung eingegangen. Welche Anforderungen und Kriterien an die Web-basierte Anwendungsentwicklung gestellt werden, insbesondere durch den Einsatz von EJB, und welche softwaretechnischen Fragen und Probleme dabei entstehen können, wird erörtert.

Im Spektrum der verschiedenen Web-Technologien für die Entwicklung Web-basierter Anwendungen wird im Kapitel 2.2 detailliert auf das Konzept der Enterprise JavaBeans (EJB) eingegangen. Die wichtigsten Merkmale des EJB-Konzepts sowie der EJB-Architektur werden beschrieben. Danach wird im einzelnen auf das EJB-Komponentenmodell eingegangen. Als Teil des EJB-Komponentenmodells werden die drei EJB-Typen erläutert. Anschließend werden das explizite und implizite Transaktions- und Persistenzmanagement sowie die EJB QL für die Realisierung Web-basierter Anwendungen besprochen. Welche Einschränkungen die EJB-Spezifikation aufweist, wird am Ende dieses Kapitels verdeutlicht.

Im Anschluss daran wird im Kapitel 2.3 das Zusammenspiel der Web-Komponenten und EJB näher erläutert. Als letztes werden im Kapitel 2.4 die Vorteile und Nachteile der EJB für die Praxis genannt.

2.1 Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen

Das World Wide Web[1] (WWW) öffnet neue Wege und stellt neue Herausforderungen an die klassische Softwareentwicklung, besonders an die Vorgehensweise und die Wahl der Methoden für die Entwicklung Web-basierter Anwendungen. Neue Fragen und Probleme bei der Ermittlung der Anforderungen und Kriterien der Web-basierter Anwendungsentwicklung werden aufgezeigt. Dazu kommt, dass einige softwaretechnische Entwurfsprinzipien und Qualitätskriterien in der Entwicklung Web-basierter Anwendungen immer mehr an Bedeutung gewinnen.

Das World Wide Web basiert auf einer Client-Server Architektur. Der Web-Browser[2] dient als universelles Frontend für das World Wide Web. Zur Kommunikation wird das HTT-Protokoll (*Hypertext Transfer Protocol, HTTP*) verwendet [FIELDIN99]. HTTP stellt ein Protokoll für die Anwendungsebene dar, das besonders für verteilte, kooperative, Hypermedia Informationssysteme geeignet ist. Es ist ein generisches, zustandsloses Protokoll, das für weit mehr als einfache Text-basierte (*Hypertext*) Aufgaben verwendet werden kann. So dient es unter anderem als Kommunikationsmittel

[1] Im Jahre 1990 begann Tim Berners-Lee am CERN das Projekt World Wide Web [CONNOL00]

[2] Als Web-Browser (auch WWW-Browser) bezeichnet man ein Client-Programm für den Zugriff auf den Web-Server. Der Web-Server ist für die Bereitstellung der Daten zuständig.

zwischen User-Agenten[3] und Proxies[4] bzw. Gateways[5] zu anderen Informationssystemen. Ein weiteres Merkmal des HTTPs ist die Typisierung und die Darstellung der Datenrepräsentation. HTTP–Kommunikation findet gewöhnlich über TCP/IP Verbindungen statt.

Das HTT-Protokoll ist ein *Request - Response* Kommunikationsprotokoll (s. Abbildung 1). Der Web-Client sendet in Form einer *Request-Methode,* einen *Request* an den Web-Server, und der Web-Server antwortet mit einer *Response-Message*. Sendet der Web-Client eine Anfrage an den Web-Server, fordert er zum Beispiel ein Dokument[6] an, so wird eine Verbindung mit dem Web-Server hergestellt, das Dokument übertragen und die Verbindung wieder beendet. Zur Identifikation von Dokumenten wird ein spezielles Adressierungsschema *URI*[7] (*Uniform Resource Identifiers*) verwendet. Dadurch ist jedes Dokument eindeutig über *URI* identifizierbar. Mit der Eingabe der Web-Adresse als *URL* (*Uniform Resource Locator*) ruft der Web-Client das Dokument auf.

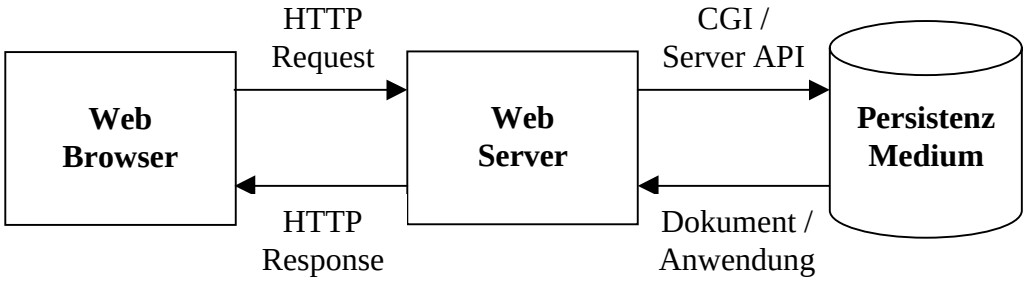

Abbildung 1 - HTT-Protokoll ein Request/Response Kommunikationsprotokoll

Die meisten Dokumente im Web sind mit der Auszeichnungssprache *HTML* (*Hypertext Markup Language*) beschrieben. Bei einer Web-basierten Anwendung werden nur die Inhalte in Form einer Datei oder eines Dokumentes übertragen und dargestellt. Hierzu unterscheidet man in Abhängigkeit des Ausführungsortes in server- und client-seitige Web-basierte Anwendungen. Server-seitige Anwendungen sind für die Erzeugung bzw. Bereitstellung der Daten verantwortlich. Sie ermöglichen den Zugriff auf Daten, welche nicht statisch vorliegen und nicht mit HTML ausgezeichnet sind. Der Web-Browser wird im Prinzip nur zur Ein- und Ausgabe verwendet. Client-seitige Anwendungen sind dagegen mit Hilfe der verschiedenen Techniken wie z.B. der Skriptsprache „*Javascript*“ (vgl. [FLANAG98]) für die Repräsentation der Daten und die Benutzerinteraktionen zuständig (vgl. [TURAU99]).

In dieser Arbeit liegt der Schwerpunkt auf den server-seitigen Web-basierten Anwendungen. Im weiteren Verlauf der Arbeit werden diese der Einfachheit halber Web-

[3] Der Client der den HTTP-Request initiiert. Dies können öfters die Web-Browser, Editoren, Spiders (web-travesing robots) oder End-User Tools sein.
[4] Ein vermittelndes Programm, welches für andere Klienten sowohl als Server als auch Client agieren kann.
[5] Ein für andere Server vermittelnder Server.
[6] Ein beliebiges Dokument, wie Bild, Music- oder Video-Clip, ist hierunter zu verstehen.
[7] Auch bekannt als: WWW-Adressen, Universal Document Identifiers, Universal Resource Identifiers und schließlich als Kombination aus Uniform Resource Locators (URL) mit Uniform Resource Names (URN).

basierte Anwendungen genannt. Unter Web-basierten Anwendungen wird in dieser Arbeit folgendes Verständnis bzw. folgende Sicht vertreten:

Die Web-basierte Anwendung läuft server-seitig und verwendet das HTT-Protokoll auf der Kommunikationsebene. Dabei dient der Web-browser als universales Frontend, das Anfragen entgegennimmt und an die server-seitige Web-basierte Anwendung weiterleitet sowie Antworten über HTTP an den Web-browser zurückliefert. Bei dem Entwurf und der Realisierung der Web-basierten Anwendung ist sicherzustellen, dass die server-seitige Web-basierte Anwendung nach den Ansätzen der softwaretechnischen Entwurfsprinzipien und Qualitätskriterien implementiert wird.

Bei Web-basierten Anwendungen können die Daten aus komplexen Informationsquellen (wie z.B. Datenbanken, File System) stammen oder dynamisch zur Laufzeit erzeugt werden. Dies wird entweder durch den Aufruf eines externen Programms oder Skripts, das z.B. in Perl, Tcl, C, etc. programmiert wurde, oder entsprechend durch die Erweiterung der Serverfunktionalität realisiert. Der Unterschied zwischen diesen zwei Techniken beruht auf Verwendung bestimmter Mechanismen sowie der Ressourcenverwaltung und -benutzung:

- Bei der Technik des Aufrufs einer Anwendung über das externe Programm kommt bei einer Web-Server-Anfrage ein durch das CGI[8] (*Common Gateway Interface*) aufgebautes Gateway, das die Kommunikation zwischen dem Web-Server und dem Programm kanalisiert, zustande. Diese Kommunikation ermöglicht den Zugriff auf die Informationsquelle, so dass die Information für einen Web-Client wie eine Datei (oder Dokument) auf einem Web-Server erscheint. Für jede Anfrage an ein Gateway wird ein neuer Prozess für das Gateway-Programm gestartet. Diesem Prozess werden die vom Web-Client erhaltenen Eingabedaten und Umgebungsvariablen übergeben. Die Antwort des Gateways liest der Web-Server von der Standardausgabe des Prozesses und übermittelt diese mit Hilfe des Web-Servers an den Web-Browser des Web-Clients. Danach wird der Prozess beendet. Der Nachteil dieser Technik ist, dass der Web-Server für jede CGI-Anfrage einen eigenen Prozess erzeugen muss. Der einhergehende Verbrauch von Betriebssystemressourcen kann sich somit negativ auf die Verfügbarkeit des Servers auswirken (vgl. [TURAU99]).

- Bei der Technik der Erweiterung der Serverfunktionalität lädt der Web-Server dagegen bei der ersten Anfrage die dynamische Bibliothek (oder Klasse). Diese verbleibt (nach Möglichkeit) im Speicher und kann so bei weiteren Anfragen genutzt werden. Es kann eine beliebige Anzahl von Anfragen gleichzeitig bearbeitet werden. Hierdurch entfällt der mit der Prozesserzeugung einhergehende Aufwand. Solche Bibliotheken (oder Klassen) arbeiten im gleichen Prozessraum wie der Web-Server und können so auf alle vom Server verwendeten Ressourcen zugreifen. Da die Kommunikation nicht nur auf die Web-Server Umgebungsvariablen und Standardeingabe/-ausgabe, sondern auch auf weitere bereitgestellte Bibliotheken (oder Klassen) erweitert wird, steigert dies die Effizienz dieser Technik. Diese

[8] Siehe http://www.w3.org/CGI/

Erweiterung der Serverfunktionalität[9] ist plattform- und serverunabhängig (vgl. [HUNTER01]).

In dieser Arbeit wird der Fokus der Web-basierten Anwendungsentwicklung auf die Technik der Erweiterung der Serverfunktionalität gesetzt. Dazu wird die Java 2 Enterprise Edition (J2EE) Architektur für die server-seitige Entwicklung von Web-basierten Anwendungen verwendet.

Bedingt durch die oben beschriebenen technischen Gegebenheiten des HTT-Protokolls, der Client-Server Architektur und der Verwendung des Web-Browsers als universales Frontend für Web-basierte Anwendungen weisen jene überwiegend folgende technische Einschränkungen und Nachteile auf:

⇒ HTTP ist zustandslos: Jede Anfrage eines Web-Clients findet den Web-Server im gleichen Zustand vor. Der Web-Server speichert keinen Interaktionszustand nachdem er eine Anfrage abgearbeitet hat (vgl. [GILESCH96]).

⇒ Einseitige Interaktion: Bei Web-basierten Client-Server Architekturen kann die Interaktion nur vom Web-Client ausgehen. Der Web-Server kann von sich aus nicht die Kommunikation anstoßen (vgl. [GILESCH96]).

⇒ Abhängigkeit von Server-Belastung: In Abhängigkeit von der Server-Belastung ist mit entsprechenden zeitlich schwankenden Antwortzeiten zu rechnen. Darüber hinaus ist im Falle eines Server-Ausfalls die Gefahr inkonsistenter Zustände eines Servers oder der Web-basierten Anwendung ziemlich hoch.

⇒ Anwendungsoberfläche ist an HTML-Spezifikation gebunden: Durch die Nutzung des Web-Browsers als universales Frontend in der Web-basierten Anwendung ist die Gestaltung der Anwendungsoberfläche stark an die HTML-Spezifikation gebunden.

Diese Einschränkungen und Nachteile, die auch als Web-Plattform-Charakteristika bezeichnet werden können, üben in der Entwicklung Web-basierter Anwendungen bei der Wahl der geeigneten softwaretechnischen Vorgehensweise sowie bei der Ermittlung der Anforderungen und Kriterien einen bedeutsamen Einfluss aus. Darüber hinaus wirft die Aneignung einer Web-Technologie nach [NAWA99], die ständiger Weiterentwicklung unterläuft, neue Fragen und Probleme für die softwaretechnische Vorgehensweise sowie Ermittlung der Anforderungen und Kriterien in der Entwicklung Web-basierter Anwendungen auf. Diese Fragen und Probleme werden von [NAWA99] in drei Bereiche abgegrenzt:

- Das Fehlen von Wissen über Technologien, da Web-Technologien nie vollständig entwickelt sind und sich ständig verändern (Technologiebezogener Bereich)

[9] Hiervon ausgeschlossen ist die Erweiterung der Serverfunktionalität durch NSAPI von Netscpae, ISAPI von Mircosoft, Apache Modulen, SSI und ähnlichen Produkten.

- Das Wissen über Prozesse und Methoden ist unzureichend, da der Entwurf von e-Anwendungen andere Herangehensweisen und spezifischere Kombinationen aus unterschiedlichen Ressourcen erfordert, als ein klassischer Systementwurf (Projektbezogener Bereich).

- Domänenwissen ist schwer zu erhalten, da e-Business Applikationen die Grenzen von einer Organisation überschreiten (Anwendungsbezogener Bereich).

Nicht nur die technischen Einschränkungen und Nachteile sowie die Weiterentwicklung der Web-Technologien werfen bei der Ermittlung von verlässlichen Anforderungen und Kriterien Fragen und Probleme auf, sondern es treten auch wiederholt einige Grundsatzfragen bei verschiedenen Web-basierten Anwendungsentwicklungsprojekten nach [BLJEKL02] auf:

- Wie können die (initialen) Anforderungen für netzbasierte Web-Anwendungen definiert werden?

- Wie können Anforderungen systematisch gesammelt werden, wenn die Anwendergruppe (die Web-Nutzer) unbekannt und in ihren Charakteristika schwer zu beschreiben sind?

- Welche Akteure sollten in den Prozess eingebunden sein und auf welche Weise?

- Was sind die Konsequenzen für die Anforderungsermittlung und den Entwicklungsprozess bei einer stetigen Weiterentwicklung des technischen Systems, das die Grundlage für die Anwendung darstellt?

Diese Fragen und Probleme wurden nach den traditionellen Ansätzen der Softwareentwicklung in der Entwicklung Web-basierter Anwendungen nur teilweise behandelt. Ausschlaggebend hierfür war die bisherige Ad-hoc[10] Vorgehensweise in der Entwicklung Web-basierter Anwendungen sowie die technischen Besonderheiten der Web-Plattform. Die Web-basierten Anwendungen waren lange Zeit vornehmlich Inhalts-orientiert und in vielerlei Hinsicht Design-angetrieben. Entscheidend hierfür war die andere Sichtweise auf das Medium Web-Plattform. Nach [MURUG01] werden neun Punkte als wesentliche Unterschiede zu der klassischen Softwareentwicklung hervorgehoben. Diese beziehen sich auf „look and feel“ orientierte Web-basierte Anwendungsentwicklung, die als Ziel überwiegend die statische Dokumentenlieferung hatte.

Ob sich traditionelle Ansätze der Softwareentwicklung ohne weiteres, unter Rücksichtnahme auf technische Besonderheiten der Web-Plattform und deren Technologievielfalt, auf die Entwicklung der Web-basierten Anwendungen übertragen lassen oder ob es einer neuen Vorgehensweise oder Methodik bedarf, ist

[10] „...applications were developed without methodological support, without the right tools, simply on the basis of good common sense and individual skills” [CODAET98]

Forschungsgebiet in vielen Universitäten und Unternehmen. Dennoch kann davon ausgegangen werden, dass diese zwei Bereiche sich immer mehr überschneiden werden und in näherer Zukunft unter dem Oberbegriff „Software-Engineering“ mit der entsprechenden Unterteilung in den Web- und Desktop-Bereich geführt werden.

Unter den oben genannten Gesichtspunkten ist eine vollständige Einhaltung der Vorgehensweise nach den klassischen Ansätzen der Softwareentwicklung sicherlich nicht einfach. Auf jeden Fall sollten Schritte wie Anforderungsermittlung, Systemgestaltung, Softwareentwurf, Programmierung, Funktions- und Leistungsüberprüfung, Installation und Wartung in jedem Web-Projekt bzw. in jeder Entwicklung Web-basierter Anwendungen durchgeführt werden. Selbst wenn ständig neue Anforderungen an die Entwicklung der Web-Anwendung gestellt werden und neue Technologien eingesetzt werden, ist eine geplante Vorgehensweise z.B. durch die häufigen Autor-Kritiker-Zyklen während des Entwicklungsprozesses nach dem Ansatz des Extreme Programming (nach [LIROWO02]) oder die Vorgehensweise nach dem e-Prototyping Ansatz (nach [BLJEKL02]), etc. unerlässlich.

Eine pragmatische Vorgehensweise, um Kriterien aus den Anforderungen für die Web-basierte Anwendungsentwicklung zu ermitteln, ist sehr zu empfehlen. In dieser Hinsicht stellt die Einführung der zwei Sichten: *Entwicklersicht* und *Benutzersicht*, die die Frage stellen: „**Was wollen wir von einer Web-basierten Anwendung?**“ eine große Hilfe dar.

Aus der Entwicklersicht (s. Tabelle 1) haben die technischen, softwaretechnischen und ergonomischen Anforderungen und Qualitätsmerkmale einen großen Stellenwert. Sie sind jedoch keineswegs alleine zu berücksichtigen, sondern im Rahmen der Entwicklung Web-basierter Anwendungen nur gemeinsam mit den Benutzeranforderungen zu sehen.

	ENTWICKLERSICHT
ANFORDERUNGEN	Anwendung soll plattformunabhängig und einsetzbar in heterogenen Umgebungen sein
	Anwendung soll portabel sein: „Write Once, Run Anywhere“
	Anwendung soll über das Netz schnell zugreifbar sein (gute Antwortzeiten)
	Änderungen der Anwendung sind für Benutzer sofort verfügbar
	Anwendung soll Sicherheit des Benutzers nicht beeinträchtigen
	Anwendung soll Komponenten-basiert aufgebaut sein
	Anwendung soll klare Schnittstellen für andere Anwendungen anbieten
	Anwendung soll skalierbar und robust sein
	Anwendung soll integrierbar in anderen Systemlösungen sein
	Anwendung soll wiederverwendbar sein
	Anwendung darf nicht in inkonsistenten Zustand gebracht werden
	Anwendung soll benutzerfreundlich sein
	Anwendung soll auf externe Programme/Datenquellen zugreifen können

Tabelle 1 - Entwicklersicht (Anlehnung an [OTTSCH00])

Aus Tabelle 2 wird ersichtlich, dass der Benutzer klare Vorstellungen von einer Web-basierten Anwendung hat. Seine Vorstellungen sollen durch die aktive Partizipation in dem Entwicklungsprozess wahrgenommen werden.

ANFORDERUNGEN	BENUTZERSICHT
	von jeder Plattform auf die Anwendung zugreifen können
	24h Zugriff auf die Anwendung haben
	eine einfache Benutzeroberfläche
	Gewährleistung der Sicherheit während und nach der geleisteten Arbeit
	schnelle Antwortzeiten
	sich nicht um die Aktualisierung der Anwendung kümmern müssen

Tabelle 2 - Benutzersicht (Anlehnung an [OTTSCH00])

Die Anforderungen aus der Tabelle 1 und Tabelle 2 bilden eine gemeinsame Grundlage für die Ermittlung der Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen. Jedoch stellen diese Anforderungen nur eine kleine Untermenge aller Möglichkeiten dar.

Dies alles erschwert immer weiter die Ausarbeitung und Ermittlung verlässlicher Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen. Die Enterprise JavaBeans (EJB) (s. Kap. 2.2) erheben den Anspruch geeignete Mechanismen und Konstrukte für die Entwicklung Web-basierter Anwendungen bei der Umsetzung einiger softwaretechnischer Entwurfsprinzipien und Qualitätskriterien (s. Tabelle 3) bereitzustellen.

Qualitätskriterien, die eine Web-basierte Anwendung aufweisen sollte	**Entwurfsprinzipien, die in der Entwicklung einer Web-basierten Anwendung berücksichtigt werden sollten**
- Skalierbarkeit - Robustheit - Portabilität - Sicherheit - Wiederverwendung - Integration	- Kapselung - Trennung von Zuständigkeiten - Minimierte Kopplung - Verwendung von Architektur- und Entwurfsmuster

Tabelle 3 - Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen

Somit kann zumindest die Frage nach der Ermittlung verlässlicher Anforderungen und Kriterien der Entwicklung Web-basierter Anwendungen durch den Einsatz von EJB auf bestimmte softwaretechnische Entwurfsprinzipien und Qualitätskriterien eingeschränkt werden. Diese werden im Kapitel 3.3 am Fallbeispiel eines eGovernment-Bürgerprozessportals, das als Grundlage für die Entwicklung einer Web-basierten Anwendung dient, näher untersucht.

2.2 EJB-Konzepte

In diesem Abschnitt wird detailliert die Sun Microsystems Enterprise JavaBeans (EJB) Technologie zur Umsetzung von server-seitigen unternehmenskritischen Anwendungen erläutert. Als erstes werden im Kapitel 2.2.1 die Merkmale der EJB-Architektur vorgestellt. Im Kapitel 2.2.2 werden die Bestandteile der EJB-Architektur dargestellt und anschließend wird im Kapitel 2.2.3 auf die Rollenverteilung der Beteiligten während eines Entwicklungsprozesses eingegangen. Die drei EJB Bean-Typen (Session, Entity, Message-Driven Beans) werden im Kapitel 2.2.4 näher besprochen. Im Kapitel 2.2.5 und 2.2.6 werden das Transaktions- und Persistenzkonzept erläutert. Als eine Neuerung der EJB-Spezifikation 2.0 wird im Kapitel 2.2.7 die EJB QL Abfragesprache beschrieben. Am Ende werden im Kapitel 2.2.8 einige Einschränkungen der EJB-Spezifikation aufgeführt.

2.2.1 Merkmale der EJB-Architektur

Enterprise JavaBeans (EJB) sind ein Teil der Java 2 Enterprise Edition Plattform[11] (J2EE) von Sun Microsystems. Diese stellen eine Erweiterung der Java 2 Standard Edition (J2SE) um diverse Technologien und Schnittstellen zu bestimmten Diensten dar, die für die Entwicklung von Unternehmensanwendungen benötigt werden. J2EE umfasst neben Enterprise JavaBeans u.a. folgende weitere Technologien: Java Database Connectivity (JDBC), Java Servlets, Java Server Pages (JSP), Java Messaging Service (JMS), Remote Method Invocation (RMI), Interface Definition Language (IDL), Java Naming Directory Interface (JNDI), Java Transaction API (JTA) und Java Transaction Service (JTS).

Sun Microsystems definiert Enterprise JavaBeans 2.0 (EJB) folgendermaßen:

> *The Enterprise JavaBeans (EJB) architecture is a component architecture for the development and deployment of component-based distributed business applications. Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and multi-user secure. These applications may be written once, and then deployed on any server platform that supports the Enterprise JavaBeans specification.* [SUN01]

Die EJB-Architektur von Sun Microsystems stellt eine Spezifikation dar. Sie ist Javas neues **server-seitiges** Komponentenmodell, das auf einer Komponenten-basierten Architektur für verteilte unternehmerische Anwendungen konzipiert ist. Sie spezifiziert Schnittstellen, Aufgaben und Zuständigkeiten einzelner Komponenten. Die EJB-Architektur unterstützt durch ein breites Dienstleistungsspektrum die Realisierung von skalierbaren, robusten und mehrbenutzerfähigen server-seitigen Anwendungssystemen. Da die EJB-Spezifikation kein Produkt an sich ist, muss die EJB-Spezifikation von den Server- und Application-Serverherstellern umgesetzt werden.

[11] http://java.sun.com/j2ee/overview.html

EJB eignen sich besonders als server-seitiges Komponentenmodell für verteilte Multi-Tier-Umgebungen bzw. -Architektur. Web-Anwendungen weisen häufig eine Multi-Tier-Architektur auf, wobei die Middle-Tier für die Anwendungslogik verantwortlich ist (s. Abbildung 2). EJB stellen ein Komponentenmodell für die Realisierung der Anwendungslogik in Middle-Tier zur Verfügung. Sie kapseln die Anwendungslogik in die sogenannte „business logic“. EJB bieten zudem Unterstützung für Basisdienste wie Transaktions-, Sicherheits-, Persistenz-, Ressourcenmanagement, etc. Entwickler von Web-Anwendungen werden dadurch entlastet und können sich auf die Implementation der eigentlichen Anwendungslogik konzentrieren.

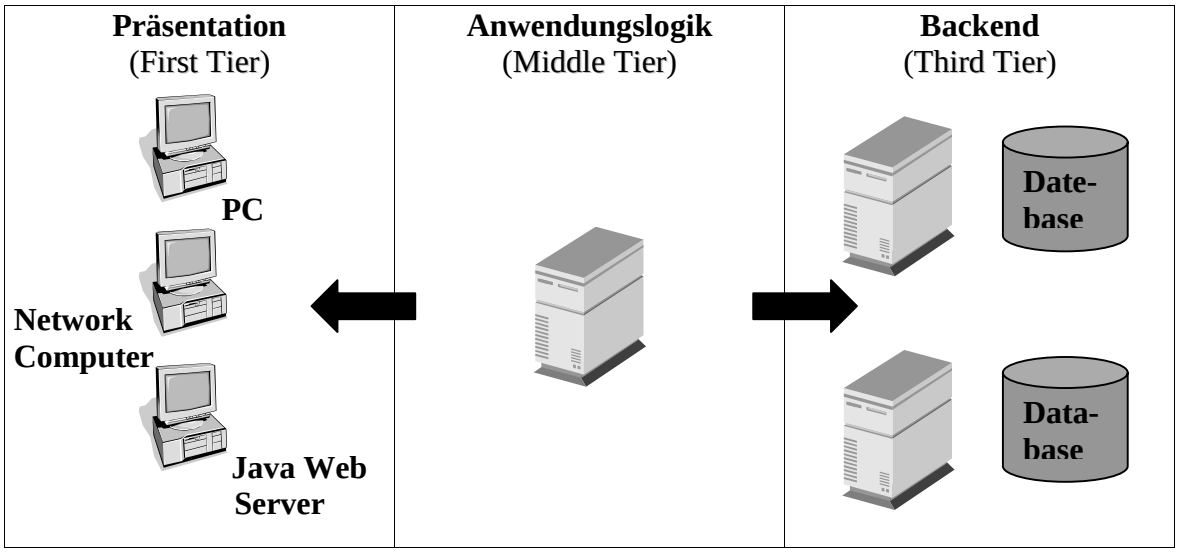

Abbildung 2 - Drei-Schichten Architektur (nach [MONSON01])

2.2.2 Architektur Überblick

Die Enterprise JavaBeans (EJB) Architektur ermöglicht komponentenbasierte Entwicklung von unternehmenskritischen Anwendungen. Durch die komponentenbasierte Architektur der EJB wird eine schnelle Erstellung von skalierbaren, robusten und mehrbenutzerfähigen unternehmerischen Anwendungen ermöglicht. EJB stellen eine kostenlose komplexe server-seitige Middleware für Anwendungsentwickler bereit. Der Einsatz von EJB eignet sich besonders gut in verteilten Umgebungen, wo die Geschäftslogik und -daten server-seitig liegen. Das EJB-Komponentenmodell unterstützt das Modellieren von Geschäftsprozessen und -daten der Anwendungslogik mit Hilfe von drei EJB Typen (siehe Kap. 2.2.4). Als Teil der J2EE-Architektur können EJB-Anwendungen in unterschiedlichen J2EE Umgebungen eingesetzt werden.

Im Folgenden werden die wesentlichen Bestandteile der EJB-Spezifikation bzw. EJB-Architektur und Laufzeitumgebung vorgestellt und detailliert erläutert. Dadurch wird ein Überblick über die einzelnen Bestandteile dieser Architektur gegeben und die Eignung und Praxistauglichkeit von EJB für die Entwicklung von Web-basierten Anwendungen kann somit besser erörtert werden. Die Abbildung 3 gibt eine einfache Übersicht über die EJB-Architektur.

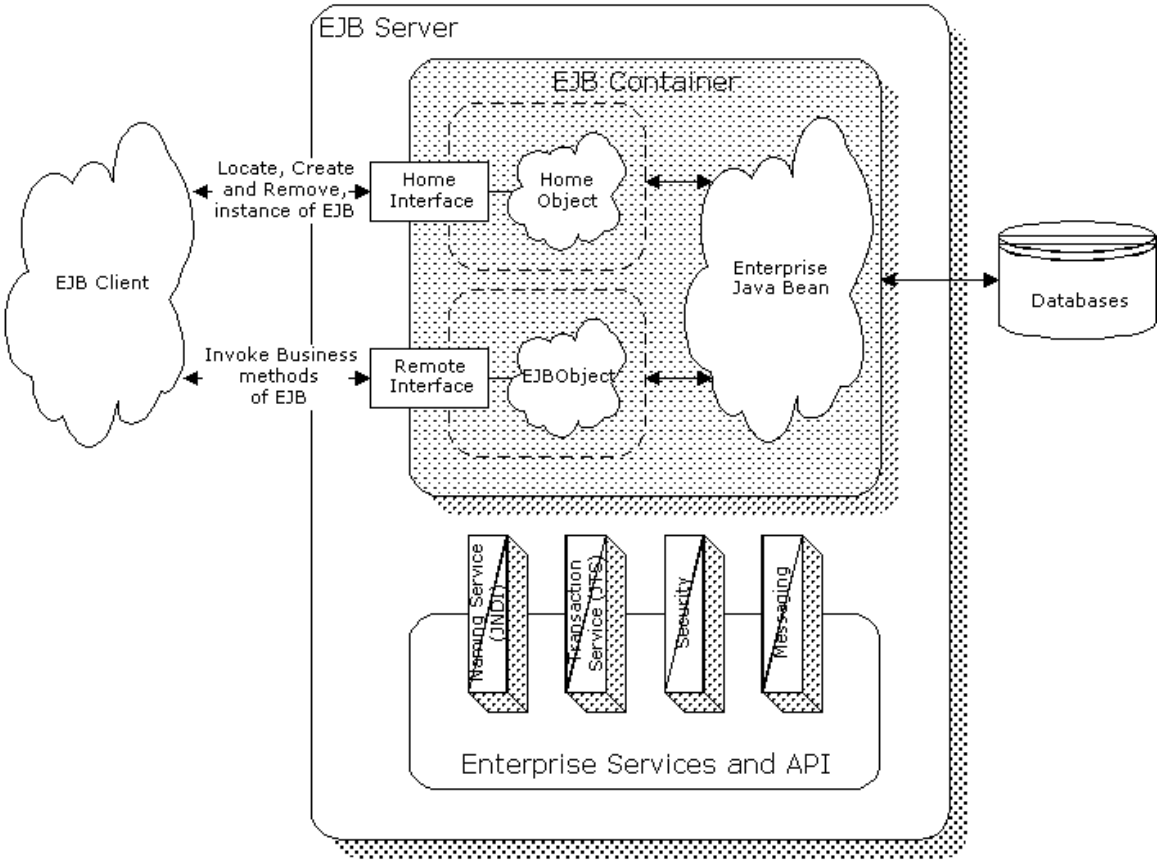

Abbildung 3 - Einfache Enterprise JavaBean Architektur[12]

2.2.2.1 EJB-Server

Der J2EE-konforme EJB-Server stellt die Umgebung für mindestens einen EJB-Container zur Verfügung. Die Grundfunktionalität des EJB-Servers ist die Kapselung der betriebssystemnahen und -spezifischen Funktionen, die dem EJB-Container über geeignete Schnittstellen bereitgestellt wird. Weitere Aufgaben des EJB-Servers sind z.B. die Garantie von Ausfallsicherheit, ggf. die Lastenverteilung zwischen mehreren Servern, das Bereitstellen eines Namens- und Verzeichnisdienstes, das Pooling von Ressourcen und das Thread- bzw. Prozessmanagement für den Einsatz mehrerer EJB-Container.

Die Spezifikation von SUN Microsystems legt die Schnittstelle zwischen dem EJB-Server und EJB-Container nicht fest, sondern überlässt dies den EJB-Serverherstellern. Daraus ergibt sich in der Praxis, dass EJB-Server und EJB-Container meistens von dem selben Hersteller kommen. Zudem kombinieren die meisten Hersteller Dienste und Anwendungen (wie z.B. Component Transaction Monitors, Web-Server, etc.) und bieten diese zusammen als Application-Server an.

[12] Autor: Gopalan Suresh Raj; http://www.idevresource.com/java/library/articles/ejbmodel.asp

2.2.2.2 EJB-Container

Der EJB-Container stellt die Infrastruktur bereit und ist für die Laufzeitumgebung einer EJBean verantwortlich. Die aufgelisteten Dienstanforderungen eines EJB-Containers sind von jedem EJB-Container Provider zu erfüllen. Jedoch können die Funktionalitäten, die nicht in der EJB-Spezifikation definiert sind, erweitert werden. Folgende Dienste werden vom EJB-Container zur Laufzeit für EJBeans zur Verfügung gestellt:

- Java 2 Platform, Standard Edition v1.3 (J2SE) API
- EJB 2.0 API (Enterprise JavaBeans)
- JNDI 1.2 (Java Naming and Directory Interface)
- JTA 1.0.1 (Java Transaction API)
- JDBC 2.0 (Java Database Connectivity)
- JMS 1.0.2 (Java Messaging Service)
- JavaMail 1.1
- JAXP 1.0 (Java API for XML Processing)

Die Abbildung 4 gibt die EJB-Client-Sicht auf Schnittstellen der EJBeans, auf die der EJB-Client zugreifen kann, wieder. Die EJBeans laufen allesamt in einem oder mehreren EJB-Container.

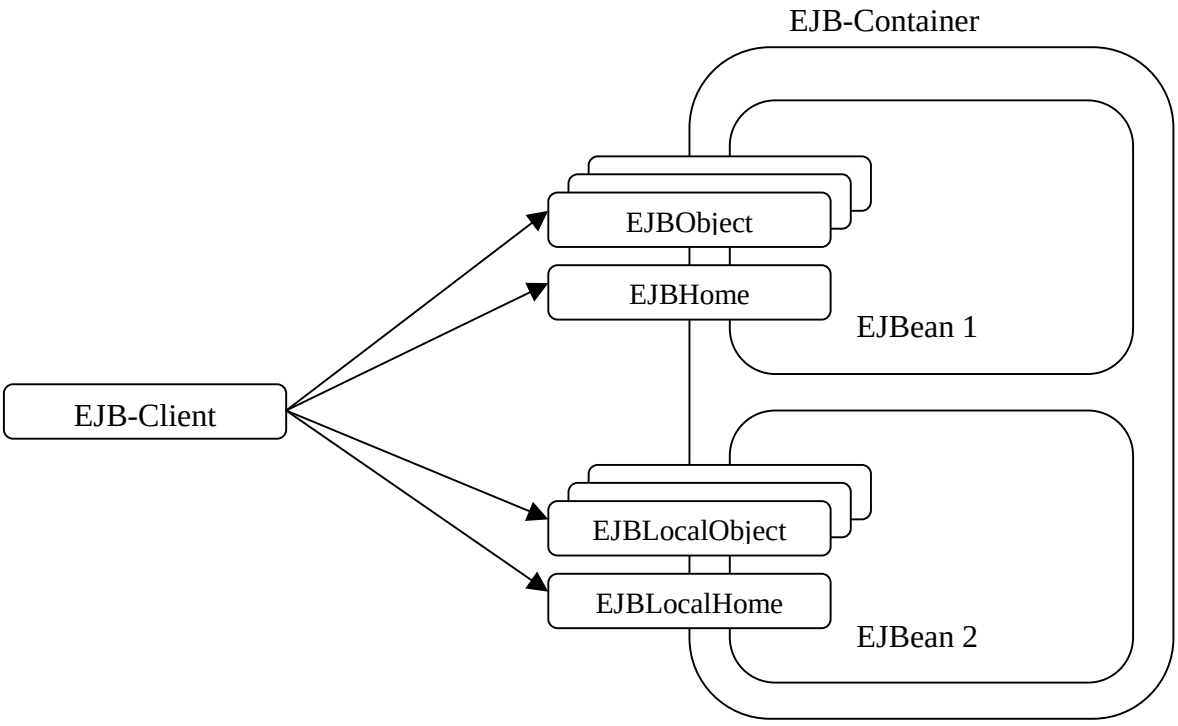

Abbildung 4 - Client View of EJBeans deployed in a EJB-Container [SUN01]

Neben diesen Diensten hat der EJB-Container noch eine ganze Reihe von Aufgaben. Im Folgenden werden diese Aufgaben kurz erläutert.

Ressourcen- und Lebenszyklusmanagement: Eine der Aufgaben des EJB-Containers ist Ressourcen-Management, wie z.B. Arbeitsspeicher, Threads, Socket-, Datenbankenverbindungen, etc. Die technische Implementierung dieser Aufgaben ist nicht in der EJB-Spezifikation definiert, sondern wird von den EJB-Container-Herstellern unterschiedlich realisiert. Da jeder EJB-Client nur über EJB-Container auf EJBeans zugreifen kann und zu jeder Zeit die Performance des EJB-Systems stabil bleiben soll, unterstützt EJB hierfür folgende zwei Mechanismen: *Instanzenpooling* und *Aktivierung*.

Bei dem Instanzenpooling hält der EJB-Container zur Laufzeit einige EJBean-Instanzen bereit, so dass sie immer wieder von EJB-Clienten verwendet werden können. Bei der Zuordnung einer EJBean-Instanz aus dem Pool zu einem EJB-Clienten wird eine Zuweisung mit den individuellen Daten initialisiert, so dass die Identität gewährleistet wird. Mit dieser Maßnahme wird das teure Erzeugen und Löschen von Instanzen minimiert, was dementsprechend die Ressourcen erheblich schont.

Im Lebenszyklus-Management einer EJBean wird das sogenannte Aktivieren und Passivieren der EJBean angewandt. Wird die EJBean für längere Zeit nicht genutzt und die Ressourcen wie Arbeitsspeicher sind knapp, kann der EJB-Container sie in einem permanenten Speicher passivieren. Dies geschieht, indem der EJB-Container sie serialisiert abspeichert (konserviert). Beim Aktivieren wird eine EJB-Instanz mit gespeicherten Zustand wieder assoziiert. Wie die genaue Ressourcen-Verwaltung und die Mechanismen für Passivieren und Aktivieren der EJBeans erfolgen, unterliegt der EJB-Container-Hersteller-Implementation.

Zugriffs- und Sicherheitsmanagement: Wie bei anderen verteilten Diensten in einer verteilten Umgebung nutzt EJB auch Mechanismen zum Auffinden von verteilten Objekten und Ressourcen. Dies geschieht durch die Verwendung des Java Naming and Diretory Interface (JNDI). Diese Schnittstelle wird vom EJB-Container bereitgestellt und ist für den Zugriff von einem EJB-Client auf eine EJBean zuständig. So kann z.B. wegen der Lastenverteilung die Verschiebung einer EJBean in einen anderen EJB-Container durch eine entsprechende Eintragsänderung im Namens- und Verzeichnisdienst erfolgen. Für den EJB-Clienten bleibt der Zugriff auf die EJBean weiterhin transparent.

Die EJB-Spezifikation unterstützt Sicherheitsmaßnahmen hinsichtlich der Authentifizierung (authentication), Zugriffskontrolle (access control) bis hin zur sicheren Kommunikation (secure communication). Der EJB-Container selbst adressiert nur die Authentifizierung und die Zugriffskontrolle. Durch die Eintragung in dem Deployment descriptor kann festgelegt werden, ob ein EJB-Client auf bestimmte Methoden einer EJBean zugreifen kann. Zudem können Rollen und Zugriffsrechte für jeden Benutzer zugeordnet und definiert werden. Die Rollen- und Zugriffsangaben werden ebenfalls im Deployment descriptor festgelegt.

Glue-code und EJBean Installationstools: Jeder EJB-Container stellt den Glue-code und Installationstools zur Verfügung. Diese Tools werden für die EJBean-Integration in einer EJB-Container-Umgebung bzw. EJB-Server eingesetzt. Der Glue-code und die Installationstools werden u.a. auch für die Generierung der in der Laufzeit benötigten EJBean Klassen oder Hilfs-Klassen, wie Stubs, Skeletons, Data Access Klassen und andere vom EJB-Container spezifisch erforderliche Klassen, verwendet. Sie unterstützen somit den Einsatz der EJBeans in einem EJB-Container.

Transaktionsmanagement: Eine weitere wichtige Aufgabe des EJB-Containers ist das Transaktionsmanagement. Der EJB-Container stellt implizite und explizite Dienste für die konsistente Datenerhaltung, Nebenläufigkeitskontrolle sowie Verwaltung und Überwachung von Client-Container-Aktionen bereit. Mehr über dieses Thema findet sich in Kapitel 2.2.5.

Persistenzmanagement: Für die persistente Datenerhaltung einer Anwendung stellt der EJB-Container implizite und explizite Dienste zur Verfügung. Für den Fall eines impliziten Persistenzmanagements (Container Managed Persistence) übernimmt der EJB-Container die automatische Speicherung des EJBean-Zustands in einem permanenten Medium, wie z.B. einer relationalen Datenbank. Mehr über dieses Thema findet sich in Kapitel 2.2.6.

2.2.2.3 Enterprise Java Bean

Eine Enterprise Java Bean (EJBean) setzt sich aus mehreren Bestandteilen zusammen. Diese unterstützen die Modellierung und Realisierung der Geschäfts-, Daten- und Benachrichtigunglogik einer Anwendungslogik nach der EJB-Vorgehensweise. All diese Bestandteile werden in einem *ejb.jar* Archiv zusammengefasst und in dem EJB-Container installiert. Das Archiv besteht aus mindestens einem Remote Interface, Remote home Interface, und/oder Local Interface, Local home Interface, Bean class, Primary key (nur Entity Beans) und Deployment descriptor.

Remote Interface: Das Remote Interface definiert die Business-Methoden einer EJBean, die von einer externen Anwendung oder einem Remote EJB-Client (siehe Kap. 2.2.2.4) aufgerufen werden können. Diese Methoden sind somit für die Außenwelt (außerhalb des EJB-Containers) sichtbar. Jedes Remote Interface muss das Interface `javax.ejb.EJBObject` erweitern, welches wiederum das Interface von `java.rmi.Remote` ableitet und dadurch das Interface eines RMI-Remote-Objektes darstellt. Das Remote Interface wird in Session und Entity Beans in Verbindung mit dem Remote home interface verwendet.

Remote home Interface: Das Remote home Interface definiert alle Lebenszyklus-Methoden einer EJBean. Diese Lebenszyklus-Methoden werden durch externe Anwendungen oder den Remote EJB-Client zum Erzeugen, Löschen und Finden einer EJBean aufgerufen. Jedes Remote home Interface muss das Interface `javax.ejb.EJBHome` erweitern, das wiederum das Interface von `java.rmi.Remote` ableitet und somit ein Interface eines RMI-Remote-Objektes darstellt. Das Remote home

Interface wird in Session und Entity Beans in Verbindung mit dem Remote interface verwendet.

Local Interface: Das Local Interface definiert für interne EJB-Clients (siehe Kap. 2.2.2.4) die Business-Methoden einer EJBean, die von anderen in der selben Java Virtual Machine befindlichen EJBeans (interne EJB-Clients) aufgerufen werden können. Die Verwendung des umfangreichen und performanzkostspieligen RMI-IIOP[13]-Protokolls wird somit umgangen. Dadurch wird eine erhebliche Performanzsteigerung der EJBean bzw. der EJB-Anwendung erreicht. Das Local Interface erweitert `javax.ejb.EJBLocalObject`. Es wird in Session und Entity Beans in Verbindung mit dem Local home Interface verwendet.

Local home Interface: Das Local home Interface definiert für interne EJB-Clients alle Methoden, betreffend den Lebenszyklus einer EJBean, die von anderen in der selben Java Virtual Machine befindlichen EJBeans (interne EJB-Cliente) aufgerufen werden können. Somit stehen die Lebenszyklus-Methoden für andere EJBeans zum Erzeugen, Löschen und Finden einer EJBean zur Verfügung. Auch hier wird die Benutzung des kostspieligen RMI-IIOP-Protokolls umgangen. Das Local home Interface erweitert `javax.ejb.EJBLocalHome`. Das Local home Interface wird in Session und Entity Beans in Verbindung mit dem Local Interface verwendet.

EJBean class: In der EJBean class werden alle Methoden, die im Remote, Remote home, Local, Local home Interface definiert wurden, implementiert. Abhängig vom EJBean-Typ und dessen Ausprägung werden jedoch ggf. manche Methoden leer oder nicht implementiert, da der EJB-Container automatisch eine entsprechende Funktionalität (meistens Callback-Methoden) bereitstellt. Dennoch müssen alle Methoden die gleichen Signaturen wie im Remote und Remote home oder Local und Local home Interface haben. Einige Callback-Methoden können in der EJBean class überschrieben werden. Die Entity Bean Klasse implementiert das Interface `javax.ejb.EntityBean` und die Session Bean Klasse `javax.ejb.SessionBean`. Beide Interfaces sind von dem Interface `javax.ejb.EnterpriseBean` und dieses wiederum von `java.io.Serializable` abgeleitet.

Die Message-Driven Beans (siehe Kap. 2.2.4.3) verwenden keine der oben aufgeführten Interfaces (Remote, Remote home oder Local, Local home), da sie weder von einem internen oder externen EJB-Client noch von einer EJBean aufgerufen werden können. Eine Message-Driven Bean wird vom EJB-Container nach Ankommen einer Nachricht aufgerufen. Die Message-Driven Bean Klasse implementiert das Interface `javax.ejb.MessageDrivenBean` und `javax.jms.MessageListener`.

[13] Java ™ Remote Method Invocation ("Java RMI") technology run over Internet Inter-Orb Protocol ("RMI-IIOP") delivers Common Object Request Broker Architecture (CORBA) distributed computing capabilities to the Java™ 2 platform. RMI over IIOP provides the ability to write CORBA applications for the Java platform without learning CORBA Interface Definition Language (IDL)
http://java.sun.com/products/rmi-iiop/index.html

Primary key: Die Primary key Klasse wird nur für Entity Beans benötigt. Ein Objekt dieser Klasse zeigt auf einen Rekord der Datenbank bzw. eine Entity Bean in der Datenbank. So lässt sich eine Entity Bean leicht finden und identifizieren. Der Primary key kann entweder *single-field* oder *compound* Typen haben. Diese müssen jedoch serialisierbare Typen sein; *single-field* z.B. primitive wrapper Typen `Integer`, `Double`, `Long`, etc. oder *compound*, eigene Klasse für komplexere Typen. Beide implementieren das Interface `java.io.Serializable`.

Deployment descriptor: Der Deployment descriptor[14] ist eine umfassende XML-Datei, die Verwaltungs- und Laufzeitinformationen (die weder in Interfaces oder Klassen einer EJBean adressiert werden) einer EJB-Anwendung bzw. eines *ejb.jar* Archivs enthält. Er tangiert als Attributebeschreibungsdatei einer EJB-Anwendung, ähnlich einer property file oder property sheet. Die Verwaltungs- und Laufzeitinformationen umfassen u.a. die Zugriffsrechte der einzelnen Methoden einer EJBean, die Persistenzart (siehe Kap. 2.2.6) einer EJBean, Anzahl der instantiierten EJBeans, etc. Diese Informationen werden im EJB-Container mit Hilfe der Installationstools eingesetzt.

Durch Veränderung der Attribute im Deployment descriptor ist es möglich während der Laufzeit einer EJB-Anwendung eine Anpassung der EJB-Anwendung bzw. EJBeans ohne eine neue Kompilierung vorzunehmen. Die genau Handhabung sowie die Aufgaben des Deployment descriptors werden von Sun Microsystems unter http://java.sun.com/dtd/ejb-jar_2_0.dtd näher beschrieben.

2.2.2.4 EJB-Client

In einer EJB-Anwendung wird zwischen einem Remote (externen) oder einem Local (internen) EJB-Client, der sich außerhalb oder innerhalb einer Java Virtual Machine[15] befindet, unterschieden. Ein externer EJB-Client kann ein Java Applet oder Servlet oder eine externe Java-Applikation sein. Ein interner EJB-Client kann eine Entity, Session oder Message-Driven Bean sein. Nur über die Entity und Session Bean Interfaces bzw. die darin definierten Methoden können externe und interne EJB-Clients auf die EJBean class zugreifen. Die EJBean class selbst ist für den externen oder internen EJB-Client nicht sichtbar. Die Unterscheidung zwischen einem externen und internen EJB-Client wurde erst in der EJB-Spezifikation 2.0 eingeführt. Der wesentliche Grund hierfür war die Vermeidung der Benutzung des performanzkostspieligen RMI-IIOP-Protokolls für interne Komponenten einer EJB-Anwendung.

Wesentliche Unterschiede zwischen Remote und Local EJB-Clients sind:

- Remote EJB-Client verwendet das Java Remote Method Interface (RMI)
- Remote-Aufrufe sind pass-by-value

[14] The deployment descriptor includes the structural information (e.g. the name of the enterprise bean class) of the enterprise bean and declares all the enterprise bean's external dependencies (e.g. the names and types of resources that the enterprise bean uses) [SUN01].

[15] In [MONSON01] wird zwischen einem EJB-Client, der sich außerhalb oder innerhalb eines EJB-Containers befindet, unterschieden.

- Remote-Objekte, die als Parameter übergeben werden, müssen serialisierbar sein.
- Remote EJB-Client ist Standort unabhängig
- Local EJB-Client kommuniziert lokal
- Local-Aufrufe sind pass-by-reference
- Local-Objekte arbeiten in der selben Java Virtual Machine (JVM)
- Local EJB-Client ist Standort abhängig

Das Lokalisieren von EJBeans durch die EJB-Clients wird mit Hilfe des Java Naming and Directory Interface[16] (JNDI) realisiert. Die vom EJB-Container angebotene JNDI-Dienstleistung ermöglicht einem EJB-Client den EJB-Server als eine Menge von Direktorien, wie auf einem Dateisystem, zu durchsuchen. Der EJB-Client hat dabei aber zu keinem Zeitpunkt die Möglichkeit auf die interne Verwaltung der EJBeans im EJB-Server Einfluss zu nehmen.

2.2.3 Rollenverteilung

Für die Entwicklung server-seitiger Anwendungen definieren Enterprise JavaBeans für die Beteiligten in dem Entwicklungsprozess sechs verschiedene Rollen. Diese Rollen unterstützen die Teilung der Aufgaben und Zuständigkeiten in einem Softwareprojekt. Die EJB-Architektur spezifiziert Verträge (contracts) an denen jede EJB-Rolle mit ihrer Komponente kompatibel zu der Komponente der anderen EJB-Rolle bleibt. Dies ermöglicht die gemeinsame Arbeit von einer Vielzahl von Personen oder Gruppen, die alle unterschiedliche Aufgaben und Spezialgebiete haben.

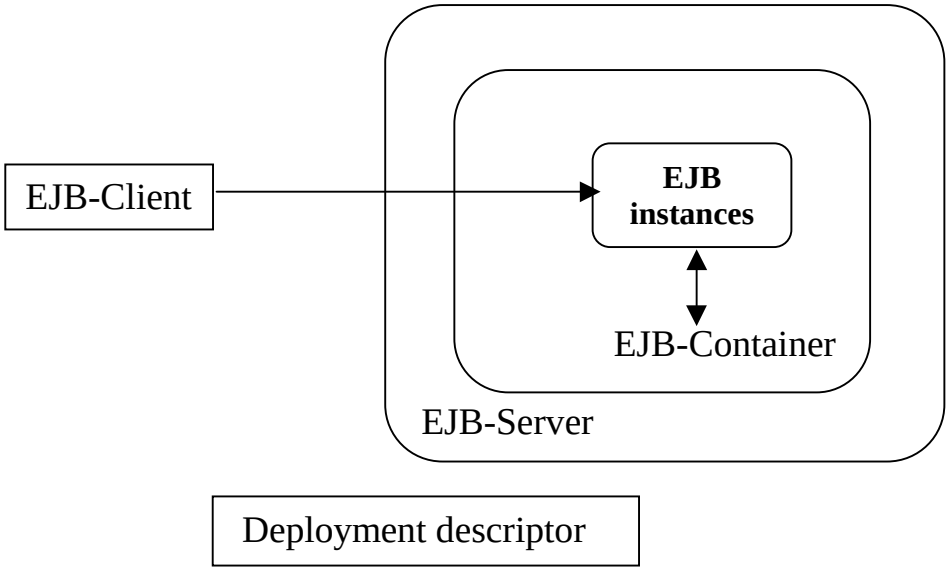

Abbildung 5 - Enterprise JavaBeans Component Contracts nach [SUN01]

[16] The Java Naming and Directory Interface™ (JNDI) is a standard extension to the Java™ platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. (http://java.sun.com/products/jndi/index.html)

Folgende Rollen sind in der EJB-Spezifikation definiert: Enterprise Bean Provider, Application Assembler, Deployer, EJB-Server Provider, EJB-Container Provider und Systemadministrator. Die Abbildung 5 gibt einen groben Überblick über die Rollenverteilung in einer EJB-Umgebung.

Enterprise Bean Provider: Der Enterprise Bean Provider implementiert die Anwendungslogik und fasst alle Komponenten der Enterprise JavaBeans in *ejb.jar* zusammen. Das beinhaltet die Bean Klasse, deren Schnittstellen Remote und Remote home und/oder Local und Local home sowie den Deployment descriptor. Der Enterprise Bean Provider ist der Entwickler der wiederverwendbaren Komponenten innerhalb der Anwendungslogik. Diese sollen unabhängig von dem Anwendungskontext in jeder J2EE einsetzbar sein. Von dem Entwickler wird nicht vorausgesetzt, dass er sich mit der system-level Programmierung auskennt. Dies umfasst normalerweise auch nicht die Programmierung von system-level Transaktionen, Nebenläufigkeiten, Sicherheitsaspekten oder anderen ähnlichen Diensten. Der Enterprise Bean Provider verlässt sich diesbezüglich auf den EJB-Container Provider.

Application Assembler: Der Application Assembler stellt eine oder mehrere EJB des Enterprise Bean Providers zu einer Anwendung bzw. zu einer oder mehreren *ejb.jar* zusammen und fügt ggf. weitere Anwendungsbestandteile (wie z.B. Servlets, JSP, etc.) hinzu. Der Application Assembler erweitert den Deployment descriptor entsprechend um weitere Anweisungen und stellt sie für den Einsatz zur Verfügung. Letztlich muss der Application Assembler sich nicht mit der Implementation der EJB auskennen, dennoch sollte er die Funktionalität der EJBean sowie deren Schnittstellen verstehen können.

Deployer: Der Deployer ist für die Installation der von dem Application Assembler erstellten EJB-Anwendung in dem spezifischen operationalen EJB-Container zuständig. Er hat fundiertes Wissen über den EJB-Container Provider und das EJB-System, in dem die Anwendung laufen wird. Zu seinen Aufgaben gehört die Abhängigkeiten, die im Deployment descriptor beschrieben sind, aufzulösen, um so die für den Betrieb der EJB-Anwendung nötigen Rahmenbedingungen zu schaffen. In manchen Fällen ist der Deployer auch dazu qualifiziert die Anpassung der Anwendung durchzuführen. Um diese Aufgabe zu erfüllen, verwendet er die vom EJB Container Provider zur Verfügung gestellten spezifischen Tools.

EJB-Server Provider: Der EJB-Server Provider ist ein Experte auf dem Gebiet der verteilten Anwendungen, wie dem verteilten Transaktionsmanagement und der betriebssystemnahen low- und system-level Programmierung. Die EJB-Spezifikation legt dabei nahe, dass der EJB-Server Provider und der EJB-Container Provider von dem selben Hersteller kommen sollten.

EJB-Container Provider: Der EJB-Container Provider stellt Deployment Tools (notwendig für den Einsatz der EJB-Anwendung) sowie Runtime Support für die eingesetzte EJB-Anwendung zur Verfügung. Aus der Sicht der EJB-Spezifikation ist der EJB-Container Provider ein Teil der operationalen Umgebung. Er ist zuständig für das

Transaktions- und Sicherheitsmanagement, für das skalierbare Ressourcen-Management und weitere erforderliche Dienste.

Systemadministrator: Der Systemadministrator ist zuständig für die Administration und Konfiguration der Netzwerk- und Serverinfrastruktur des EJB-Servers und EJB-Containers. Zu seinen Aufgaben gehört auch die Sicherstellung der Verfügbarkeit der EJB-Anwendung während der gesamten Laufzeit.

2.2.4 Typen von EJB

Die EJB-Spezifikation unterscheidet drei Arten von EJBean Typen: Session, Entity und Message-Driven Beans. In der Anwendungslogik stellen die Session Beans grundsätzlich die Geschäftslogik dar, dagegen bilden die Entity Beans Geschäftsdaten ab. Message-Driven Beans werden zur asynchronen Benachrichtigung mit den Entity oder Session Beans sowie dem EJB-Client eingesetzt. Diese Unterteilung von EJB Typen ermöglicht eine klare Trennung von Aufgaben und Zuständigkeiten in der Anwendungs-, Daten- und Benachrichtigungslogik. In der Entwicklung der Web-basierten Anwendungen unterstützen sie die Realisierung einer schichtenbasierten Architektur.

2.2.4.1 Session Beans

Üblicherweise implementiert eine Session Bean die Geschäftslogik, Geschäftsrichtlinien oder den Workflow einer Anwendung in der Anwendungslogik. Sie sind in ihrem Lebenszyklus überwiegend kurzlebig, daher auch der Name Session Beans. Sie repräsentieren die Geschäftsdaten nicht, aber sie werden für das Lesen, Aktualisieren und Hinzufügen der gemeinsam benutzten Daten in dem unterliegenden Persistenzmedium eingesetzt. Meistens werden sie in der Umsetzung der Geschäftslogik als *Verben* modelliert.

Session Beans sind in der EJB-Spezifikation als zustandsbehaftet (stateful) gedacht worden. Einen Sonderfall bilden hier die zustandslosen (stateless) Session Beans. Der wesentliche Unterschied liegt darin, dass zustandsbehaftete Session Beans über mehrere Anfragen von demselben Klienten bzw. während des gesamten Lebenszyklus der Session-Instanz für diesen Klienten, in Form von Attributen und Methoden, ihren internen Zustand (auch bekannt als *conversational state*) speichern können. Zustandsbehaftete Session Beans sind nur einem Klienten zugeordnet und können dadurch ihren internen Zustand verändern und später auf diesen zugreifen. Sie können ihren internen Zustand jedoch nicht persistent auf ein unterliegendes Persistenzmedium speichern. Darüber hinaus müssen sie selbst dafür sorgen, dass sie sich im Falle einer Rollback-Aktion während einer Transaktion in ihren internen Zustand zurücksetzen.

Im Gegensatz dazu speichern die zustandslosen Session Beans ihren internen Zustand nicht. Jede zustandslose Session-Instanz kann von jedem beliebigen Klienten verwendet werden. Somit stellen sie anonyme und für alle Klienten vollkommen transparente Dienste bereit. Die in Session Beans realisierten Methoden bieten sozusagen eine Sammlung von verwandten Diensten. Dabei benötigen sie im Gegenteil zu den

zustandsbehafteten Session Beans weniger EJB-Server Ressourcen und sind relativ einfach zu realisieren.

Die Angaben über die Art von verwendeten Session Beans werden in dem Deployment descriptor festgelegt.

2.2.4.2 Entity Beans

Entity Beans bilden die Anwendungsdaten einer Anwendung auf einem persistenten Medium ab. Sie stellen die objektorientierte Sicht der modellierten Anwendungsdaten dar. Üblicherweise werden Entity Beans als *Substantive* modelliert. Im Gegensatz zu Session Beans bleiben die Daten der Entity Beans über mehrere EJB-Client-Sitzungen oder Neustarts des EJB-Servers durch das Abspeichern auf dem Persistenzmedium erhalten. Darüber hinaus können sie während ihres Lebenszyklus von mehreren EJB-Clients gleichzeitig benutzt werden. Für die Synchronisation der Nebenläufigkeit sorgt die Zugriffs- und Ressourcenmanagementkomponente des EJB-Containers.

Jede Entity Bean hat durch die Verwendung der *Primary key* Klasse (s. Kapitel 2.2.2.3) eine eindeutige Identifikation. Für die Manipulation der Entity Bean Identifikation werden über die Remote home und/oder Local home Schnittstelle lesende und suchende Methoden standardmäßig vom EJB-Container angeboten. Die Implementierung dieser Methoden hängt von der gewählten Art des Persistenzmechanismus der Entity Beans ab.

Entity Beans lassen sich bezüglich der Art, wie die Daten der Beans persistent gehalten werden, in Bean Managed Persistence (BMP) und Container Managed Persistence (CMP) unterteilen.

Bei der Bean Managed Persistence (BMP) muss der Entwickler der Entity Beans selbst die Persistenzmechanismuslogik realisieren. Zu seinen Aufgabe gehört die Entity Beans in einem permanenten Speicher anzulegen, zu verändern, zu löschen, zu finden, etc. Hierfür benötigt der Entwickler fundierte Kenntnisse des eingesetzten Persistenzmediums sowie Wissen über Persistenzmechanismus-Handling. Der EJB-Container unterstützt den Entwickler hierbei mit Mechanismen und Namenskonventionen für die Methodenaufrufe.

Im Falle einer Container Managed Persistence (CMP) übernimmt der EJB-Container die Aufgabe, die Persistenzmechanismuslogik zu realisieren. Der Entwickler der Entity Beans muss nur die persistenten Attribute (Datenfelder) und die Attribute-Beziehung definieren. Diese und weitere Angaben von Entity Beans werden im Deployment descriptor festgelegt. Mehr über das Thema Persistenz findet sich in Kapitel 2.2.6.

2.2.4.3 Message-Driven Beans

Message-Driven Beans sind den Session Beans ähnlich, da sie die Realisierung der Geschäftslogik, Geschäftsrichtlinien oder des Workflows einer Anwendung in der Anwendungslogik durch Benachrichtigungsmechanismen unterstützen. Der grundlegende Unterschied ist, dass Message-Driven Beans nicht von EJB-Clients oder anderen

EJBeans aufgerufen werden können, sondern nur durch das Ankommen von Nachrichten über den vom EJB-Container bereitgestellten Java Message Service (JMS) aktiviert werden können. Die Methode `onMessage()`der Message-Driven Bean Klasse wird nach Ankommen der Nachricht vom EJB-Container aufgerufen. Die EJB-Spezifikation definiert Message-Driven Beans als *„stateless, server-side, transaction-aware components for processing asynchronous Java Message Service (JMS) messages.“* Message-Driven Beans sind erst ab der Version 2.0 der Enterprise JavaBeans Teil der EJB-Spezifikation.

Der Java Message Service bietet eine einfachere und flexiblere Alternative für asynchronen Austausch von Nachrichten zur Remote Method Invocation (RMI) an. Er verbraucht weniger Ressourcen und ist weniger restriktiv. Der Java Message Service stellt zwei Nachrichtendienstmechanismen: publish-and-subscribe und point-to-point zur Verfügung. Beide Mechanismen können von Message-Driven Beans verwendet werden. Ein weiterer wichtiger Aspekt von Message-Driven Beans ist zudem, dass sie in der Lage sind gleichzeitig Nachrichten zu erzeugen und zu konsumieren.

Alle Angaben zu Message-Driven Beans sowie deren Eigenschaften werden, wie für Session und Entity Beans, in dem Deployment descriptor festgelegt.

2.2.5 Transaktionen

Ein Hauptmerkmal der EJB-Architektur ist die Transaktionsunterstützung. Die Bereitstellung der impliziten und expliziten Transaktionsunterstützung durch EJB setzt neue Maßstäbe in der Entwicklung von Web-Technologien bzw. in der Realisierung robuster Web-basierter Anwendungen. In der Entwicklung von Web-basierten Anwendungen übernimmt die Transaktionsunterstützung somit in der Praxis einige Implementieraufgaben, die bisher zum Teil jede einzeln realisiert werden mussten.

Transaktionen ermöglichen dem Entwickler die Implementierung der Anwendungslogik, die atomare Aktionen über mehrere Persistenzmedien in verteilten Umgebungen durchführt. Dies umschließt alle ACID Prinzipien (siehe [GRAREU93]): Unteilbarkeit, Konsistenz, Isolation und Dauerhaftigkeit einer Transaktion. Die EJB-Spezifikation setzt auf die Java Transaction API[17] (JTA), die von dem EJB-Container bereitgestellt wird. Zudem ist der EJB-Container und EJB-Server für die Implementierung der low-level Transaktionsprotokolle zuständig.

Das Tranksaktionsmodell der EJB-Spezifikation unterstützt nur flache, nicht geschachtelte, Transaktionen. Das bedeutet, dass immer nur innerhalb des Beginns und Endes (Bestätigung oder Abbruch) einer Transaktion beliebige Operationen ausgeführt werden können. Es gilt das „Alles-oder-nichts-Prinzip“. Es gibt keine Möglichkeit, Teile einer flachen Transaktion zu bestätigen oder abzubrechen. Alle Aktionen sind untrennbar

[17] Java Transaction API specifies standard Java™ interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications. http://java.sun.com/products/jta/index.html

miteinander verbunden und werden „unit-of-work“ genannt. Alle in einem „unit-of-work“ teilnehmenden Aktionen befinden sich im gleichen Transaktionskontext.

Weiterhin unterscheidet die EJB-Spezifikation zwischen implizitem: *Container Managed Transaction Demarcation* (CMTD) und explizitem: *Bean Managed Transaction Demarcation* (BMTD) Transaktionsmanagement. Bei dem impliziten Transaktionsmanagement übernimmt der EJB-Container die vollständige Verwaltung der Transaktionsunterstützung. Über die Transaktionsattribute im Deployment descriptor wird das Verhalten der Transaktionssteuerung von EJBeans oder dessen einzelnen Methoden bestimmt. Folgende Transaktionsattribute können gesetzt werden: *NotSupported, Supports, Required, RequiresNew, Mandatory, Never*.

- *NotSupported:* Dieses Attribut setzt das Ausführen einer Transaktion, bis die gekennzeichnete Methode ihre Ausführung abgeschlossen hat, aus. Sie kann während dieser Zeit nicht von anderen Methoden, die in einer Transaktion ausgeführt werden müssen (*Required*), benutzt werden.

- *Supports:* Das Ausführen der mit diesem Attribut gekennzeichneten Methode hängt von dem Transaktionskontext des EJB-Clients ab. Beim gesetzten Transaktionskontext verhält sich der EJB-Container wie beim *Required* Attribut, ansonsten wie beim *NotSupported* Attribut.

- *Required:* Alle mit diesem Attribut gekennzeichneten Methoden werden während einer Transaktion ausgeführt. Wird die Methode nicht in einem laufenden Transaktionskontext des EJB-Clients aufgerufen, generiert der EJB-Container für diesen Methodenaufruf eine neue Transaktion.

- *RequiresNew:* Die mit diesem Attribut gekennzeichnete Methode wird, ungeachtet vom Transaktionskontext des EJB-Clients, immer in einer neuen Transaktion gestartet. Findet der Methodenaufruf in einer laufenden Transaktion statt, so wird diese bis zur Beendigung des Methodenaufrufs ausgesetzt.

- *Mandatory:* Dieses Attribut bedeutet, dass der Methodenaufruf immer in einem Transaktionskontext des EJB-Clients stattfinden muss. Ist das nicht der Fall, so wird eine Fehlermeldung ausgegeben.

- *Never:* Für dieses gesetzte Attribut darf die gekennzeichnete Methode nicht in einem Transaktionskontext des EJB-Clients ausgeführt werden. Der EJB-Container reagiert hierbei mit einer entsprechenden Fehlermeldung.

Die Transaktionsattribute werden entweder von dem Bean-Provider oder Application Assembler im Deployment descriptor genau festgelegt.

Im Falle des expliziten Transaktionsmanagements muss der Entwickler die gesamte Transaktionsunterstützung selbst steuern. Somit lassen sich Transaktionskontexte in der Anwendungslogik von dem Entwickler selbst festlegen. Hierdurch kommt es zu einer

Vermischung zwischen Anwendungs- und Transaktionslogik. Jegliche Änderung in der Transaktionslogik führt zum neuen Kompilieren der Anwendungslogik. Daher sollte man grundsätzlich das implizite Transaktionsmanagement dem expliziten vorziehen.

Auch beim Eintreten eines Fehlers während der Transaktionssteuerung werden unterschiedliche Fehlerbehandlungsmaßnahmen für das implizite und explizite Transaktionsmanagement durchgeführt. Während bei dem impliziten Transaktionsmanagement grundsätzlich ein Rollback von dem EJB-Container veranlasst wird, erhält der EJB-Client beim expliziten Transaktionsmanagement eine Exception. Der EJB-Client entscheidet nach der Exception selbst, ob er ein Rollback oder eine andere Aktion ausführt.

Session Beans und Message-driven Beans unterstützen sowohl implizites als auch explizites Transaktionsmanagement. Dagegen sind Entity Beans nur durch implizites Transaktionsmanagement zu realisieren.

2.2.6 Persistenz

Persistente Haltung von Daten ist nach der EJB-Spezifikation nur für Entity Beans vorgesehen. Entity Beans kennen zwei Arten von Persistenzmechanismen implizite *Container Managed Persistence* (CMP) und explizite *Bean Managed Persistence* (BMP). In diesem Bereich hat EJB neue Standards in der Entwicklung von Web-Technologien gesetzt. EJB ermöglichen hier die implizite und explizite Persistenzunterstützung, die in der Web-basierten Anwendungsentwicklung eine große Hilfestellung ist. Dem Entwickler wurden dadurch Alternativwege zur Realisierung der Persistenzverwaltung geboten. Hierbei ist noch wichtig zu erwähnen, dass die Persistenzverwaltung in der EJB-Spezifikation 2.0 vollständig überarbeitet wurde und nicht kompatibel zu der EJB-Spezifikation 1.1 und 1.0 ist.

Container Managed Persistenz: Der EJB-Container übernimmt die automatische Persistenzverwaltung der in Entity Beans festgelegten Attribute. Er stellt Dienste für die Abwicklung von Transaktionen und abschließende persistente Speicherung von Attributen (Daten) zur Verfügung. Die Implementierung der durch Entity Beans gekennzeichneten Attribute geschieht durch die EJB-Container-Tools. Die Aufgabe des Entwicklers besteht darin, die Entity Beans Attribute und das Relationenmodell in dem sogenannten *Abstract Persistence Schema*[18] zu definieren. Das abstrakte Persistenzschema ist Teil des Deployment descriptors. Wird als Persistenzmedium eine relationale Datenbank eingesetzt, so werden Attribute als Spalte einer Tabelle abgebildet. Außerdem können Entity Beans Attribute mit anderen Entity Beans Attributen in Relation gestellt werden. Folgende Relationsmodelle werden unterstützt: 1:1 Relation, 1:n Relation und n:m Relation. Diese Relationsmodelle können uni- und bidirektional sein.

[18] "The abstract persistence schema is the set of XML elements in the deployment descriptor that describes the persistence fields and the relationship fields." [MONSON01]

Bei der Container Managed Persistence muss die Entity Bean eine abstrakte Klasse sein. Die konkrete Klasse wird von dem EJB-Container zur Laufzeit generiert. Persistente Attribute (CMP) und Relationsattribute (CMR) dürfen nicht in der Entity Bean Klasse implementiert werden. Nur deren Signaturen werden dort festgelegt; sie werden virtuelle Attribute genannt. Die tatsächliche Festlegung von Persistenz- und Relationsattributen wird im Abstract Persistence Schema gemacht. Alle virtuellen Attribute sind standardmäßig persistent. Diese Attribute können Java serialisierbare Typen, die das Interface `java.io.Serializable` implementieren, oder Java primitive Typen haben. Man kann auch eigene serialisierbare Typen definieren. Diese werden *dependent value classes* genannt.

Der Entwickler muss nur die abstrakten get- und set-Methoden (*accessor methods*) von Attributen in der Entity Bean Klasse angeben. Die Remote oder Local Schnittstelle legt fest, welche Methoden für EJB-Clients sichtbar sein sollen. Diese Methoden müssen die gleichen Signaturen wie die in der Entity Bean Klasse definierten get- und set-Methoden haben.

Bean Managed Persistenz: Im Gegensatz zur Container Managed Persistence obliegt dem Entwickler bei der Bean Managed Persistence die Aufgabe die Persistenzmechanismuslogik selbst zu realisieren. Der größte Vorteil der Bean Managed Persistence ist die etwas größere Flexibilität in der Gestaltung und im Umgang mit der Persistenzlogik. Der Nachteil dabei ist, dass der Entwickler fundierte Kenntnisse vom Persistenzmedium haben muss. Dies alles ist zudem mit viel Arbeits- und Zeitaufwand verbunden.

Der EJB-Container hat in diesem Fall nur die Durchreich-Funktionalität. Er fordert den Entwickler auf, sich an bestimmte Namenskonventionen zu halten sowie Methodenaufrufe der Callback-Funktionen zu tätigen.

2.2.7 EJB Query Language

Enterprise JavaBeans Query Language (EJB QL) wurde in der EJB-Spezifikation 2.0 eingeführt. Sie ist die EJB Standard-Abfragesprache für Verhaltensdefinition der Finder- und Select-Methoden. EJB QL ist eine deklarative Abfragesprache, ähnlich der Structured Query Language (SQL-92), zugeschnitten für den Gebrauch im abstrakten Persistenzschema bzw. Deployment descriptor der Entity Beans für Container Managed Persistence. In der EJB-Spezifikation wird eine EJB QL Abfrage wie folgt definiert: *"An EJB QL query is a string which must contain a SELECT clause and a FROM clause, and which may contain a WHERE clause"*.

Die Einführung und Verwendung einer solchen EJB QL Abfragesprache ist eine Neuerung in dem Gesamtspektrum der Web-Technologien. Vor allem, wenn man bedenkt, dass die Abfragen extern (und nicht im Programm-Quellcode) im Deployment descriptor mit anderen EJBean Laufzeiteinstellungen erfasst werden. Darüber hinaus abstrahiert EJB QL Abfragen von dem zugrundeliegenden Persistenzmedium. In dem EJB-Container werden Informationen über das eingesetzte Persistenzmedium festgelegt.

Erst zur Laufzeit übersetzt der EJB-Container für das zugrundeliegende Persistenzmedium die EJB QL Anweisungen und führt sie entsprechend aus.

EJB QL hat zwei Abfragemethodentypen:

- Find-Methoden
- Select-Methoden

Finder-Methoden werden in dem Remote home und/oder Local home Interface der Entity Bean definiert und können von EJB-Clients verwendet werden. Als Ergebnis wird eine Referenz auf das gefundene EJBObject (EJBLocalObject) oder eine Referenz auf die Kollektion als `java.util.Collection` von EJBObjects (EJBLocalObjects) zurückgegeben. Der Rückgabetyp der Finder-Methoden ist entweder das Remote (Local) Interface oder eine Objekt-Kollektion. Die Namenskonvention der Finder-Methoden ist ejbFind<*Methoden-Name*>.

Dagegen werden Select-Methoden in der Entity Bean Klasse definiert und sind einzig von der Entity Bean Klasse benutzbar. Die Select-Methoden werden üblicherweise für das Auffinden der anderen verwandten Entity Beans verwendet. Der Rückgabetyp der Select-Methoden ist entweder das EJBObject (EJBLocalObject) oder eine Kollektion als `java.util.Collection` oder `java.util.Set` von EJBObjects (EJBLocalObjects). Alle Select-Methoden werden nach der ejbSelect<*Methoden-Name*> Namenskonvention benannt.

Die Einführung von EJB QL in der EJB-Spezifikation hat eine deutliche Performanzverbesserung, Flexibilität und Portabilität der Container Managed Persistence der Entity Beans gebracht, dennoch hat sie einige Schwachstellen. Besonders hebt sich hier die unvollständige Umsetzung der SQL-Abfragesprache in EJB QL hervor. So fehlt die ORDER BY Klausel sowie viele nützliche Funktionen, wie z.B. COUNT(), MAX(), MIN(), etc. Eine weitere Schwachstelle ist die Weglassung des `java.util.Date` Typs in EJB QL.

2.2.8 Einschränkungen von EJB

Die Enterprise JavaBeans Spezifikation unterliegt einigen Einschränkungen, die die Handhabung und die Anwendbarkeit der EJB beeinträchtigen können. Diese Einschränkungen beziehen sich auf die Implementierung der EJBean (s. Kap. 2.2.2.3). Hier werden einige wichtige Einschränkungen in Anlehnung an [ADAT01], [ROMAN02] und [SUN01] zusammengefasst:

- EJBean darf nicht die Funktionalität der Abstract Window Toolkit (AWT) für die Ausgabe der Informationen auf dem Bildschirm oder Eingabe über die Tastatur nutzen.

- EJBean darf nicht über das Package `java.io` auf Dateien oder Direktorien zugreifen.

- EJBean darf keine Netzwerkverbindungen über Socket aufbauen bzw. nicht die Rolle des Netzwerkservers übernehmen.

- EJBean darf keine statischen Variablen benutzen (statische Konstanten sind erlaubt). Hierfür sollten alle statischen Variablen als final deklariert werden.

- EJBean darf keine Threadsynchronisation anwenden, Threads starten oder stoppen.

- EJBean darf keine Reflection API, um Klassen- und Variabeleninformationen zu erhalten, verwenden.

- EJBean darf nicht direkt File descriptor lesen oder schreiben.

- EJBean darf nicht auf native Bibliotheken zugreifen.

- EJBean darf nie *this* als Parameter eines Methodenaufrufs übergeben oder als Rückgabeparameter liefern, um einem EJB-Client nie die Möglichkeit zu geben direkt auf sie zuzugreifen.

- EJBean darf kein eigenes Sicherheitsmanagement mit Hilfe der Klassen des Packages `java.security` implementieren oder Sicherheitspolicen ändern.

Die meisten dieser Einschränkungen kollidieren mit den Aufgaben und Diensten des EJB-Containers.

2.2.9 Zusammenfassung

In Kapitel 2.2 wurde detailliert auf die Enterprise JavaBeans Spezifikation eingegangen. Der Schwerpunkt lag dabei auf dem Einsatz von EJB in der Anwendungslogik bzw. als Middelware einer Web-basierten Anwendung.

Als erstes wurde ein grober Überblick über die EJB-Architektur gegeben. Es wurden die wesentlichen Bestandteile der EJB-Architektur: EJB-Server, EJB-Container, Enterprise Java Bean und EJB-Client besprochen. Ein allgemeines Verständnis über die Aufgaben und Zuständigkeiten dieser Bestandteile wurde vermittelt. Die Handhabung der Enterprise JavaBeans sowie der technische Hintergrund für die praktische Umsetzung wurden gezeigt. Des weiteren wurde die vom EJB-Konzept für den Beteiligten in einem Entwicklungsprozess vorgeschlagene Rollenverteilung dargelegt. Jede dieser Rollen hat eine bestimmte Aufgabe und muss sich dabei an die von der EJB-Spezifikation festgelegten Verträge halten.

Ein ganz wesentlicher Teil der EJB-Spezifikation sind die EJB Typen: Session, Entity und Message-Driven Beans. Durch sie wird das Modellieren und Realisieren der Geschäfts-, Daten- und Benachrichtigungslogik weitestgehend erleichtert. Sie sind die tragende Säule der technischen Umsetzung des EJB-Konzepts. Dazu gehört ebenfalls der Deployment descriptor, in dem alle Laufzeiteinstellungen und -anpassungen einer EJB-

Anwendung vorgenommen werden können. Diese klare Trennung von Aufgaben und Zuständigkeiten durch die EJB-Typen ermöglicht eine Kapselung einzelner Bestandteile einer Web-basierten Anwendung in die zugehörige Einheiten.

Im Anschluss daran wurde das Konzept des Transaktions- und Persistenzmanagements der EJB-Spezifikation erklärt. Diese zwei Fähigkeiten von EJB haben eine besondere Bedeutung für die Entwicklung der Web-basierten Anwendungen, da sie einen erheblichen Teil der wiederkehrenden Programmieraufgaben aus diesen Bereichen erledigen. EJB setzt damit in der Entwicklung von Web-Technologien die selben Maßstäbe, die in der Softwareentwicklung aus dem Desktop-Bereich angewandt werden. Das Dienstleistungsspektrum der EJB-Spezifikation unterstützt somit in vielerlei Hinsicht die Entwicklung robuster und skalierbarer Web-basierter Anwendungen.

Als eine Neuerung der EJB-Spezifikation 2.0 wurde die EJB QL Abfragesprache besprochen. Der große Vorteil der EJB QL ist, dass die Abfragen nicht im Programm-Quellcode geschrieben werden. Sie werden getrennt in dem Deployment descriptor festgehalten. Dazu sollte erwähnt werden, dass in der bisherigen Entwicklung Web-basierter Anwendungen die SQL-Abfragen immer in der Anwendungslogik erfasst wurden. EJB unterstützt somit eine weitere Stufe der Trennung von Zuständigkeiten und bietet eine solide Abstraktion von dem darrunterliegenden Persistenzmedium.

Dass jede Technologie oder Spezifikation ihre Einschränkungen besitzt wurde ganz zuletzt auch für die EJB-Spezifikation 2.0 deutlich.

2.3 Web-Komponenten und EJB

Die Java 2 Enterprise Edition (J2EE) Spezifikation definiert eine Plattform für Web-basierte Anwendungsentwicklung, die Enterprise JavaBeans, Servlets und Java Server Pages (JSP) umfasst. Die J2EE Spezifikation erhebt den Anspruch, die Lücke zwischen Web-Komponenten und Enterprise JavaBeans zu schließen. Während Web-Komponenten für die Interaktionslogik zuständig sind, stellen Enterprise JavaBeans ein robustes, skalierbares und transaktionsfähiges Komponentenmodell für die Entwicklung der Anwendungslogik bereit. J2EE bietet diesen Technologien die Möglichkeit eine gemeinsame Web-Plattform zu bilden.

In diesem Kapitel werden einzelne Web-Komponenten, Servlets und JSP als Vertreter der Interaktionslogik in der Web-basierten Anwendungsentwicklung kurz vorgestellt und erläutert. Das Zusammenspiel der Servlets und JSP mit Enterprise JavaBeans in einer J2EE Laufzeitumgebung wird beschrieben. Am Ende dieses Abschnitts wird das Zusammenstellen der Web-Komponenten in sogenannten Enterprise-Produkten bzw. -Archiven für den Web-Einsatz erklärt.

2.3.1 Servlets

Servlets basieren auf der Java Programmiersprache und wurden von SUN Microsystems entwickelt. Sie sind Javas Schlüsselkomponente für server-seitige Web-Entwicklung. Servlets bieten eine einfache und leistungsfähige Schnittstelle für die Generierung von dynamischen Web-Seiten. Gewöhnlicherweise wird die Servlets Schnittstelle als Web-Server-Erweiterung angeboten. Obwohl Servlets für verschiedene Request-Response Protokolle verwendet werden können, werden sie am häufigsten für den Einsatz mit dem HTT-Protokoll im Web angewendet.

Servlets arbeiten nach dem gleichen Prinzip wie CGI Skripte (z.B. Perl, PHP, etc.). Daten werden über Request-Anfragen übermittelt und nach deren Verarbeitung über Response an den Web-Browser zurückgegeben. Im Unterschied zu CGI Skripten, die zu jeder Request-Anfrage einen neuen Prozess auf dem Server generieren, werden einmal aufgerufene Servlets von dem *Servlet Engine* (Servlet Container) aufbewahrt und können weitere Anfragen im gleichen Prozessraum bedienen (s. Kapitel 2.1). Dadurch werden weniger Ressourcen verbraucht und eine Performanzsteigerung der Request-Anfragen-Bearbeitung wird erreicht. Ein weiterer Vorteil von Servlets ist, dass der gesamte Java-Sprachumfang zur Verfügung steht, inklusive der Bibliotheken. Damit lassen sich z.B. Datenbankabfragen über JDBC (Java Database Connectivity), Zugriffe auf Business-Systeme oder einfache Dateizugriffe realisieren.

Servlets sind den Sessions Beans ähnlich, da beide Aktionen in Vertretung für Web-Clients ausführen. Sie können die Web-Clients eindeutig durch Sessions (`HttpSession`) identifizieren und haben unmittelbaren Zugriff auf das Persistenzmedium. Dennoch stellen Servlets keine impliziten Dienste für Transaktions-, Persistenz- und Ressourcenverwaltung zur Verfügung. Darüber hinaus bieten sie keine feine Abstraktions- und Modellierungsmöglichkeit innerhalb der Anwendungslogik, wie es bei

EJB der Fall ist, sondern eine Trennung innerhalb in Interaktions- und Anwendungslogik. Die Abbildung 6 präsentiert Web-Komponenten und EJB gemeinsam in einem J2EE-Application-Server. Dabei wird ersichtlich, dass sowohl der Web-Container als auch der EJB-Container gleiches Dienstleistungsspektrum anbietet.

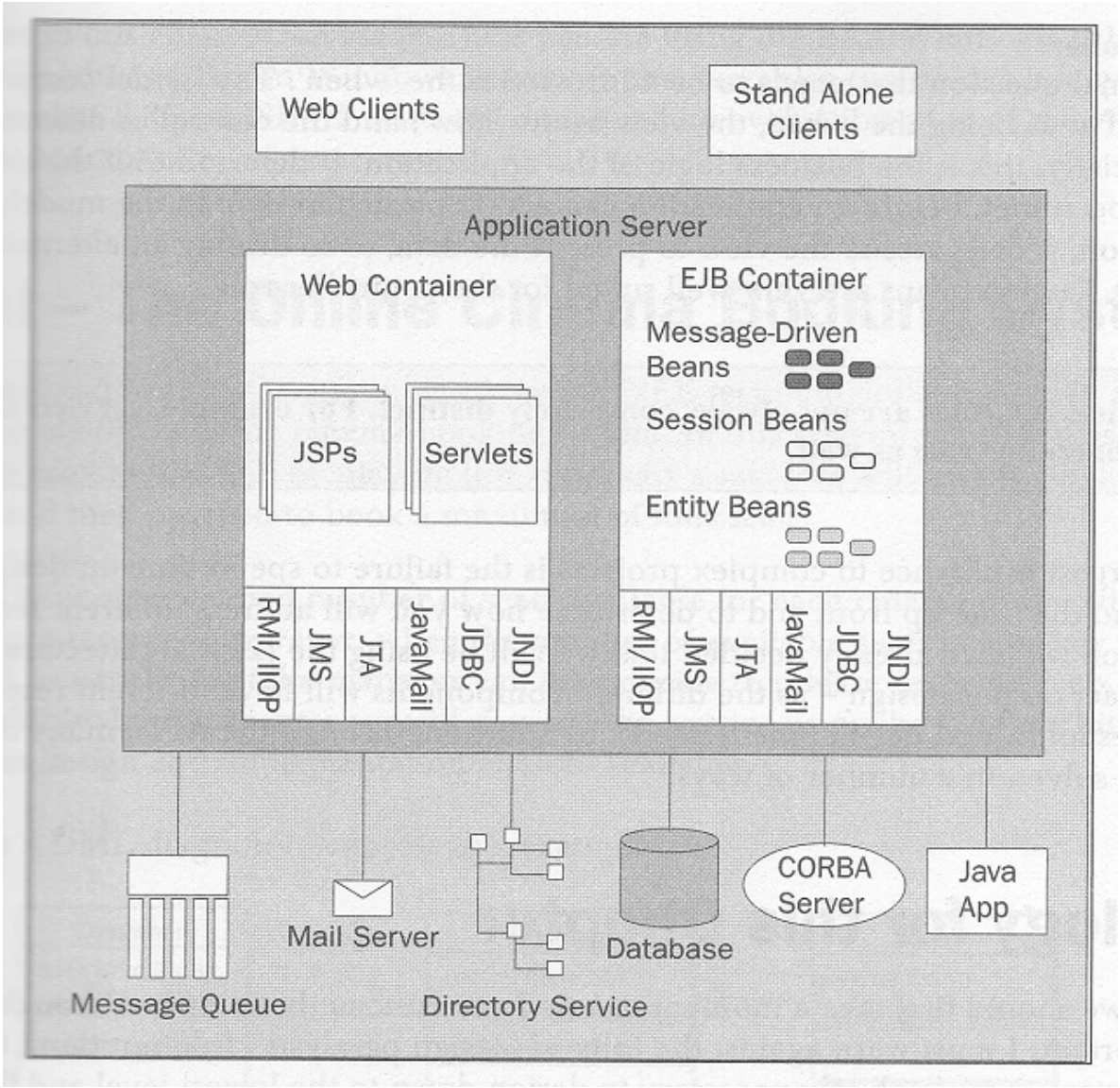

Abbildung 6 - Dienste eines J2EE Application-Servers nach [ADAT01]

Servlets sowie andere Web-Komponenten werden in der EJB-Umgebung als EJB-Clients (s. Kapitel 2.2.2.4) behandelt. Das Lokalisieren von EJBeans wird in Servlets mit Hilfe von JNDI Enterprise Naming Context (ENC) durchgeführt. Nach dem Lokalisieren der EJB können Servlets über die Remote oder RemoteHome Schnittstelle sondierende und verändernde Operationen durchführen. Abbildung 7 zeigt ein Architektur-Beispiel mit Servlets/JSP in der Interaktionslogik, EJB in der Anwendungslogik und einer relationalen Datenbank auf der Persistenzebene.

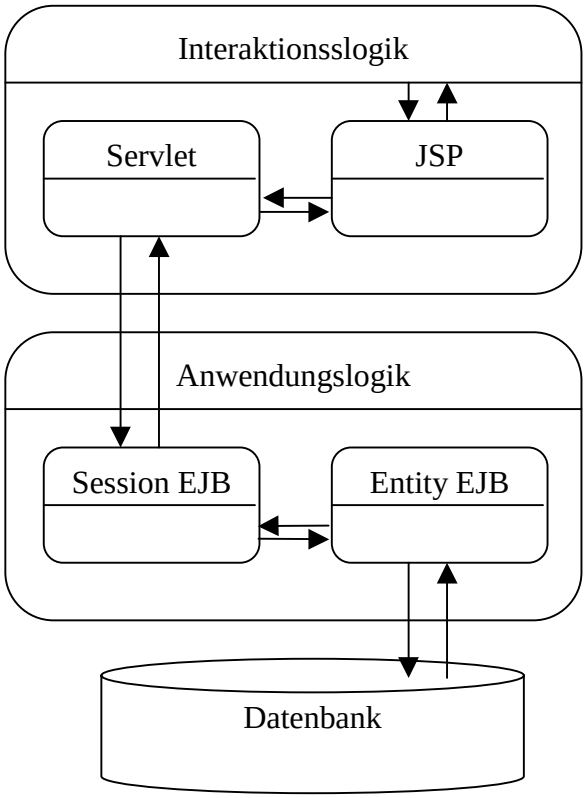

Abbildung 7 - Ein Architektur-Beispiel: Servlets/JSP mit EJB

2.3.2 Java Server Pages (JSP)

Java Server Pages (JSP) ermöglichen eine schnelle Erstellung von Web-Komponenten, mit sowohl statischen als auch dynamischen Inhalten. JSP basieren auf der gleichen Java Technologie wie Servlets, aber sie bieten eine andere Herangehensweise in der Erstellung von Web-Komponenten. Sie ermöglichen die direkte Einbettung von Java-Sprachkonstrukten in die Interaktionslogik. Somit können JSP als eine Web-Skriptsprache agieren, die server-seitige Konstrukte und Mechanismen für dynamische Erzeugung von Web-Seiten unterstützen.

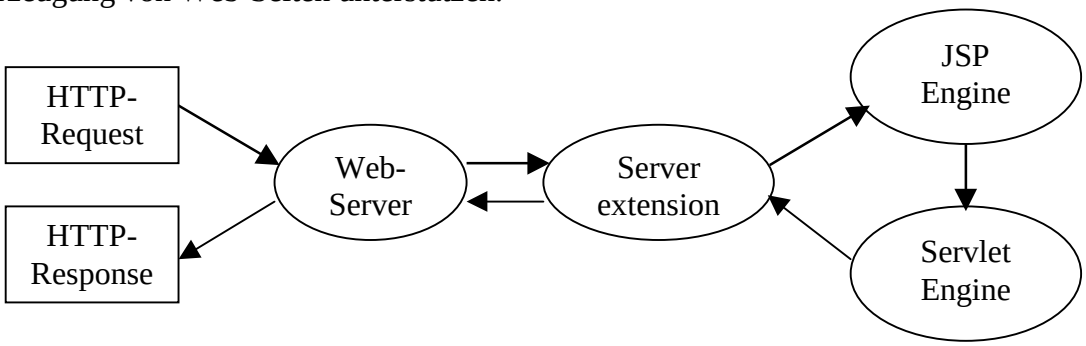

Abbildung 8 - Der Aufruffluss einer JSP-Seite

Java Server Pages werden bei ihrem ersten Aufruf vom *JSP Engine* in Servlets kompiliert. Danach werden sie vom *Servlet Engine* aufbewahrt und verwaltet. Beim

nächsten Aufruf führt der *JSP Engine* schon geladene Servlets, sofern sie nicht geändert wurden, aus. Wurden sie jedoch geändert, dann veranlasst *JSP Engine* eine Neukompilierung der JSP (s. Abbildung 8).

Genau wie bei den Servlets werden EJB über JNDI ENC lokalisiert. Der Vorteil der JSP gegenüber Servlets ist, dass JSP das Erstellen sprachlicher Konstrukte (*Tag Library*), die direkten Zugriff auf Eigenschaften der Enterprise JavaBeans realisieren können, ermöglichen. Mit Hilfe dieser sprachlichen Konstrukte kann man das Handling der EJB-Anwendungslogik kapseln. Als wiederverwendbare sprachliche Konstrukte sind Tag Libraries somit in unterschiedliche J2EE-Web-Komponenten einsetzbar.

Manche EJB-Serverhersteller haben die EJB-Spezifikation insofern angepasst, dass Web-Komponenten wie JSP oder Servlets auf lokale Interfaces von EJB zugreifen können. Daher unterstützen diese EJB-Serverhersteller das Ausführen von EJB, JSP und Servlets in der gleichen Java Virtual Machine, was zu einer deutlichen Performanzsteigerung der Web-Anwendung führt.

2.3.3 Packaging Web Components

Web-Komponenten werden in sogenannten Web-Containern (z.B. in einem von Tomcat, JRun, etc.) eingesetzt und ausgeführt. Der Web-Container stellt Dienstleistungen, wie Anfragebearbeitung, Sicherheitsaspekte, Nebenläufigkeit und Lebenszyklusmanagement bereit. Darüber hinaus haben Web-Komponenten Zugang zu Java 2 Enterprise Edition (J2EE) Schnittstellen wie, JNDI, Transaktionen, Email-Dienst, etc. Manche Eigenschaften einer Web-Komponente, wie Konfigurationsparameter und Laufzeiteinstellungen, können somit genau wie unter EJB in dem sogenannten *Web Application Deployment Descriptor* vor dem Einsatz im Web-Container festgelegt werden. Danach werden Web-Komponenten zusammen mit dem *Web Application Deployment Descriptor* in einem *Web Application Archive* (WAR) erfasst. Die Struktur des WAR-Archivs ist dem Java Klassen-Archiv JAR ähnlich.

In ähnlicher Weise definiert J2EE eine hierarchische Archivstruktur EAR für J2EE Applikationen. Die Archivstruktur wird in einem XML Deployment descriptor zusammengefasst. Diese fassen Enterprise JavaBeans und Web-Komponenten sowie andere Elemente wie Icons, Beschreibungen, etc. in einer J2EE Applikation zusammen (s. Abbildung 9).

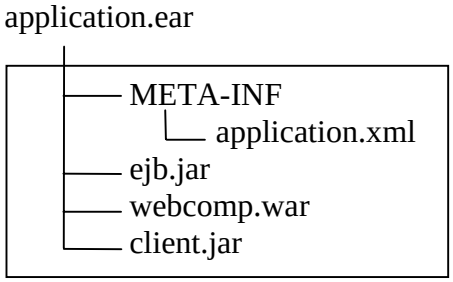

Abbildung 9 - J2EE EAR Archivstruktur

2.3.4 Zusammenfassung

Die J2EE stellt eine gemeinsame Plattform für die Realisierung sowie den Einsatz von EJB und Servlets/JSP in der Web-basierten Anwendungsentwicklung dar. Die Servlets und JSP ermöglichen eine einfache Handhabung und Umgang mit EJB. Durch Servlets/JSP werden keine weiteren vermittelnden Komponenten oder Schichten für die Interaktion mit EJB benötigt. Durch die Java-Sprachkonstrukte kommt die Interaktion sowie das Zusammenspiel der Servlets/JSP mit EJB über JNDI ENC zustande. Die wesentlichen Unterschiede zwischen der Servlets-/JSP- und EJB-Technologie für den Einsatz in der Web-basierten Anwendungsentwicklung wurden kurz umrissen. Hierfür wurde die Servlets- und JSP-Technologie beschrieben. Wie die praktische Umsetzung der Servlets/JSP in der Interaktionslogik mit EJB in der Anwendungslogik aussieht wird in Kapitel 3.3 behandelt.

Am Ende dieses Kapitels wurde das Zusammenstellen der Web-Komponenten mit den EJB-Komponenten gemeinsam in einem Web Application Archive bzw. Enterprise Application Archive erläutert.

2.4 EJB in der Praxis?

Die Enterprise Java Beans Architektur bietet ein server-seitiges Komponentenmodell für die Realisierung von robusten, skalierbaren und mehrbenutzerfähigen Web-basierten Anwendungen. Die EJB-Architektur stellt hierfür eine Reihe von Diensten für die Entwicklung solcher Web-basierten Anwendungen bereit. Diese vom EJB-Komponentenmodell impliziten und expliziten Dienste umfassen Persistenz-, Transaktions-, Sicherheits-, Zugriffs-, Nebensläufigkeits-, Ressourcen- und Lebenszyklusmanagement. Diese stellen besondere Erleichterung für den Anwendungsentwickler dar, weil er sich nicht mit der betriebssystemnahen Programmierung beschäftigen muss, sondern sich voll auf die Realisierung der Anwendungslogik (vgl. [ADAT01], S. 480) konzentrieren kann. Weiterhin ermöglicht die Plattformunabhängigkeit der EJB durch die Verwendung der Java-Programmiersprache[19] deren Einsatz in verschiedenen Plattformen und in heterogenen Umgebungen. Darüber hinaus unterstützt die EJB-Architektur eine klare Trennung von Interaktions-, Anwendungs- und Datenlogik sowie die Realisierung schichtenbasierter Web-basierter Anwendungen (s. Abbildung 10).

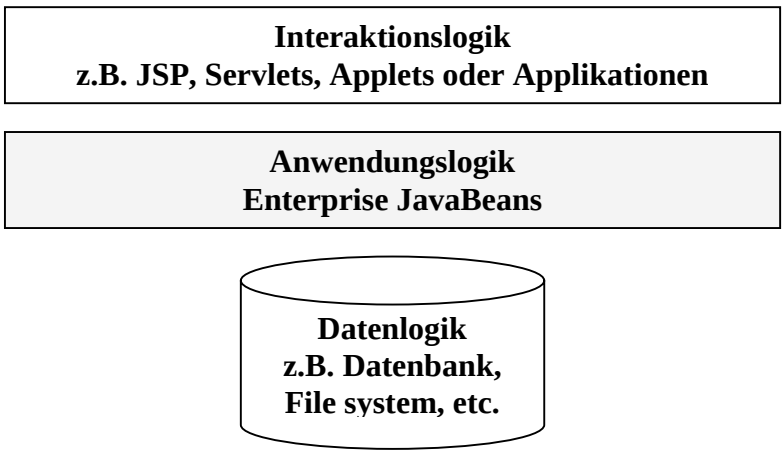

Abbildung 10 - Schichtenmodell-Beispiel einer EJB-Anwendung

In der jetzigen neuen EJB-Spezifikation 2.0 hat das EJB-Komponentenmodell durch klar definierte Schnittstellen und Zuständigkeiten eine deutliche Verbesserung gegenüber den älteren Versionen erreicht. Eine dieser Verbesserungen ist die vollständige Unterstützung der automatischen Container Managed Persistence für Entity Beans. Der Nachteil dabei ist, dass sie nicht rückwärts kompatibel zu den vorigen Versionen ist. Diese Eigenschaft war in den vorigen Versionen unzulänglich von der EJB-Spezifikation spezifiziert worden und dadurch von den EJB-Server- und EJB-Container-Herstellern unvollständig umgesetzt worden. Zu den wesentlichen Neuerungen der EJB-Spezifikation 2.0 gehört der asynchrone Java Message Service (JMS) bzw. die Message-Driven Beans und die EJB QL Abfragesprache.

[19] Ansatz von Java: “Write Once Run Anywhere”

Die Einführung von Message-Driven Beans, EJB QL sowie die Unterstützung der Container Managed Persistence Relationsmodelle für Entity Beans in der EJB-Spezifikation 2.0 führt zu noch breiteren Einsatzmöglichkeiten des EJB-Komponentenmodells in der Entwicklung Web-basierter Anwendungen. Gemeinsam mit Web-Komponenten unterstützt das EJB-Komponentenmodell die Realisierung von stabilen und kompatiblen Web-Plattformen für server-seitige unternehmerische Anwendungen.

Die wesentlichen Vorteile von Enterprise JavaBeans für die Entwicklung Web-basierter Anwendungen in der Praxis sind (in Anlehnung an [ADAT01], [MONSON01], [ROMAN02]):

- Kapselung: Durch die Kapselung der Anwendungslogik in der Middelware ermöglicht sie eine zentrale Anwendungslogikprozesskontrolle.
- Trennung von Zuständigkeiten: Das EJB-Komponentenmodell unterstützt die Trennung von Präsentations-, Anwendungs- und Datenlogik.
- Komponentenbasiert: EJB ist Komponentenbasiert aufgebaut. Sie unterstützen jedoch auch eine Komponentenbasierte Entwicklung der Anwendungslogik.
- Zentrales Management: Als server-seitiges Komponentenmodell stellen EJB Mittel für eine zentrale Management-, Ressourcen-, Sicherheitsmaßnahmen-, Nebenläufigkeits- und Versionskontrolle zur Verfügung.
- Laufzeiteinstellungen: Laufzeiteinstellungen und -anpassungen einer EJB-Anwendung können mit geringem Aufwand vorgenommen werden. Es bedarf keiner neuen Kompilierung der Anwendung.
- Implizite Dienste: Durch die implizite Transaktions- und Persistenzverwaltung erleichtern und übernehmen EJB einen Teil der Realisierung der Anwendungslogik.
- Thinner Clients: EJB unterstützen die Implementation der Thinner Clients (ultra-lightweight client interface) durch die Integration von J2EE Technologien, wie Java Servlets/JSP.
- Herstellerunabhängig: EJB-Architektur ist Herstellerunabhängig.

Neben diesen Vorteilen weisen EJB einige Nachteile für die Praxis auf (in Anlehnung an [ADAT01], [MONSON01], [ROMAN02]):

- Performanzkosten: Durch die Interaktion des externen EJB-Clients mit der EJB-Anwendung über die Remote Method Invocation (RMI), d.h. über Remote bzw. RemoteHome Schnittstellen (s. Kapitel 2.2.2.3), entstehen erhebliche Performanzeinbußen.
- Zerbrechlichkeit der EJB-Architektur: Bei der Anwendung der EJB-Architektur muss darauf geachtet werden, dass einzelne Komponenten entsprechend der EJB-Vorgehensweise implementiert werden.
- Erhöhte Komplexität: EJB-Anwendungen sind durch Vereinigung von mehreren Diensten und Kommunikationsprotokollen komplex.
- Höhere Lernkurve: Das erlernen der EJB-Technologie erfordert von Entwicklungsprozessbeteiligten hohen Einsatz.

- Technology for Technology's Sake: EJB-Technologie sollte nur dann angewandt werden, wenn das komplexe Dienstleistungsspektrum der EJB tatsächlich in Anspruch genommen werden wird.
- Hohe Investitionskosten: Die Kosten für n-tier Architekturen sind erheblich höher als standalone bzw. traditionelle Client-Server Anwendungen. Diese erstrecken sich von Anschaffungskosten für Hardware und Software (EJB-Server oder Application-Server) bis hin zu Entwickler- und Entwicklungszeitkosten.

Die Tabelle 4 gibt zusammenfassend die Vorteile und Nachteile der EJB in der Praxis wieder.

Vorteile	***Nachteile***
- Kapselung - Trennung von Zuständigkeiten - Komponentenbasiert - Zentrales Management - Laufzeiteinstellungen - Implizite Dienste - Thinner Clients - Herstellerunabhängig	- Performanzkosten - Zerbrechlichkeit der EJB-Architektur - Erhöhte Komplexität - Höhere Lernkurve - Technology for Technology's Sake - Hohe Investitionskosten

Tabelle 4 - Vorteile und Nachteile der EJB für die Entwicklung Web-basierter Anwendungen

Im nächsten Kapitel wird anhand des Fallbeispiels aus dem eGovernment-Bürgerprozessportal-Bereichs die Eignung und Tauglichkeit der EJB in der Entwicklung von Web-basierten Anwendungen in der Praxis untersucht und bewertet.

3 EJB für ein Bürgerprozessportal

Dieses Kapitel erörtert die Eignung von EJB für die Realisierung eines eGovernment-Bürgerprozessportals am Fallbeispiel „Briefwahlantrag“. Als erstes wird im Kapitel 3.1 eine kurze Einführung der allgemeinen Anforderungen und Besonderheiten von eGovernment-Bürgerprozessportalen gegeben.

Im Kapitel 3.2 wird der im Projektseminar nach dem Serviceflow Management (SfM) Ansatz entwickelte Web-basierte SfM Prototyp eines Web-basierten Servicepoints, dabei insbesondere die Anwendungslogik, vorgestellt. Einzelne Komponenten des Web-basierten SfM Prototyps werden näher besprochen. Nach bestimmten softwaretechnischen Qualitätskriterien, die auf die industrielle Anwendbarkeit der Anwendungslogik hinauslaufen, werden in Kapitel 3.2.4 Schwachstellen des Web-basierten SfM Prototyps inspiziert.

Eine Re-Implementierung des mit Enterprise JavaBeans (EJB) entwickelten Web-basierten SfM Prototyps wird im Kapitel 3.3 vorgestellt. Dabei werden einzelne Bestandteile der neuen Architektur dargelegt und am Fallbeispiel „Briefwahlantrag“ angewandt und besprochen. Eine Bewertung des neu entwickelten EJB Web-basierten SfM Prototyps nach softwaretechnischen Entwurfsprinzipien und Qualitätskriterien sowie im Bezug auf den Anwendungskontext eGovernment-Bürgerprozessportal findet im Kapitel 3.3.3 statt.

3.1 eGovernment-Bürgerprozessportal: Einführung und allgemeine Anforderungen

Seit längerer Zeit nutzen Unternehmer die Möglichkeiten und Chancen des Internets für sich. Allmählich werden durch das Internet auch in anderen Lebensbereichen Veränderungen bemerkbar. Diesen Veränderungen können sich auch Regierung und Verwaltung, vor allem im Umgang mit dem Bürger, nicht entziehen. Daher ist in einer digital vernetzten Gesellschaft die Reaktionszeit für erfolgreiches Handeln kürzer geworden. Für den Bürger und für Unternehmen wird der elektronische Zugang zur öffentlichen Verwaltung und die digitale Abwicklung von Behördenverfahren immer wichtiger. Der Staat sollte den Forderungen der Bürger und Unternehmen nach Bedürfnisorientierung, Qualität, Schnelligkeit und Transparenz der staatlichen Leistungen entgegenkommen. Der Einsatz der neuen Web-Technologien kann mit dazu beitragen, diese Anforderungen zu bewältigen.

Mit Hilfe von Informations- und Kommunikationstechnologie sollte der Staat das Ziel verfolgen die staatliche Leistungserbringung zu optimieren und die Beteiligung der Bürger und Unternehmen am Entscheidungsprozess zu erhöhen. eGovernment soll als Antwort auf diese Anforderungen verstanden werden, mit denen sich der Staat heute und in Zukunft konfrontiert sieht. Eine allgemeine Definition für eGovernment gibt Schedler ([SCHEDL00], S. 35) in seinem Artikel: „eGovernment und neue Servicequalität in der Verwaltung?“ folgendermaßen:

> *„Electronic Government ist eine Organisationsform des Staates, welche die Interaktion und Wechselwirkungen zwischen dem Staat und den Bürgern, privaten Unternehmen, Kunden und öffentlichen Institutionen durch den Einsatz von modernen Informations- und Kommunikationstechnologien (IKT) integriert.“*

Demnach umfasst eGovernment die Unterstützung der Beziehungen, Prozesse und der politischen Partizipation innerhalb der staatlichen Stellen auf allen Ebenen sowie zwischen den staatlichen Stellen und all ihren Anspruchsgruppen durch die Bereitstellung geeigneter Interaktionsmöglichkeiten mittels elektronischer Medien (vgl. [SPAH00]).

Was Bürger und Unternehmen von einem eGovernment-Bürgerprozessportal erwarten (in Anlehnung an [KÄSTN00] und [SCHEDL00]):

- elektronischen Zugriff auf öffentliche Informationen
- elektronischen Austausch von Dokumenten und Abwicklung von Behördenverfahren
- besseren Zugang zur öffentlichen Verwaltung – jederzeit, kostengünstig und vor allem auch außerhalb der Amtsstunden
- bessere Qualität staatlicher Leistungen – schneller, „one-stop-shop“ für alle Verwaltungsebenen, nahtlose Verwaltungsverfahren
- kürzere Dauer von verwaltungsinternen Abläufen
- höhere Transparenz der Leistungen von Staat und Verwaltung
- Senkung der Kosten von Leistungserbringung der Verwaltung (z.B. elektronischer Austausch von Daten statt erneuter Eingabe)

Aus der eGovernment-Definition und den Anforderungen der Bürger und Unternehmen ergaben sich für die klare Trennung von Leistungserbringung folgende Teilbereiche ([GISLER01]):

- eAssistance
- eAdministration
- eDemocracy

Unter dem Begriff eAssistance wird der Einsatz von Informations- und Kommunikationstechnologien zur Unterstützung der alltäglichen Lebensgestaltung verstanden. Bezeichnend für diese Art der elektronischen Dienstleistungen ist, dass sie nicht zu einer Transaktion nach öffentlichem Recht im Sinne der eAdministration führen ([GISLER01], S. 35). Die drei wichtigsten Arten von eAssistance-Dienstleistungen sind:

- Allgemeine Informationen (Öffnungszeiten der Behörden, Ortsplan, etc.)
- Werbung in eigener Sache (Standortattraktivität)
- Unterstützung zur Erfüllung gesetzlicher Aufträge, oft im Bereich der sozialen Wohlfahrt (Jobbörsen, Weiterbildungsveranstaltungen, etc.)

eAdministration bezeichnet den Einsatz der Informations- und Kommunikationstechnologien zur Unterstützung des internen und externen Behördenverkehrs, also des amtlichen Geschäftsverkehrs ([GISSPA01], S. 23). Anwendungsbeispiele der eAdministration-Dienstleistungen sind:

- Elektronische Steuererklärung (eTaxes)
- Elektronische Volkszählung (eCensus).

Unter dem Begriff eDemocracy werden Diskussionen darüber zusammengefasst, wie sich die informations- und kommunikationstechnische Infrastruktur dazu nutzen lässt, demokratische Kommunikations- und Beteiligungsstrukturen zu beleben. Die Ausgestaltungsmöglichkeiten reichen von der Information über Sachgeschäfte, elektronische Diskussionsforen bis hin zu elektronischen Abstimmungen oder Wahlen, dem eVoting oder der eElection. Dabei wird unterschieden zwischen eDemocracy im engeren und im weiteren Sinne. Im engeren Sinne steht eDemocracy für die elektronische Durchführung der jeweils verfassungsrechtlich vorgesehenen formalen Entscheidungsakte. Im weiteren Sinne umfasst eDemocracy auch die Bemühungen, die Bürger stärker in politische Meinungsbildungs- und Selbstorganisationsprozesse einzubeziehen ([GISSPA01], S. 23).

Nach dieser eGovernment-Einführung neigt man sofort dazu eGovernment als „eBusiness des Staates“ zu bezeichnen bzw. als ein weiteres ergänzendes eBusiness-Leistungsspektrum zu betrachten. So stellt sich folgende Frage für die Softwareindustrie in der Entwicklung von eGovernment-Bürgerprozessportalen: Inwiefern gibt es Zusammenhänge zwischen eGovernment und eBusiness? Dass eGovernment dem eBusiness nicht gleichzusetzen ist, und dass die Anforderungen und Kriterien unterschiedlich sind, wird in der Tabelle 3 von [GISLER00] deutlich.

	eBusiness	eGovernment
Zielpublikum	Angesprochenes Kundensegment	alle Bürger mit ihren Rechten und Pflichten
Produkte	Homogenes Produktsortiment	Diversifiziertes Produktsortiment
Markt	Wettbewerb	Monopol
Marktausrichtung	Gemäss Nachfrage	Gemäss Rechtsquellen
Organisation	Flexibel	Gesetzliche Basis
Prozessgestaltung	Flexibel	Gesetzliche Basis
Einfluss der Führungsebene	Durchsetzbarkeit	Politik Internat. Abkommen
Reaktionsgeschwindigkeit	Hoch	Tief

Tabelle 5 - eBusiness vs. eGovernment [GISLER00]

Ein Modellkonzept für eGovernment-Bürgerprozessportale im Internet-Zeitalter könnte (in Anlehnung an [LENK00]) so erreicht werden:

- Vorinformationen: eGovernment-Bürgerinformationssysteme (Bürgerprozessportale) sollen Vorinformationen liefern und auf die richtige Anlaufstelle verweisen.
- Online-Formulare: Angeschlossen hieran können Formularserver Zugangshilfen bieten, auch wenn das ausgefüllte Formular (noch) nicht online übermittelt werden kann.
- Personalisierung: Für die Kontaktaufnahme mit der Verwaltung ist den Bürgern Zugang zu eröffnen. Ferner ist das Bürgeranliegen mit den dafür vorgesehenen Leistungsarten in einer Personalisierung zusammenzuführen, die einem Vertragsschluss einspricht.
- Geschäftsprozess-Reorganisation: Der Prozess der Leistungserstellung ist dabei nach Möglichkeit neu zu gestalten, um die Vorteile des Zusammenwirkens von Front-Office und Back-Office in telekooperativem Arbeiten über Entfernungen und Organisationsgrenzen hinweg voll auszuschöpfen.
- Gute Interaktionsgestaltung: Die Interaktionslogik sorgt für optimale Kommunikation von Bürgern mit den zuständigen Stellen während der gesamten Abwicklung des Vorgangs.
- Sensitive Daten: Die Empfindlichkeit der Bürgerdaten sowie Datenschutz sind in der Realisierung von Bürgerprozessportalen besonders zu beachten.
- Redundanz: Auf der technischen Ebene ist das Ausfallrisiko einzelner Komponenten zu minimieren. Ein redundantes Komponentenmodell sowohl auf der Hardware- als auch Softwareebene sollte gewährleistet sein.
- Bearbeitungszeit: Trotz des großen Anfragevolumens, das eGovernment-Bürgerprozessportale haben können, sollten die Antwortzeiten zwischen Bürger und Verwaltung nach Abwicklung des Vorgangs in einem zumutbaren Zeitrahmen liegen.

Aus der beschriebenen Herangehensweise in der Erstellung von eGovernment-Bürgerprozessportalen können einige bedeutsame Kriterien für die softwaretechnische Umsetzung hervorgehoben werden. Die Tabelle 6 stellt einige dieser Kriterien, die zum Teil einen hohen Stellenwert im Bereich eGovernment einnehmen, dar.

Kriterien	***Besonderheiten im eGovernment Bereich***
Prozessportal	organisationsübergreifend in verteilten Umgebungen; empfehlenswert eine Personalisierung
Authentifizierung- und Zugriffskontrolle	sensitive Daten jedes einzelnen Bürgers; empfehlenswert strenge Sicherheitsvorkehrungen
Skalierbarkeit und Robustheit	skalierbare und robuste Systeme; empfehlenswert automatische Persistenz- und Transaktionsunterstützung
Prozesspooling und Nebenläufigkeit	sparsamer Umgang mit Betriebsmitteln; empfehlenswert automatische Prozesspooling- und Nebenläufigkeitsunterstützung
Mehrbenutzerfähigkeit	hohes Anfragevolumen; empfehlenswert automatische Unterstützung von Clustering
Dynamische Modellierung und situative Anpassung	schnelle Änderungen im System; empfehlenswert zentrale Stelle für Laufzeitveränderungen und -anpassungen
Wiederverwendbarkeit und Integration	generische Lösungen; empfehlenswert einheitliche technische Basis

Tabelle 6 - Besonderheiten der eGovernment-Bürgerprozessportalen

Da die Organisation und Verwaltung des eGovernment-Dienstleistungsspektrums innerhalb eines eGovernment-Bürgerprozessportals einen erheblichen Aufwand in der Gestaltung der Informations- und Kommunikationstechnologie darstellt, bietet der Einsatz der EJB-Technologie, die besonders stark auf einige dieser Aspekte eingeht, geeignete Konstrukte für deren Realisierung. In Kapitel 3.3.3.3 wird die Einsetzbarkeit der EJB in einem eGovernment-Bürgerprozessportal untersucht und bewertet.

3.2 Der Web-basierte SfM Prototyp

In diesem Kapitel wird zunächst der Ansatz des Serviceflow Managements (SfM) erläutert. Im Anschluss daran wird der Projektverlauf im Rahmen des Projektseminars **„**Service als Leitbild: Softwareunterstützung für Dienstleister**“** im Fachbereich Informatik, Universität Hamburg beschrieben. Der Web-basierte SfM Prototyp eines Web-basierten Servicepoints am Fallbeispiel „Briefwahlantrag“ des eGovernment-Bürgerprozessportals www.hamburg.de mit dem Schwerpunkt auf der Anwendungslogik wird vorgestellt. Auf die grobe und feine Architektur des Web-basierten SfM Prototyps wird eingegangen. Im Folgenden werden die Schwachstellen des SfM Prototyps nach Skalierbarkeit, Robustheit, Portabilität, Sicherheit, Wiederverwendung und Integration analysiert.

3.2.1 Serviceflow Management

Im Projektseminar wurde der Serviceflow Management (SfM) Ansatz als Grundlage für den Bau des Web-basierten SfM Prototyps verwendet. Der Ansatz von Serviceflow Management beruht auf der Unterstützung und Verwaltung des Flusses einer organisationsübergreifenden Dienstleistung, die als Folge von Teilleistungen erbracht wird. Die einzelnen erbrachten Teilleistungen werden unter Einbeziehung der Kunden- bzw. Bürgerbedürfnisse vom Dienstleister über Raum-, Zeit- und Teamgrenzen zu einem zusammenhängenden *Serviceflow*, der eine kontinuierliche, umfassende Gesamtleistung darstellt, zusammengefasst. Dieser Ansatz basiert auf der Metapher vom „Fluss der Dienstleistung“, obwohl hier nicht die Dienstleistung fließt, sondern die Dienstleistungs-Prozessrepräsentation (vgl. [KLIWET00]).

Serviceflow Management wird von [KLIWET00] folgendermaßen definiert:

> *„Serviceflow Management organisiert und unterstützt die Durchführung der Leistungserbringung gemäß der situativen Erfordernisse im Verlauf des Serviceflows auf der Basis von erprobten Standardabläufen. Es unterstützt sowohl individuelle Serviceflows als auch ihre parallele Erbringung.“*

Diese Definition von Serviceflow Management stellt ins Zentrum den Service (Dienstleistung). Sie umfasst die Leistung einer Organisation im Spannungsfeld von Routinisierung und individueller Kundenorientierung.

Serviceflow Management stütz sich auf folgende Service Definition von Klischewski und Wetzel [KLIWET00]:

> “*Service ist eine unternehmerische Leistung, die ihre Wertschöpfung darauf gründet, die Bedürfnisse des Kunden zu erkennen und zu befriedigen. Dafür werden nach Möglichkeit betriebliche Standardabläufe auf die Erfordernisse der jeweiligen Dienstleistungssituation angepasst*”.

Nach dieser Definition wird sowohl der Dienstleister als auch der Kunde in den Mittelpunkt des Dienstleistungsprozesses gestellt. Dem Kunden wird somit ermöglicht jederzeit während des gesamten Verlaufs in den Prozess einzugreifen. Aus diesem Grund sieht sich der Kunde im Fluss des Dienstleistungsprozesses als fester Teil dieses Prozesses, dem Dienstleistungen folgen bzw. begleiten. Dadurch wird der Verlauf des Dienstleistungsprozesses flexibler gemacht, aber andererseits stellt es eine große Herausforderung für den Dienstleistungserbringer dar. Er muss damit rechnen, dass der Dienstleistungsprozess sich situativ über Raum-, Zeit- und Organisationsgrenzen erstreckt und sich auf komplexe, kundenspezifische Hintergrundprozesse abstützt.

Der Serviceflow (Fluss von Dienstleistungen) wird von organisationsübergreifenden Serviceerbringern als Einzelleistung in den sogenannten „Servicepoints“ erbracht. Jeder Servicepoint bietet eine oder mehrere spezifische Dienstleistungen. In jedem Servicepoint werden Aktivitäten oder Operationen einer Aufgabe abgearbeitet. Dies geschieht in Interaktion mit den Kunden oder auch automatisch. Ein Servicepoint muss kein physikalischer Ort sein, sondern kann auch ein virtueller Ort sein, der für die entsprechende Einzelleistung zuständig ist. Darüber hinaus stellt ein Servicepoint dem Kunden Hilfsmittel zur Verfügung, die zur Erledigung der Aufgaben benötigt werden, aber auch situativ an sein Anliegen angepasst werden können. Die Abbildung 11 zeigt ein Serviceflow-Modell für das Fallbeispiel „Prozessportal für Serviceflow Management/eGovernment Hamburg“.

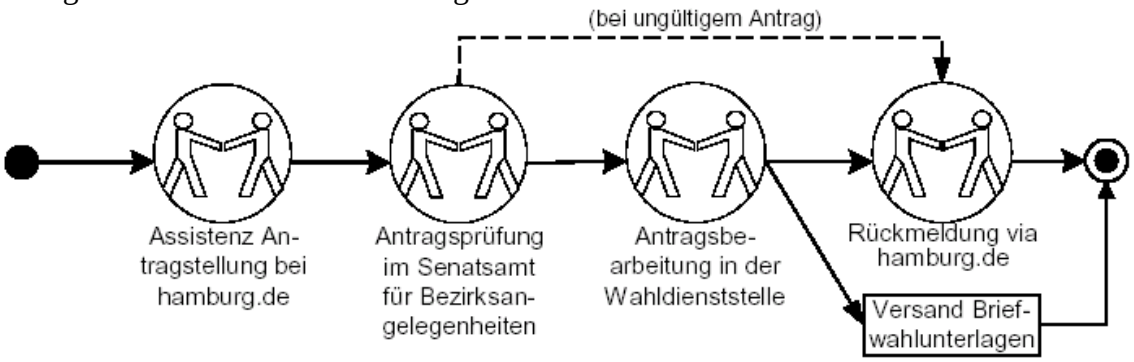

Abbildung 11 - Beispiel eines Serviceflow-Modells nach [KLIWET00]

Die wichtigsten Merkmale des Serviceflow Managements (SfM) in Anlehnung an [KLIWET00], [KLIWETa01] sind:

- Bei SfM steht die Dienstleistung für Kunden im Vordergrund.
- SfM stellt Muster für die Prozesslogik bereit, aber die „Logik“, die den Übergang zwischen den einzelnen Aufgaben in einem Prozess „beherrscht“, ist abhängig von den jeweiligen Bedürfnissen und dem Verhalten des einzelnen Kunden.
- SfM bietet die Möglichkeit einer umfassenden, vielschichtigen Abstimmung zwischen nicht im voraus einzuschränkenden Einzelleistungserbringern, deren Form, Umfang und Zeitpunkt nicht (immer) voraussehbar ist.
- Die situative Anpassung der Leistungserbringung ist zentraler Teil der Wertschöpfung und von daher herausragendes Ziel von SfM.

- Die Entstehung bzw. spezifische Historie jedes individuellen Serviceflows muss für jeweils alle Einzelleistungserbringer, die direkt an der Schnittstelle zum Kunden arbeiten, einsehbar sein.

Dadurch ermöglicht SfM die flexible Planung und Bearbeitung des individuellen Serviceflows. Es unterstützt die etablierten Kooperations- und Koordinationsformen zwischen den jeweiligen Einzelleistungserbringer. Zudem stellt es Mechanismen zur Unterstützung der Abarbeitung von Hintergrundvorgängen, standardisierten routinierten Abläufen, Nebenläufigkeit sowie Kapazitätsberechnungen und Auswertungen bereit.

Softwaretechnisch könnte man Serviceflow als kundenbezogenes „Servicefloat“, welches das gesamte Wissen über den vollständigen Dienstleistungsprozess besitzt, modellieren. Dieses würde dann von Servicepoint zu Servicepoint weitergeleitet und dabei laufend fortgeschrieben. Eine Abweichung von Standard- und Routineabläufen sowie situative Anpassung wird dem Dienstleister in jedem Servicepoint ermöglicht. Darüber hinaus wird dem Dienstleister, zur Befriedigung der Kundenwünsche, nach dem SfM Ansatz die Möglichkeit gegeben, in jedem Servicepoint parallele Prozesse zu starten, die zu einem Zeitpunkt aufeinandertreffen oder enden. Ergebnisse aus diesen Prozessen werden entsprechend aufbereitet und dem Kunden in seinem Servicepoint dargelegt.

Der Servicepoint stellt Dienste und Aufgaben eines Serviceerbringers zur Verfügung und beinhaltet weitere organisationsübergreifende Serviceerbringer-Angaben, die allesamt in einem sogenannten „Servicepointscript“ festgelegt werden. Verarbeitungsschritte und andere Informationen werden im Servicefloat dokumentiert, wie z.B. der aktuelle Servicepoint, der nächste Servicepoint sowie vorige Servicepoints. (Sowohl Servicefloats als auch Servicepointscripts können im softwaretechnischen Sinn als bearbeitende Materialien verstanden werden.)

Servicefloats setzen sich aus folgenden Bestandteilen zusammen (siehe auch [KLISCHE01], [KLIWET02]):

- Eindeutiger Identifikator
- Einem Typ, der die Art des Serviceflows repräsentiert
- Basisinformationen über den Kunden (z.B. Kundennummer, Name, Adresse)
- Basisinformationen über den (oder die) Serviceprovider an den vorgesehenen Servicepoints
- Aktueller Servicepoint
- Liste der nächsten „anzulaufenden“ Servicepoints
- Liste der schon „durchlaufenen“ Servicepoints
- Liste von akkumulierten Nachbedingungen, die von den schon „durchlaufenen“ Servicepoints hinterlassen wurden
- Liste von Dokumenten, die im Laufe des Serviceflows im Rahmen der Leistungserbringung an den Servicepoints produziert wurden

Servicepointscripts setzen sich aus folgenden Elementen zusammen (siehe auch [KLISCHE01], [KLIWET02]):

- Eindeutiger Identifikator
- Einen Namen, der den Servicepoint identifiziert, an dem das Servicepointscript ausgeführt wird
- Einen Typ, der die Art des Servicepointscripts repräsentiert
- Eindeutiger Identifikator des Servicefloats, in dessen Rahmen das Servicepointscript zum Einsatz kommt
- Basisinformationen über den (oder die) Serviceprovider
- Basisinformationen über den Kunden (z.B. Kundennummer)
- Aktuelle Aktivität
- Liste der noch auszuführenden Aktivitäten
- Liste der schon ausgeführten Aktivitäten
- Liste der Vorbedingungen der Menge der Aktivitäten
- Liste der Nachbedingungen der Menge der Aktivitäten
- Liste von Dokumenten, die im Rahmen der Leistungserbringung an dem Servicepoint erstellt werden

Das Leitbild Serviceflow Management bietet somit für den IT-Einsatz eine integrierende Sichtweise der vernetzten Dienstleistungen und stellt die Grundlage für die Verbindung von statischer und dynamischer Modellierung und situativer kundenbezogener Handlungsanpassung dar.

3.2.2 Projektverlauf

Das Projektseminar des Arbeitsbereichs Softwaretechnik des Fachbereichs Informatik der Universität Hamburg fand im Wintersemester 2000/2001 unter dem Lehrveranstaltungstitel: **„***Service als Leitbild: Softwareunterstützung für Dienstleister***“** statt. Lernziel der Lehrveranstaltung war die Bedeutung des Leitbilds „Service“ für die Softwareentwicklung auf den Ebenen Anwendungsorientierung, Systemarchitektur, Basistechnologien, Programmiertechniken und Benutzungsschnittstellen. Weiteres Lernziel war, wie die praktische Unterstützung von Dienstleistungen aussehen kann und wie anhand des Anwendungsbeispiels im Rahmen des Bürgerprozessportals www.hamburg.de die softwaretechnische Umsetzung des Ansatzes Serviceflow Management (SfM) realisiert werden kann. Weiterführende Literatur zu Serviceflow Management findet sich unter: [KLIWET00], [KLWEBA01], [KLIWETb01], [KLIWETa01], [KLISCHE01], [KLIWET02].

Im Projektseminar wurde zunächst die Frage „Was Service ausmacht?“ ausführlich diskutiert. Im Anschluss daran wurde das Leitbild Serviceflow Management besprochen und die Grundannahmen über die mögliche computergestützte Unterstützung aufgestellt. Daraufhin fanden die ersten aufgabenbezogenen Anforderungsermittlungen[20] des Ist-Zustands des Anwendungsbeispiels „elektronische Beantragung von Briefwahlunterlagen“ bei der Stadt Hamburg bzw. Senatsamt für Bezirksangelegenheiten (SfB) der Stadt Hamburg statt.

[20] Siehe Projekthandbuch STEPS in [FLOYD91]

Im nächsten Schritt wurden neue Web-Technologien vorgestellt. Darunter insbesondere der Einsatz von XML, Servlets, JSP und das Entwurfsmetapher der „Fachlichen Services“. Weiterhin erlangten die Lehrveranstaltungsteilnehmer erste Einblicke in die Web-Content-Management-Systeme (WCMS) „Vignette StoryServer 5“ und „VIP – Gauss Enterprise“, da diese im Anwendungsbeispiel verwendet wurden. Bei der Realisierung des Web-basierten SfM Prototyps wurde von den jeweiligen Prägungen und Besonderheiten eines WCMS in der Umsetzung des Leitbilds Serviceflow Management abstrahiert.

Für das Anwendungsbeispiel wurden persönliche Interviews mit den Verantwortlichen des SfB durchgeführt. Daraus resultierten die ersten Szenarien[21] und Glossare[22] der Anwendungswelt. Aus den Ergebnissen wurden Umgangsformen identifiziert, die in dem Soll-Zustand der Anwendungswelt modelliert wurden. Aus der Rückkopplung mit den Verantwortlichen des SfB wurden Unzulänglichkeiten des Soll-Zustandes erörtert und entsprechend verbessert oder ergänzt. Aus dem Soll-Zustand und auf Grundlage des Leitbilds Serviceflow Management resultierte die erste Systemvision.

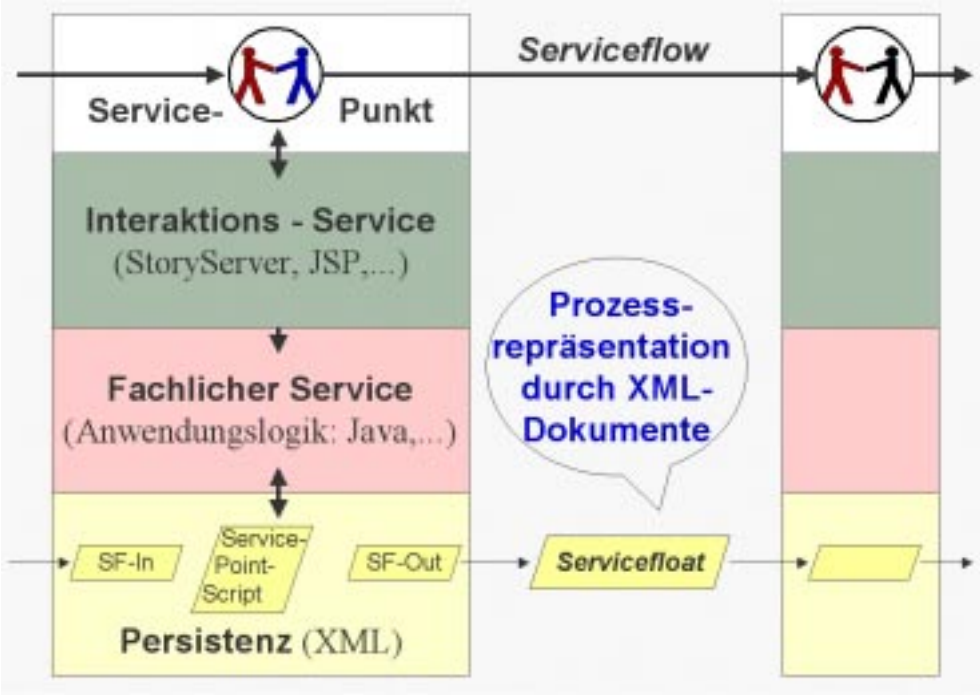

Abbildung 12 - Umsetzung des Web-basierten „Serviceflow Management“ Prototyps

Die Abbildung 12 zeigt das Schichtenmodell der Web-basierten Systemvision des Serviceflow Managements und hat folgende Schichten bzw. Aufgaben:

[21] Szenarien beschreiben Arbeitskontexte des Anwendungsfelds, Arbeitsabläufe in Form von Handlungstudien sowie vorhandene Problemlösungen. Sie sind beispielhaft und episodisch. [ZÜL98]
[22] Glossare beschreiben bereits definierte Begriffe der Anwendung, technisch rekonstruierte Begriffe der Anwendung und neue Begriffsbildungen. Sie sind an der Fachsprache der Benutzer orientiert. [ZÜL98]

- Servicepoint (Ort der Serviceleistung)
- Interaktions-Service (Kommunikations- und Interaktionsstelle mit dem Dienstleister)
- Fachlicher-Service (Kapselung der Servicepoint- bzw. Anwendungslogik)
- Persistenzmedium (Konservierung des Zustands)

Aus dem Leitbild des Serviceflow Managements, den Anforderungsermittlungsergebnissen, der Systemvision sowie den gewählten Web-Technologien wurden die Schnittstellen zunächst für die Anwendungslogik und danach für die Interaktionslogik und dem zugrundegelegten Persistenzmedium ausgearbeitet. Während dieser Arbeitsphase war für alle Projektteilnehmenden die lose Kopplung zwischen den jeweiligen Schichten von großer Bedeutung. Seiteneffekt dieser Vorgehensweise war die eigenständige Arbeit in Projektgruppen. Darüber hinaus prägte die heterogene und verteilte Umgebung der IT-Infrastruktur des Anwendungsbeispiels die Herangehensweise insofern, dass ein organisationsübergreifender Datenaustausch im Vordergrund stand. Des weiteren sollte noch erwähnt werden, dass zu dem angegebenen Zeitpunkt die Verwendung einer relationalen oder XML-fähigen Datenbank nicht im Gespräch war.

Daher wurde der Einsatz der XML-Technologie zur Realisierung des Serviceflow Managements auf der Persistenzebene gewählt. Folglich wurden nach dem Serviceflow Management Ansatz anhand des Soll-Zustands, nach mehreren zyklischen Revisionen, die *Servicefloat-DTD* und die *Servicepointscript-DTD* modelliert. Diese enthalten alle Informationen über den „Fluss der Dienstleistung“ und ihre konkrete Abarbeitung in einem oder mehreren Servicepoints. Für das Anwendungsbeispiel dienten sie als Vorlage für die Erstellung von XML-Master-Dokumenten wie für die „elektronische Beantragung von Briefwahlunterlagen“.

Kriterien für den Entwurf des Web-basierten Serviceflow Management Prototyps waren:

- Anlehnung an die evolutionäre Vorgehensweise “Werkzeug & Material (WAM)[23] Metapher“
- Orientierung an folgenden softwaretechnischen Kriterien:
  - Austauschbarkeit
  - Wiederverwendung des Designs (Pattern)
  - Unabhängigkeit von WCMS/Web-Frontend
- Portabilität durch Java + XML

Der Entwurf der Fachlichen Service Anwendungslogik basierte stark auf der Wahl des Persistenzmediums XML. In Anbetracht dieser Herangehensweise sind Teile der Anwendungslogik technisch dem Document-Object-Model[24] (DOM) sehr nah. Jedoch

[23] siehe [ZÜLLIG98]

[24] The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page. http://www.w3.org/DOM/

wurde der gesamte Softwareentwurf von uns durch die gewählte Architektur sowie die generisch fachlich motivierten Umgangsformen von dem Einsatz der verschiedenen Persistenzmedien abstrahiert.

Am Ende des Projektseminars wurde der Web-basierte SfM Prototyp eines Web-basierten Servicepoints erfolgreich demonstriert. Zugleich wurde festgestellt, dass der Web-basierte SfM Prototyp und dabei insbesondere die Anwendungslogik einige Schwachstellen aufwies und überarbeitet werden sollte. Im nächsten Abschnitt wird der Web-basierte SfM Prototyp näher beschrieben .

3.2.3 Vorstellung des Web-basierten SfM Prototyps

Zusammen mit den Projektverantwortlichen wurde die Entwicklung des Web-basierten SfM Prototyps nach dem Serviceflow Management Ansatz vorangetrieben. Die grundlegende grobe Architektur des Self-Service Web-basierten SfM Prototyps eines Servicepoints zeigt die Abbildung 13. Die Implementierung der Web-basierten SfM Anwendungslogik wurde von Timmy Blank und mir (dem Autor) realisiert. In Timmy Blanks Diplomarbeit [BLANK01] wird ab Seite 41 detailliert auf die Vorgaben und Vorgehensweise der Implementierung der Anwendungslogik sowie den Web-basierten Prototyp eingegangen. Im folgenden Abschnitt wird der Prototyp und die Anwendungslogik zusammenfassend vorgestellt.

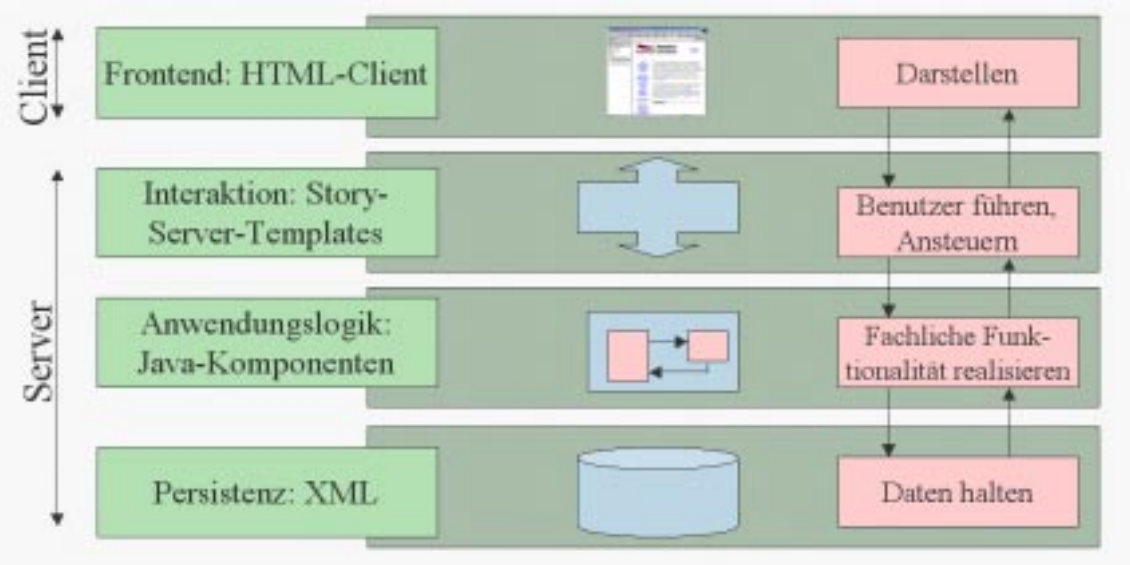

Abbildung 13 - nach Züllighoven: Schema der Web-basierten Servicepoint-Architektur

Als Einstiegs- und Ausstiegspunkt des Self-Service Web-basierten Prototyps eines Web-basierten Servicepoints dient für den Bürger das Web-Frontend. Dadurch interagiert der Bürger mit der server-seitigen Anwendung. Die erste Schicht der Web-basierten server-seitigen Anwendung ist der Interaktionsservice, der in unserem Anwendungsbeispiel mit den vom StoryServer angebotenen Templates realisiert wurde. Der Interaktionsservice

führt die Interaktion mit dem Bürger und zeitgleich werden die Parameter an die Anwendungslogik in der darrunterliegenden Schicht übergeben.

Die Verbindung zwischen den StoryServer Templates (Interaktionsschicht) und der Anwendungslogik kommt über TclBlend[25] zustande. Über diese Schnittstelle wird die Anwendungslogik nach fachlicher Funktionalität angesprochen und liefert entsprechend Daten aus den Servicefloats und Servicepointscripts an die Interaktionsschicht. Ankommende und Abgehende Servicefloats werden dabei auf dem Persistenzmedium File-System als XML-Dateien abgelegt. Die Abbildung 14 gibt eine Übersicht über den gesamten Vorgang.

3.2.3.1 Umgang mit Servicefloats und Servicepointscripts

Die Daten bzw. der Zustand der Servicefloats und Servicepointscripts wird mit Hilfe der XML-Auszeichnungssprache in einzelnen Dateien gespeichert. XML[26] steht für „eXtensible Markup Language". Die wichtigste Zielsetzung von XML (W3C-Konsortium) ist die Trennung von Inhalt und Darstellung in Dokumenten. Ein weiteres Ziel ist die Plattform-, Applikations- und (Programmier-) Sprachenunabhängigkeit. Außerdem ist es in der XML-Auszeichnungssprache möglich eigene Tags zu definieren und deren Semantik selbst zu bestimmen. In unserem Anwendungsbeispiel sind die XML-Dateien (Servicefloats und Servicepointscripts) bzw. die darin enthaltene Struktur der Elemente (Tags) nach den Regeln der DTD-Grammatik in DTDs (Document Type Definition) festgelegt. Dadurch wurde von uns der Typ und die Struktur von XML-Dokumenten bestimmt. Darüber hinaus bietet sich die Definition und die Nutzung von DTDs für die Validierung der Gültigkeit einer XML-Datei.

Der Umgang mit den Servicefloats und den Servicepointscripts basiert auf der XML-Manipulation. In unserer Implementation werden sie technisch auf dem Document Object Model[27] (DOM) abgebildet. Dieses ist das Programmierungs-API für gültige HTML- und wohlgeformte XML-Dokumente. Die logische Struktur des XML-Dokumentes wird in eine baumartige Struktur umgewandelt. Jeder Baum bzw. Teilbaum stellt wiederum Bäume von Objekten dar. DOM stellt zudem noch eine Reihe von verändernden und sondierenden Operationen für die Manipulation dieser Bäume bereit. Die Baumstruktur eines DOM-Objektes besteht aus einem Root-Knoten und einem oder mehreren Child-Knoten. Die Implementation von DOM wurde schon in mehreren Programmiersprachen, wie Java, Perl, PHP, C++, etc., realisiert.

Wie die Abbildung 14 zeigt, werden ankommende Servicefloats in einem Servicepoint vom Servicepointmanager in ein DOM-Objekt umgewandelt. In jedem Servicepoint liegt das Master-Servicepointscript, das die Aufgaben und Dienste eines Servicepoints festlegt.

[25] Tcl Blend ist eine dynamisch ladbare Erweiterung für den Tcl-Interpreter. Der TclBlend führt die dynamische Initialisierung der virtuellen Java Maschine durch und ermöglicht somit den Zugriff auf Java-Objekte und -Klassen. Weiterführende Literatur über Tcl-Java Anbindung unter: http://tcl.activestate.com/software/java/

[26] siehe http://www.w3.org/XML/

[27] aktuelle Version: http://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20020114/DOM3-Core.pdf

Vom Servicepointmanager wird veranlasst, dass für den betreffenden Bürger ein Original des Servicepointscripts erstellt wird. Im Original bzw. Servicepointscript werden alle den Bürger betreffenden Aktivitäten eingetragen. So ist es z.B. in unserem Web-basierten SfM Prototyp des Anwendungsbeispiels möglich in einem Servicepoint neue Aktivitäten hinzuzufügen oder sie zu entfernen oder auch die Reihenfolge der Durchführung dieser zu ändern. Darüber hinaus kann bestimmt werden welcher Servicepoint im Servicefloat als nächstes dran kommt (oder gelöscht werden soll). Dafür werden in dem jeweiligen Servicefloat die vorgegebenen Vor- und Nachbedingungen verifiziert. Des weiteren ist es möglich nach dem Durchführen einer Aktivität diese als abgearbeitet abzulegen oder aber auch die letzte Aktivität zurückzuholen und zu bearbeiten. Um eine Aktivität ausführen zu können, werden zunächst die Vor- und Nachbedingungen einer Aktivität geprüft. Die Vor- und Nachbedingungen für eine Aktivität werden im Servicepointscript festgelegt. Die gesamte Umgangshistorie wird jeweils in Servicefloats bzw. Servicepointscripts dokumentiert.

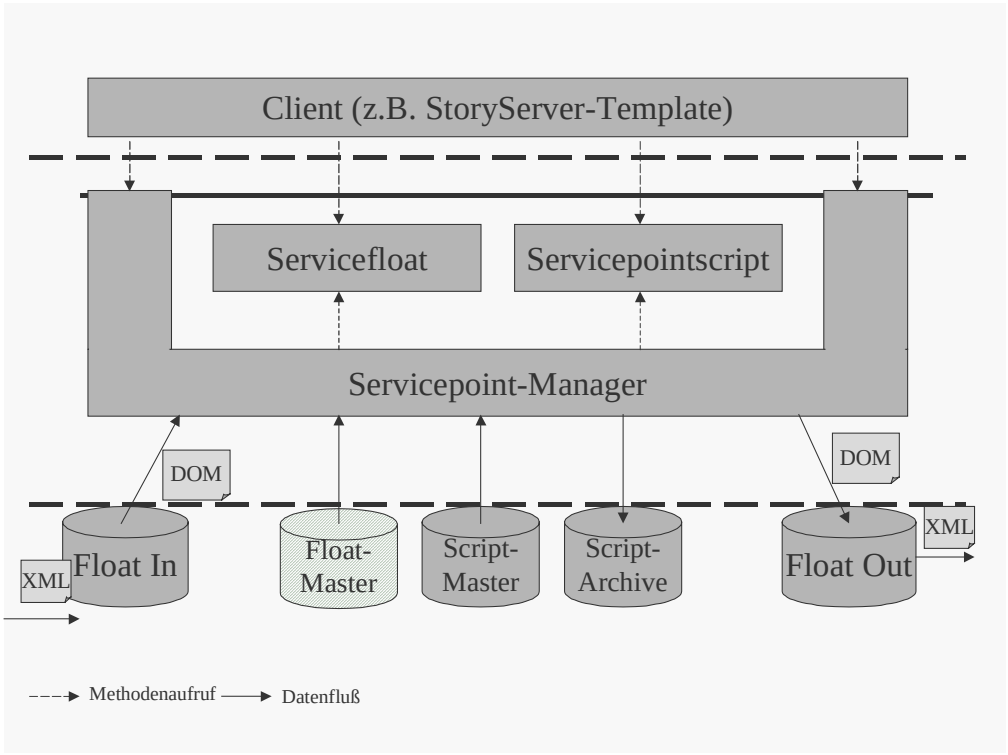

Abbildung 14 - Detailansicht der Web-basierten Servicepoint-Architektur

Weiteres Merkmal des Servicepointmanagers ist die Möglichkeit Dokumente (z.B. Word-Datei, Foto, etc.) an die Servicefloats und Servicepointscripts anzuhängen bzw. sie zu verwalten. Sie werden in Servicefloats entsprechend von einem Servicepoint zum nächsten zusammen mit der Umgangshistorie versandt. Zum Schluss wandelt der Servicepointmanager das DOM-Objekt entsprechend wieder zu XML-Dateien um und macht diese versandfertig. Der Servicepointmanager stellt hierfür die erforderlichen Dienste zum Versenden und Empfangen zur Verfügung. In unserem Anwendungsbeispiel werden die XML-Dateien vom Servicepointmanager an einem FTP-Account versandt

und empfangen. Die bearbeiteten oder fertigen Servicepointscripts werden in den jeweiligen Servicepoints archiviert. So bleibt die vollständige Umgangs- und Dokumenthistorie in jedem Servicepoint für jeden Beteiligten transparent.

3.2.3.2 Anwendungslogik: Fachliche Services

Das Konzept der Fachlichen Services basiert auf der Kapselung des fachlichen Wissens in unabhängigen Einheiten, die handhabungs- und präsentationsunabhängig sind (siehe [OTTSCH00] S. 67). Fachliche Services werden in zusammengehörige Dienstleitungen gebündelt, deren Benutzungsschnittstelle unabhängig von dem Anwendungskontext, wie z.B. Internet, Desktop, etc. ist.

> *„Fachliche Services sind fachliche und technische Einheiten eines großen verteilten Anwendungssystems. Sie vergegenständlichen Geschäftslogik derart, dass nachvollziehbare und in einem Anwendungskontext zusammengehörige Leistungen gekapselt werden. Fachliche Services sind so realisiert, dass die konkrete Art und Weise offen bleibt, wie diese Leistungen erbracht und wie sie dem Anwender an der Benutzungsschnittstelle präsentiert werden.“* [WOLF01]

Fachliche Services besitzen nach [WOLF00] folgende Eigenschaften:

- kapseln und vergegenständlichen fachliches Wissen
- implementieren Anwendungslogik sowie handhabungs- und präsentationsunabhängige Inhalte
- bieten evtl. dynamische Inhalte unter statischer Service-Schnittstelle an
- Service-Schnittstelle kann in mehrere fachliche oder technische Aspekte unterteilt sein
- können mit anderen Fachlichen oder Technischen Services zusammenarbeiten (z.B. in hierarchischer Beziehung)
- werden entworfen in Autor-Kritiker-Zyklen im Zusammenspiel mit den Anbindungen der Benutzerschnittstellen

Das Kapseln der Anwendungslogik in Fachlichen Services ermöglicht eine Handhabungs- und Präsentationsunabhängigkeit von benutzenden Interaktions-Komponenten und dem darrunterliegendem Persistenzmedium. Dadurch eignet sich deren Realisierung besonderes gut als Middelware in heterogenen Umgebungen und in n-Tier-Architekturen.

In vielerlei Hinsicht ist das technische Konzept der Fachlichen Services dem der EJB ähnlich. Beide präsentieren die Middelware einer Anwendung und kapseln die Anwendungslogik. Dadurch sind beide handhabungs- und präsentationsunabhängig. Besonderes der Einsatz in verteilten Umgebungen mit Mehrbenutzerfähigkeit verbindet beide Konzepte. Im Vergleich mit den Anforderungen an Fachliche Services sind mehrere softwaretechnische Entwurfsprinzipien und Qualitätskriterien wie z.B.

Kapselung, Verteilung, Wiederverwendbarkeit, Portabilität und Austauschbarkeit in EJB wieder zu finden.

Ideale technische Eigenschaften der Fachlichen Services Schnittstelle nach [WOLF00]:

- Methoden werden als atomar angesehen, in diesem Sinne ist der Service zustandslos.
- Deshalb müssen alle für eine Methode implizit oder explizit notwendigen Parameter jedes Mal übergeben werden.
- Ein Fachlicher Service kann Nachrichten verschicken, deshalb ist eine Registrierung der Nachrichten möglich. In diesem Sinne ist der Fachliche Service nicht zustandslos.
- Materialien der Anwendung werden über prozessübergreifende gültige Identifikatoren benannt.

In der Anwendungslogik des Web-basierten SfM Prototyps eines Web-basierten Servicepoints ist der Servicepointmanager angesiedelt, der die zentrale Einheit eines jeden Servicepoints darstellt. Die Anwendungslogik stellt sozusagen zusammen mit dem Servicefloat und Servicepointscript in unserem Web-basierten SfM Prototyp einen Fachlichen Service dar. Die Anwendungslogik des Fachlichen Service wurde in Java Programmiersprache implementiert. Dazu wurde für die Manipulation der XML-Dokumente auf der Persistenzebene die Xerces Java-XML-Parser[28] Bibliothek, die technisch auf die DOM-API abbildet, verwendet. Das File-System des Web-Servers diente als Persistenzmedium für die XML-Dokumente. Um den Softwareentwurf der Anwendungslogik einfach und übersichtlich zu halten, haben wir uns entschlossen den Servicepointmanager nach dem Singleton Entwurfsmuster (vgl. [GHJV96] S. 139) zu realisieren. Als Zentrale Einheit eines Servicepoints stellt der Servicepointmanager folgende Dienste bereit:

- Verwalten von Servicefloats und Servicepointscripts
- Erzeugen der notwendigen Mustervorlagen (DTDs)
- Erzeugen neuer Servicefloats und Servicepointscripts (XML-Master-Dokumente) aus Mustervorlagen
- Empfang und Versand von Servicefloats
- Archivierung abgearbeiteter Servicepointscripts

Der Servicepointmanager als zentrale Einheit bietet somit sondierende und verändernde Umgangsformen an. Er übernimmt für den Bürger als eine Art „Black-box“ über die Anwendungslogik verwaltende Aufgaben auf der Persistenzebene. Der Servicepointmanager stellt für die anfallenden Aufgaben wie Materialien verwalten, versorgen und poolen geeignete Mechanismen, die auf dem Entwurfsmuster der Materialverwaltung (vgl. [ZÜLLIG98], S. 299) konstruiert sind, zur Verfügung. Weiterhin muss der Servicepointmanager an jedem Servicepoint dafür sorgen, dass unterschiedliche Mustervorlagen zur Erzeugung der Servicepointscripts und Servicefloats

[28] siehe http://xml.apache.org/xerces-j/index.html

geeignet verwaltet werden. Da diese Verwaltung von Mustervorlagen außerhalb des Materialmanagers stattfindet, ist sie nach dem Entwurfsmuster von Repertoires (vgl. [GRYCZ96] S. 191) realisiert worden. Des weiteren ist für das Empfangen und Versenden von Servicefloats zwischen den einzelnen Servicepoints das Konzept der Postkörbe (vgl. [ZÜLLIG98], S. 440) umgesetzt worden. So wenden sich die Servicefloats (Arbeitsmaterial) an bestimmte zuständige Komponenten des Servicepointmanagers, die das Versenden und Empfangen durchführen. Darüber hinaus bietet der Servicepointmanager ein Archiv für abgearbeitete Servicepointscripts.

Der Servicepointmanager wird über die neu an einem Servicepoint angekommenen Servicefloats anhand des Benachrichtigungsmechanismus des Beobachter-Entwurfsmusters (vgl. [GRYCZ96] S. 257) informiert. So führt eine Änderung des Servicepointzustands dazu, dass alle abhängigen Objekte benachrichtigt und aktualisiert werden.

3.2.3.3 Interaktionsservice

Der Interaktionsservice wurde in unserem Anwendungsbeispiel mit Hilfe von StoryServer Templates als Web-Frontend realisiert. Das Web-Frontend bzw. Templates kommunizieren über TclBlend mit dem Servicepointmanager. Das Web-Frontend steuert den Bürger durch die ganze Anwendung. Es besteht aus mehreren HTML-Eingabe-Formularen (s. Abbildung 15), Status-Informations-Seiten und einer Danke-/Ausblick-Seite.

Abbildung 15 - Web-Frontend des SfM Prototyps

Mit der Eingabe des Vor- und Nachnamens initialisiert das Web-Frontend für diesen Bürger über den Servicepointmanager die Anwendungslogik. Der Servicepointmanager schaut zunächst, ob es für den Bürger für diesen Vorgang im Servicepoint schon einen Servicefloat und ein korrespondierendes Servicepointscript gibt. Falls ja, dann holt der Servicepointmanager aus dem Archiv das dazugehörige Servicefloat und das Servicepointscript. Hinterher bietet der Servicepointmanager dem Bürger an, den Vorgang dort weiterzuführen, wo er zuletzt war. Führt der Bürger zum ersten Mal diesen Vorgang aus, dann erzeugt der Servicepointmanager ein neues Servicefloat und für diesen Servicepoint ein korrespondierendes Servicepointscript. Jeder vom Bürger ausgeführte Zwischenschritt wird über die URL-Paramenter an das nächste Template übergeben.

Die Anwendungslogik speichert unmittelbar jeden einzelnen Zwischenschritt auf dem Persistenzmedium und steuert zeitgleich die Web-Interaktion weiter. Die Interaktion endet mit der Aufforderung „Antrag senden“. Hierzu liest der Servicepointmanager die Informationen über den nächsten Servicepoint aus dem Servicefloat und bereitet den Versand vor. Handelt es sich um die letzte Servicefloat-Station, so wird das Servicefloat und das Servicepointscript persistent auf dem File-System archiviert. Andernfalls schickt der Servicepointmanager das Servicefloat an den nächsten vorgesehenen Servicepoint weiter.

3.2.4 Schwachstellenanalyse

In diesem Abschnitt wird der Web-basierte SfM Prototyp eines Web-basierten Servicepoints und die Anwendungslogik nach den in Kapitel 2.1 genannten softwaretechnischen Qualitätskriterien untersucht:

Skalierbarkeit

Der Servicepointmanager wurde nach dem Singleton Entwurfsmuster (vgl. [GHJV96] S. 139) realisiert. In unserem Anwendungsbeispiel wurde WCMS Vignette StoryServer 5 eingesetzt. Dabei wurde die Verbindung zwischen der Anwendungslogik und dem StoryServer mittels TclBlend umgesetzt. Im nachhinein stellte sich heraus, dass die von uns implementierte Anwendungslogik, konkret der Servicepointmanager, in dieser Infrastruktur-Konstellation einige Beschränkungen und Schwachstellen mit sich bringt.

Beim StoryServer ist es grundsätzlich möglich, mehrere sogenannte Delivery-Server (siehe [VIGN99]) einzusetzen, welche die Seitengenerierung und -auslieferung übernehmen. Somit ist jeder Delivery-Server eigenständig in der Lage die Seitengenerierung vorzunehmen. Weiterhin können die Delivery-Server auf mehreren verschiedenen Rechnern (Cluster) betrieben werden. Auf jedem einzelnen läuft dann entsprechend eine eigene Java Virtual Machine, der wiederum ein Tcl-Interpreter zugeordnet ist. Diese Maßnahme des WCMS Vignette StoryServer Herstellers basiert auf dem Hintergrund, dass auch größere Anfragevolumen bearbeitet werden können. Die Implementierung des Servicepointmanagers nach dem Singleton-Entwurfsmuster hat für unseren Web-basierten SfM Prototyp jedoch die Auswirkung, dass es auf jedem

(verteilten) Rechner viele verschiedene Exemplare des Servicepointmanagers geben würde. Dies würde dazu führen, dass mehrere Servicepointmanager laufen, die kein Wissen über die Existenz von den anderen hätten. Letztlich würden verschiedene Servicepointmanager mehrere Exemplare von Servicefloats bzw. Servicepointscripts für einen Kunden bereitstellen.

Da die Entwicklungsumgebung nur mit einem Vignette StoryServer Delivery-Server betrieben wurde, hat sich diese Schwachstelle zunächst nicht bemerkbar gemacht. Darüber hinaus ist die Schwachstelle bis zum Ende des Projektseminars unbemerkt geblieben, weil nur ein einziges Template mit dem Servicepointmanager interagiert hatte, so dass die Reaktion auf die gleichzeitige Benutzung des Servicepointmanagers durch mehrere Templates nicht getestet werden konnte. Das Singleton-Entwurfsmuster stellt somit eine deutliche Schwachstelle bei der Realisierung des Servicepointmanagers in dieser Infrastruktur-Konstellation dar.

Robustheit

Bei unserem Web-basierten SfM Prototyp wurde die Gefahr inkonsistenter Zustände, die besonders in Web-basierten Anwendungen durch Verbindungsabbrüche oder auch konkurrierende Zugriffe verursacht werden können, in unserem Anwendungsbeispiel auf Servicefloats und Servicepointscripts, vorkommen, nicht beachtet,. Insbesondere in kritischen Weiterschaltungen bzw. Übergaben der Zuständigkeiten und Daten von einem Servicefloat an ein Servicepointscript können solche Ausgangslagen bedeutende Seiteneffekte verursachen. Hinzu kommt noch, dass der Servicepointmanager alle Dokumente (u.a. auch Servicefloats und Servicepointscripts) im Original verarbeitet, so dass beim Auftreten eines Fehlers, nicht nur die Daten unbrauchbar gemacht werden, sondern die ganze Anwendungslogik in inkonsistenten Zustand gebracht werden würde. Das kann sowohl in der Kommunikation (s. Abbildung 9) zwischen der Anwendungslogik mit dem Interaktionsservice als auch mit dem Persistenzmedium auftreten. Diese Schwachstelle wird auf der Persistenzebene zusätzlich durch den Umstand verstärkt, dass die Xerces-DOM-Implementationen nicht „Thread-Safe“ (vgl. [McLAU00] S. 47) ist.

Portabilität

Durch die Verwendung der Programmiersprache Java und XML auf der Persistenzebene wurde dieses softwaretechnische Qualitätskriterium in unserem Web-basierten SfM Prototyp größtenteils erfüllt. Dennoch weist der Prototyp einige wenige Schwachstellen bezüglich der Portabilität auf:

- Festlegung der Datei- und Ordner-Pfade für Servicefloats und Servicepointscripts für XML-Master, -Out, -In, -Archive (s. Abbildung 10) im Java-Code.
- Keine zentrale oder portable Verwaltung von Benutzerdaten. Dadurch ist die Zuordnung von Kunde – Servicefloat – Servicepointscript in verschiedenen Servicepoints nicht machbar. Da wir in unserem Anwendungsbeispiel nur einen Servicepoint modelliert haben, stand die Realisierung der Benutzerverwaltung nicht

im Vordergrund. Außerdem mussten wir uns aus zeitlichen Gründen auf die wesentlichen Funktionalitäten des Web-basierten SfM Prototyps konzentrieren.

Sicherheit

Sicherheitsaspekte sind in der Modellierung und Implementierung des Web-basierten SfM Prototyps kaum berücksichtigt worden:

- Keine Gewährleistung einer Authentifizierung und Zugriffskontrolle (authorization and access control) in dem Interaktionsservice
- Keine Sicherung der Authentifizierung über Web-Frontend
- Wahrung der Integrität des gesamten SfM Prototyps nicht vorhanden
- Keine Möglichkeit einen Nachweis über korrekt ausgeführte Kommunikation aufzubringen
- Keine Mechanismen für die Sicherung einer korrekten und vollständigen Verfügbarkeit
- In der Kommunikation zwischen dem Web-Frontend (Interaktionsservice) und der Anwendungslogik gibt es keine Schutzmechanismen im Umgang mit den Original-Dokumenten.
- Versenden von Servicefloats von einem Servicepoint zu den anderen wird unverschlüsselt gemacht
- Öffentlicher Zugang auf das File-System bzw. den Ordner, auf dem die Servicefloats und Servicepointscripts, XML-Master, -In, -Out oder -Archive auf dem Multi-User verwendeten Web-Server liegen. Kein Schutzmechanismus implementiert

Wiederverwendung

Im gesamten Modellierungs- und Entwicklungsprozess war eine der wichtigsten Designentscheidungen die Wiederverwendung (siehe Kap. 3.2.2). Dies erstreckte sich besonders auf zwei Ebenen: Architektur des Web-basierten SfM Prototyps und auf die einzelnen Komponenten durch die Verwendung von Entwurfsmustern (vgl. [BLANK01] S. 95). Beide Ebenen haben dieses Qualitätskriterium sowohl durch die grobe als auch feine Entwurfsstrukturierung erfüllt. Erst wurde das schichtenbasierte Architekturmuster, das die jeweils einzelnen Schichtenaufgaben und -zuständigkeiten weitgehend kapselt (s. Abbildung 13) und umsetzt, angewandt. Die Prägungen und Besonderheiten der einzelnen Schichten sind durch die klar definierten und getrennten Schnittstellen abstrahiert worden. Zudem ist aus dem Entwurfsmuster der den Fachlichen Services zugewiesenen Aufgaben in hohem Maße die Wiederverwendung verwirklicht worden. Weiterhin wurde durch die Verwendung von Entwurfsmustern in den einzelnen Schichten z.B. in dem Servicepointmanager die Architektur des SfM Prototyps in dem vorgegebenen Zeit- und Projektrahmen sorgfältig ausgearbeitet. Darüber hinaus sind durch die minimierte Kopplung der einzelnen Komponenten in den Schichten keine zyklischen Abhängigkeitsbeziehungen entstanden (außer dem gewollten Einsatz des Beobachtermusters im Servicepointmanager).

Integration

Durch die Verwendung der plattform- und systemunabhängigen Komponenten zur Realisierung des Web-basierten SfM Prototyps, hier im einzelnen:

- HTML für Web-Frontend
- Interaktionsservice; setzt voraus, dass eine geeignete Möglichkeit eine Java-Verbindung herzustellen, die die Java-Komponente ausführen wird, besteht.
- Anwendungslogik; setzt eine Java-Umgebung sowie XML als Persistenzmedium voraus

ist die Integration in allerlei Systemen leicht möglich. Vor allem die Handhabungsunabhängigkeit der einzelnen Schichten des Web-basierten SfM Prototyps, die auf dem Entwurfsmuster der Fachlichen Services gebaut worden sind, ist ausschlaggebend für die Integration in der Infrastruktur-Konstellation, wie in dem Anwendungsbeispiel.

Aus diesen Gründen setzt der Einsatz des Gesamtpakets des Web-basierten SfM Prototyps allerdings einige Re-Implementierungen der einzelnen Schichten und Komponenten voraus.

3.2.5 Zusammenfassung

In diesem Kapitel wurde der Web-basierte SfM Prototyp eines Web-basierten Servicepoints vorgestellt, der auf Basis der Technologien Java, XML, StoryServer 5.0 und TclBlend realisiert wurde. Als erstes wurde der Projektverlauf und die Vorgehensweise in der Erstellung des Web-basierten SfM Prototyps beschrieben. Welche softwaretechnischen Kriterien für die Erstellung des Prototyps zu dem Zeitpunkt wichtig waren, wurde aufgeführt. Im Anschluss daran wurde das Leitbild des Serviceflow Managements, das die theoretische Grundlage des Web-basierten SfM Prototyps bildet, erläutert.

Anhand der Architekturübersicht des Web-basierten SfM Prototyps ist verdeutlicht worden, welche Technologien auf welcher Ebene eingesetzt wurden und welche Aufgaben den einzelnen Komponenten zukommen. Der StoryServer ist für die Realisierung des Interaktionsservice zuständig. Java wird für die Umsetzung der Anwendungslogik eingesetzt und XML ist das Persistenzmedium. Als Kommunikations-Brücke zwischen StoryServer-Templates und der Java-Komponente wurde TclBlend verwendet.

Die konkrete technische Realisierung des Web-basierten SfM Prototyps, insbesondere der Umgang mit Servicefloats und Servicepointscripts, die Anwendungslogik (insbesondere Servicepointmanager) und der Interaktionsservice wurden anhand des Fallbeispiels erläutert. Der Servicepointmanager als zentrale Einheit eines Servicepoints mit den dazugehörigen Aufgaben (Verwaltung von Servicefloats und Servicepointscripts, Erzeugen der notwendigen Mustervorlangen (DTDs), Erzeugen neuer Servicefloats und

Servicepointscripts aus Mustervorlagen, Empfang und Versand von Servicefloats und Archivierung abgearbeiteter Servicepointscripts) wurde beschrieben. Durch die Anlehnung an die Entwurfsmetapher der Fachlichen Services sind einzelne Schichten des Web-basierten SfM Prototyps erstellt worden. Die Verwendung von einigen Entwurfsmustern haben die Implementierung der internen Architektur des Servicepointmanagers geprägt. Der Servicepointmanager selbst wurde nach dem Singleton-Entwurfsmuster realisiert. Danach ist auf die Umsetzung des Interaktionsservices eingegangen worden. Im Anschluss daran wurde die Benutzungsoberfläche des Self-Servicepoints mit Hilfe des StoryServers vorgestellt.

Zum Schluss wurde der Web-basierte SfM Prototyp und die Anwendungslogik nach softwaretechnischen Qualitätskriterien hinsichtlich der Skalierbarkeit, Robustheit, Portabilität, Sicherheit, Wiederverwendung und Integration untersucht. Das Ergebnis dieser Schwachstellenuntersuchung für den Einsatz des Gesamtpakets des Web-basierten SfM Prototyps war Anlass für eine Re-Implementierung der einzelnen Schichten und Komponenten. Tabelle 7 gibt eine Übersicht über die Ergebnisse der Schwachstellenuntersuchung.

Schwachstellen	***Einschränkungen durch***
Skalierbarkeit	Singleton Entwurfsmuster im Servicepointmanager
Robustheit	Umgang mit Original-Anwendungsmaterialien sowie hohe Gefahr inkonsistenter Zustände
Portabilität	Festlegung der Pfade im Java-Code sowie fehlende zentrale oder portable Verwaltung von Benutzerdaten
Sicherheit	Keine Gewährleistung von Authentifizierung sowie Zugriffskontrolle
Wiederverwendung	Keine Einschränkungen; Verwendung von Architektur- und Entwurfsmustern
Integration	Wenig Einschränkungen; Abhängig von der Hard- und Software Infrastruktur-Konstellation

Tabelle 7 - Übersicht der Schwachstellenanalyse

Im nächsten Abschnitt wird die neue Architektur des Web-basierten SfM Prototyps nach der EJB-Vorgehensweise vorgestellt und untersucht.

3.3 Architektur der mit EJB realisierten SfM Anwendungslogik

In diesem Kapitel wird die Architektur des mit EJB realisierten Web-basierten SfM Prototyps eines Web-basierten Servicepoints mit Schwerpunkt auf der Anwendungslogik demonstriert. Zuerst wird im Kapitel 3.3.1 die Vorgehensweise in der Re-Implementierung der SfM Anwendungslogik mit EJB dargelegt. Eine Optimierung der jetzigen Anwendungslogik mit EJB wird im Kapitel 3.3.2 vorgeschlagen und implementiert. Die neue Architektur des Web-basierten SfM Prototyps wird danach beschrieben und besprochen. Eine Bewertung der neuen Architektur nach softwaretechnischen Entwurfsprinzipien und Qualitätskriterien wird im Kapitel 3.3.3 vorgenommen. Im Anschluss daran wird außerdem eine Eignung von EJB für den Anwendungskontext eGovernement-Bürgerprozessportal durchgeführt.

3.3.1 Vorgehensweise in der Re-Implementierung der SfM Anwendungslogik mit EJB

Dieser Abschnitt gibt einen kurzen Überblick über meine Vorgehensweise bei der Re-Implementierung der SfM Anwendungslogik mit EJB sowie den letzten Stand der Implementation. In Kapitel 3.3.2 und Kapitel 3.3.3 wird auf die Implementation der einzelnen Komponenten der Anwendungslogik näher eingegangen. In der Re-Implementierung der SfM Anwendungslogik mit EJB wurde der technischen Reife des EJB Web-basierten SfM Prototyps meinerseits eine niedrigere Priorität beigemessen. Der Schwerpunkt lag auf der Reife und dem Einsatz der EJB-Technologie in der Entwicklung Web-basierter Anwendungen.

Die Re-Implementierung der SfM Anwendungslogik wurde nach der EJB-Vorgehensweise durchgeführt. In Anlehnung an die EJB-Vorgehensweise und an die Fachliche Services Entwurfsmetapher entstand die Unterteilung der Anwendungslogik des neuen Web-basierten SfM Prototyps in die Fachlichen Services und Integrationsservice nach Abbildung 16.

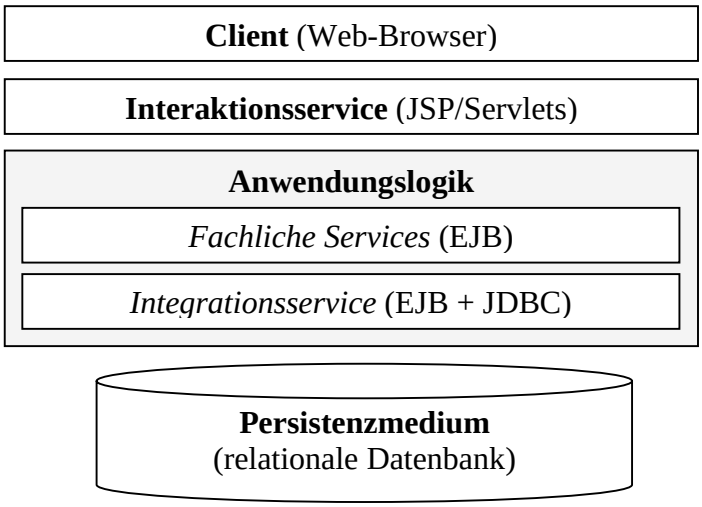

Abbildung 16 - EJB Web-basierter SfM Prototyp: Architektur-Übersicht

Der erste Schritt in der Re-Implementierung der SfM Anwendungslogik mit EJB war die Identifizierung der Anwendungs- und Hilfsmaterialien. Die Ergebnisse der Identifizierung bzw. Anforderungsermittlung flossen in die Definition und Modellierung der Materialien und Attribute in einer Tabellenstruktur der relationalen Datenbank (s. Anhang 6.3) auf der Persistenzebene. Zeitgleich wurden die Materialien in Normalformen überführt. Dadurch wurden die komplexen Beziehungen (Tabellen) durch Aufteilung der Attribute von einer Tabelle auf mehrere Tabellen in einfache Beziehungen umgewandelt.

Als nächstes wurden im Integrationsservice der Anwendungslogik die Entity Beans (s. Kap. 2.2.4.2), die die objektorientierte Sicht der modellierten Anwendungs- und Hilfsmaterialien darstellen, für jedes einzelne Material implementiert. Durch die EJB-Vorgehensweise wurden für die jeweiligen Anwendungsmaterialien die Remote und Remote home Schnittstellen (s. Kap. 2.2.2.3) und bei den Hilfsmaterialien nur die Local und Local home Schnittstellen erstellt (s. Anhang 6.2). Entsprechend wurden im Deployment descriptor die Laufzeiteinstellungen der einzelnen in Entity Beans modellierten Materialien festgelegt.

In den Fachlichen Services der Anwendungslogik wurden fachlich motivierte Dienstleistungen basierend auf der Entwurfsmetapher der Fachlichen Services ausgearbeitet und spezifiziert. Der Softwareentwurf des EJB Web-basierten SfM Prototyps enthielt als zentrale Einheit eines jeden Web-basierten Servicepoints den Servicepointmanager sowie einzelne zustandslose Fachliche Services. Implementiert sind der Servicepointmanager und die zustandslosen Fachlichen Services für Front-Office- und Back-Office-Vorgänge. Eine Funktions- und Leistungsüberprüfung dieser Komponenten wurde vorgenommen. Die einzelnen Sicherheits- und Zugriffseinstellungen der Anwendungslogik-Schicht wurden im nächsten Schritt festgelegt. Welche Rollen auf welche EJBeans (und deren einzelne Methoden) zugreifen dürfen, wurde im Folgenden definiert.

Im Interaktionsservice ist ein Thinner Client mit Servlets/JSP realisiert worden. Der Einfachheit halber wurde das HTML-Design der Web-Frontend Oberfläche schlicht gehalten. Dabei ist nur die Ablauflogik bzw. die Interaktion mit der Anwendungslogik in Servlets/JSP implementiert worden. Dem Interaktionsservice wurde meinerseits in der Re-Implementierung des Web-basierten SfM Prototyps eine niedrigere Priorität zugeschrieben. Im Vordergrund stand die Tauglichkeit des EJB-Komponentenmodells in der Entwicklung Web-basierter Anwendungen, insbesondere in der Anwendungslogik.

Wie schon am Anfang dieses Kapitels gesagt wurde, ist der EJB Web-basierte SfM Prototyp properitär implementiert worden. Die vorhandene implementierte Funktionalität der Anwendungslogik für die Bewertung des Web-basierten SfM Prototyps sowie der EJB-Technologie war nichtsdestotrotz ausreichend. Dennoch wurde für eine umfassendere Bewertung der EJB zur Entwicklung Web-basierter Anwendungen weiterführende Literatur aus [ADAT01], [MONSON01], [ROMAN02] und [SUN01] verwendet. Außerdem wurden aus diversen Erfahrungsberichten, insbesondere aus denen des J2EE Community Portals http://www.theserverside.com, Rückschlüsse gezogen.

3.3.2 Optimierung des Web-basierten SfM Prototyps mit EJB

Hinsichtlich der in Kapitel 3.2.4 aufgeführten Schwachstellen des Web-basierten SfM Prototyps eines Web-basierten Servicepoints werden in diesem Abschnitt durch den Einsatz von EJB Verbesserungs- und Optimierungsvorschläge umgesetzt. Der wesentliche Unterschied zum alten Web-basierten SfM Prototyp besteht beim Einsatz des server-seitigen Komponentenmodells der Enterprise JavaBeans Technologie (Kap. 2.2) in der Anwendungslogik sowie im Integrationsservice, gewöhnlich Teil der EJB. Im Interaktionsservice wurden die Web-Komponenten Java Servlets (Kap. 2.3.1) und JSP (Kap. 2.3.2) verwendet. Außerdem wurde auf der Persistenzebene eine relationale Datenbank eingesetzt. Für den neuen Prototyp wurden auf der Persistenzebene des weiteren folgende Anwendungsmaterialien: Servicefloat, Servicepointscript, Customer, Editor, Provider, Address und Document sowie zwei Hilfsmaterialien: XML-In und XML-Out entworfen und modelliert. Im Anschluss wird die neue Architektur (s. Abbildung 16) präsentiert, gefolgt von einzelnen Details der jeweiligen Komponenten.

Der Web-Browser soll weiterhin als universales Frontend verwendet werden. Damit soll die einfache Nutzung des Web-Browsers (mit all den Nachteilen und Gefahren des HTTP-Protokolls) und die unkomplizierte Benutzerführung erhalten bleiben. Eine wichtige Bedeutung zur Nutzung des Web-Browsers im Bereich der Web-basierten Anwendungen, insbesondere bei Bürgerprozessportalen, ist die ständige Verfügbarkeit der Anwendung über das Internet und das nicht Beschäftigen mit der Installation/Deinstallation oder dem Aktualisieren der Anwendung.

Der Interaktionsservice ist auf Java Servlets und Java Server Pages (JSP) realisiert worden, weil sie problemlos und ohne eine vermittelnde Middelware durch die Java-Konstrukte auf die Fachlichen Services der Anwendungslogik zugreifen kann. Darüber hinaus ermöglicht der Einsatz dieser Technologien die Benutzung des Session-Konzepts, das die Bürgerzuordnung während einer Sitzung in Verbindung mit den EJB-Konstrukten der Anwendungslogik enorm erleichtert. Auch bezüglich softwaretechnischer Qualitätskriterien, wie Skalierbarkeit, Robustheit, Portabilität, Sicherheit, Wiederverwendung und Integration unterstützt der Einsatz von Servlets und JSP in der Web-basierten Anwendungsentwicklung eine zügige Erstellung des Interaktionsservice.

Wie schon in Kapitel 2.2 erwähnt, stellt EJB das server-seitige Komponentenmodell für die Realisierung der unternehmenskritischen Anwendungslogik zur Verfügung. In Abhängigkeit des Anwendungskontexts wird der Anwendungslogikträger bzw. einzelne Fachliche Services üblicherweise in Session Beans (siehe Kap. 2.2.4.1) oder Message-Driven Beans (siehe Kap. 2.2.4.3) realisiert. Dabei stützt sich die Architektur des neuen Web-basierten SfM Prototyps auf weitere Dienste des EJB-Containers (siehe Kapitel 2.2.2.2), wie Transaktionsmanagement, Mail-Service, XML-Processing, etc.

Der Integrationsservice kann als Bestandteil der Anwendungslogik-Architektur betrachtet werden, da dieser im Wesentlichen die Anwendungsdaten mit Hilfe der EJB-Konstrukte und EJB-Container-Dienste auf dem Persistenzmedium abbildet. Die Anwendungsdaten werden in Entity-Beans (siehe Kap. 2.2.4.2) technisch implementiert. Die

Anwendungsdaten werden durch das Abstract Persistence Schema (siehe Kap. 2.2.6) und Java Database Connectivity (JDBC) in der relationalen Datenbank gespeichert. Diese Vorgehensweise erlaubt einer Web-basierten Anwendung von der jeweils eingesetzten relationalen Datenbank zu abstrahieren und somit den Integrationsservice in jeder J2EE Umgebung einzusetzen.

Ein weiterer Optimierungsvorschlag ist die Nutzung einer relationalen Datenbank auf der Persistenzebene[29]. Der Einsatz von Datenbanken in Web-basierten Anwendungen, besonders in Bürgerprozessportalen, bietet sich vor allem wegen des großen Datenvolumens und dem einfachen Umgang mit den Daten an. So wird der Umgang mit den Daten durch die impliziten Datenbank-Funktionalitäten hinsichtlich der Manipulation, wie Komposition, Schnittmenge, Selektion, etc. deutlich erleichtert. Weitere Vorteile des Datenbankeinsatzes sind Skalierbarkeit und Robustheit sowie das vom Datenbank-Management bereitgestellte Sicherheitskonzept bezüglich der Benutzer- und Zugriffsrechte. Einen signifikanten Vorteil der EJB stellt die Einfachheit der Verbindung der EJB Query Language (Kap. 2.2.6) mit der relationalen Datenbank dar. Ein weiterer Aspekt in der Optimierung der Entwicklung des Web-basierten SfM Prototyps, hier am Fallbeispiel eines Bürgerprozessportals, ist das Management der Beteiligten in dem Entwicklungsprozess. Hierfür stellt die EJB-Architektur die sogenannten EJB-Rollen (siehe Kap. 2.2.3) bereit, die eine unterstützende Sichtweise für den Entwicklungsprozess anbieten.

3.3.2.1 Die Anwendungslogik-Architektur

In diesem Abschnitt wird ein allgemeiner Überblick über die Anwendungslogik-Architektur des neuen Web-basierten SfM Prototyps gegeben. Die konkrete Realisierung des Web-basierten SfM Prototyps und die einzelnen fachlich motivierten Services und Methoden werden erst im nächsten Abschnitt vorgestellt. Das Grundgerüst der Web-basierten SfM Anwendungslogik-Architektur besteht aus folgenden Komponenten:

- Fachliche Services
- Integrationsservice

Fachliche Services: Der Wissensträger der Anwendungslogik stellt über dessen EJBean Remote (Local) und Remote (Local) home Interfaces (siehe Kap. 2.2.2.3) die fachliche Funktionalität für die Remote und Local Clients (siehe Kap. 2.2.2.4) bereit. Jede über den Interaktionsservice gestellte Anfrage wird über diese Schnittstellen bearbeitet. Darauf folgend bieten die Schnittstellen einzelne Fachliche Services an. Anfragen durch interne Klienten innerhalb der Anwendungslogik werden von den einzelnen Schnittstellen der Fachlichen Services entgegengenommen. Diese Vorgehensweise ermöglicht das Kapseln des fachlichen Wissens in einzelne unabhängige Einheiten, die fachlich getrennte Zuständigkeiten besitzen. Daraus werden einzelne Fachliche Services bereitgestellt, die den fachlich motivierten Umgang mit der Anwendungslogik unterstützen.

[29] Der Entwurf der Tabellenstruktur der Anwendungs- und Hilfsmaterialien: Servicefloat, Servicepointscript, Customer, Editor, Provider, Address, Document, XML-In und XML-Out ist im Anhang Abschnitt 6.1 dieser Arbeit zu sehen.

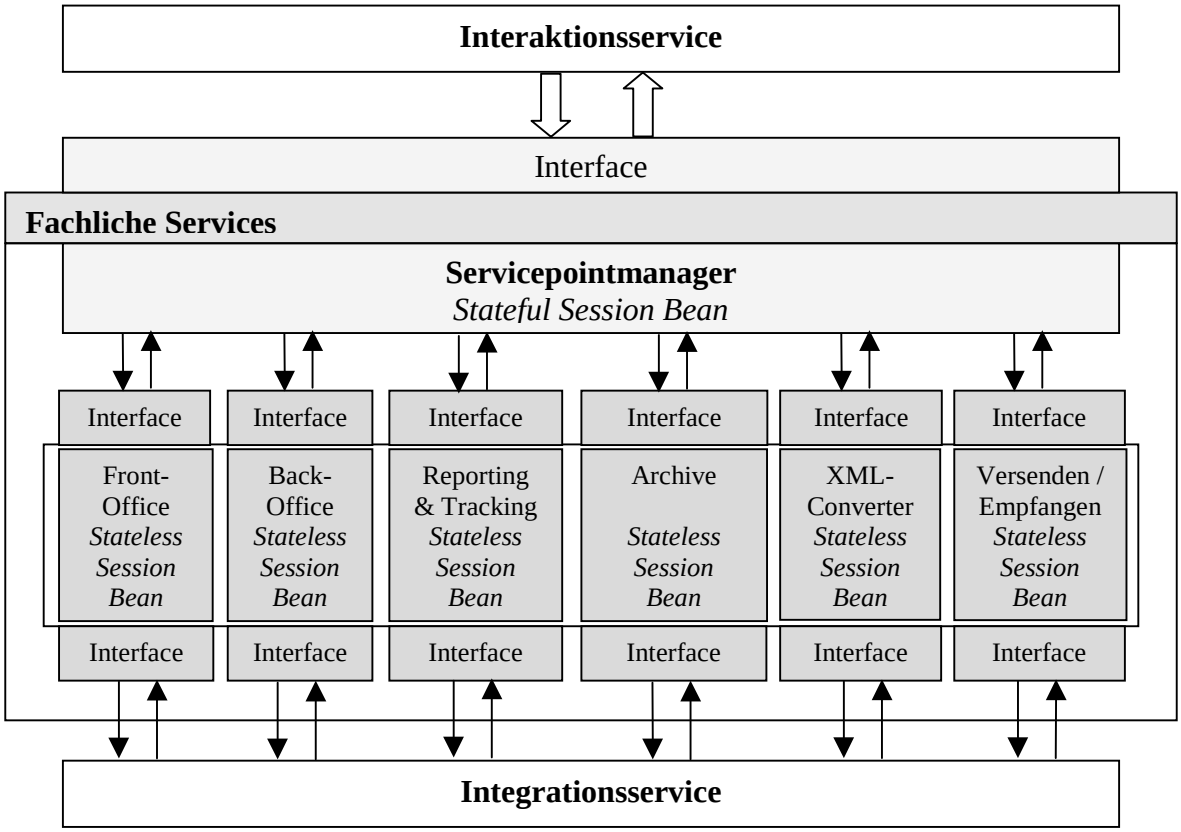

Abbildung 17 - Übersicht der Fachlichen Services Komponente

Der Bürger veranlasst den Interaktionsservice über die fachliche Schnittstelle (s. Abbildung 17) mit dem Servicepointmanager zu kommunizieren. Die Servicepointmanager Schnittstelle übernimmt dabei die Aufgabe den fachlich motivierten, sondierenden und verändernden Umgang durchzuführen. Der Servicepointmanager nimmt die Anfragen entgegen und delegiert sie an die einzelnen spezialisierten Fachlichen Services weiter. Einzelne Fachliche Services sind als zustandslose Session Beans realisiert worden. Deren Realisierung stützt sich dabei in großem Maße auf die EJB-Vorgehensweise und das EJB-Komponentenmodell. Das Wissen des Servicpointmanagers und der einzelnen Fachlichen Services ist in den implementierten EJBean Klassen angesiedelt, die für die Außenwelt nur über ihre fachlichen Schnittstellen erreichbar sind.

Die Kommunikation zwischen einzelnen Fachlichen Services findet entsprechend über deren fachliche Schnittstellen statt. Jeder einzelne Fachliche Service ist außerdem in der Lage über den Integrationsservice auf dem darrunterliegenden Persistenzmedium Daten und Fachwerte direkt zu lesen, zu aktualisieren oder zu löschen. Zudem werden Benutzbeziehungen, Delegation und Aggregation zwischen den einzelnen Fachlichen Services eingesetzt. Das Versenden (über XML-Out) und Empfangen (in XML-In) von Servicefloats läuft über den spezialisierten Fachlichen Services „Versenden/Empfangen“ und nicht über den Servicepointmanager.

Die Fachlichen Services werden durch weitere Dienste des EJB-Komponentenmodells in der Umsetzung dieser Anwendungslogik-Architektur unterstützt. Das Messaging über Nachrichtendienstmodelle: publish-and-subscribe oder point-to-point steht für den Austausch zwischen den Fachlichen Services zur Verfügung. Ferner sorgt das implizite Transaktionsmanagement (siehe Kap. 2.2.5) dafür, dass die Methoden der Fachlichen Services in einen Transaktionskontext gesetzt werden. Dies geschieht in dem die entsprechenden Eintragungen im Deployment descriptor (siehe Kap. 2.2.2.3) vorgenommen werden.

Folgende Fachliche Services werden angeboten:

- Servicepointmanager: Der Fachliche Service Servicepointmanager ist die zentrale Einheit für die Anfragen des Interaktionsservice.
- Front-Office: Fachlicher Service für die Front-Office-Vorgänge, wie z.B. das Finden, Lesen, Ändern und Schreiben von einzelnen Servicefloats und Servicepointscripts
- Back-Office: Fachlicher Service für die Back-Office-Vorgänge, wie z.B. Anlegen eines Editors, Providers, Citizen, etc.
- Reporting & Tracking: Fachlicher Service für Reporting und Tracking der Vorgänge. Eine Übersicht über schon abgeschlossene Vorgänge wird gegeben.
- Archive: Fachlicher Service für die Archivierung der Vorgänge und der bearbeiteten Anwendungsmaterialien.
- XML-Converter: Fachlicher Service für das Lesen, Finden, Ändern und Schreiben von versand- und empfangsfertigen XML-Dateien. In ihrer Gesamtheit ist das eine technische Konstruktion zur Umwandlung der Anwendungsmaterialien in das XML-Format und umgekehrt. Wird ein anderes Austauschformat verlangt, kann dieser Fachliche Service durch jenes ersetzt werden.
- Versenden/Empfangen: Fachlicher Service für das Versenden und Empfangen der Anwendungsmaterialien im XML-Format (sowohl im Self-Service- als auch Distributed-Service-Modi)

Integrationsservice: Die Anwendungs- und Hilfsmaterialien werden über den Integrationsservice auf dem Persistenzmedium konserviert. Der Integrationsservice sorgt für die automatische Persistenzverwaltung der Materialien und bietet fachlich motivierten Umgang mit diesen an. Zudem stellt der Integrationsservice über die EJB-Container-Dienste, im Falle eines Systemabsturzes, die Wiederherstellung des letzten konsistenten Zustands (der einzelnen Materialien) sicher. Die Materialien werden in mehrere Entity Beans (siehe Kap. 2.2.4.2) aggregiert. Abbildung 18 gibt eine Übersicht des Integrationsservice.

Der Einblick in das Innere des Integrationsservice zeigt auf der softwaretechnischen Entwurfsebene den Zusammenhang dieser Architektur mit dem EJB-Komponentenmodell auf. Jedes Anwendungsmaterial repräsentiert eine Entity Bean, das mit den Konstrukten und Diensten des EJB-Containers, hier im einzelnen der impliziten Container Managed Persistence (siehe Kap. 2.2.6) sowie der Container Managed Transaction Demarcation (siehe Kap. 2.2.5), realisiert wurde. Dabei stellt jedes Material, ähnlich wie bei den Fachlichen Services, mindestens eine Remote (Local) oder/und

Remote (Local) home Schnittstelle für den fachlich motivierten Umgang, wie z.B. das Erzeugen, Löschen oder Finden eines Materials oder über Getter- und Setter-Methoden persistente Attribute abzurufen bzw. zu setzen, zur Verfügung.

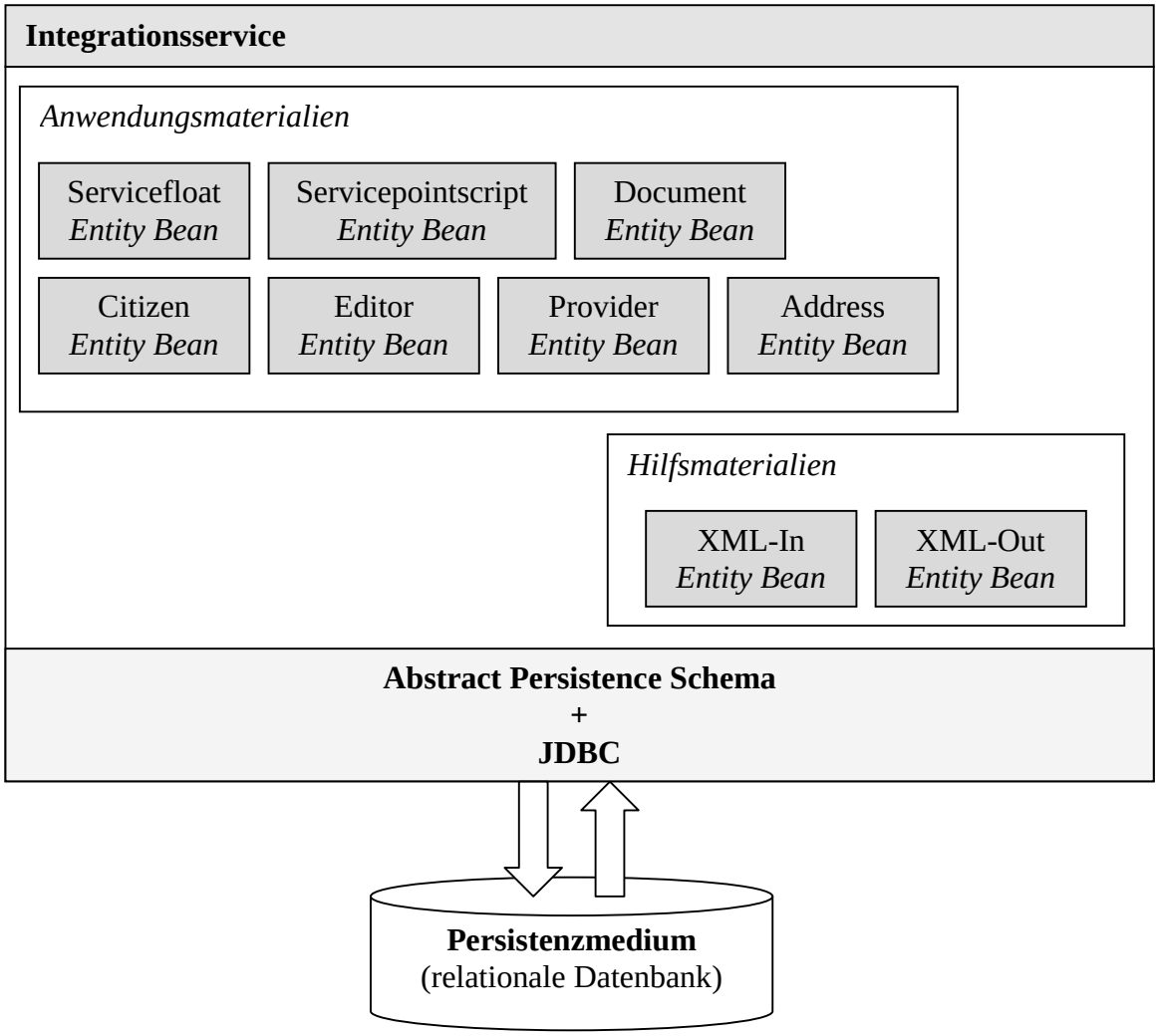

Abbildung 18 - Integrationsservice-Übersicht

Der Aufbau des Integrationsservices besteht aus in den Entity Beans realisierten Anwendungs- und Hilfsmaterialien sowie dem Abstract Persistence Schema. Jede Entity Bean bildet die Tabellenstruktur bzw. die Attribute eines Materials ab. Relationen zwischen den Entity Beans bzw. deren Attributen werden in dem Abstract Persistence Schema festgelegt. Die Angaben über den Integrator, hier der JDBC, wird während der Einsatzzeit auf dem EJB- bzw. Application-Server mithilfe von Installations-Tools festgelegt. Der EJB-Container verwendet die Angaben des Abstract Persistence Schema sowie der implementierten Entity Bean Klassen, um mit dem Persistenzmedium zu kommunizieren.

Der Integrationsservice agiert mit der Fachlichen Services Komponente in dem Gesamtkontext der Anwendungslogik als ganz normaler Dienstleistungsabnehmer. Der

Integrationsservice ist für die Fachlichen Services ein interner EJB-Client (siehe Kap. 2.2.2.4) und hat nur über die einzelnen fachlichen Schnittstellen einen Zugang zum Dienstleistungsspektrum der gesamten Fachlichen Services.

Der EJB-Container bleibt weiterhin die tragende Komponente hinsichtlich des impliziten Transaktions- und Persistenzmanagements. Einzelne Methoden der in Entity Beans modellierten Materialien werden durch das Festlegen der Transaktionsattribute im Deployment descriptor transaktionsfähig oder -los gemacht. Welche Attribute der Materialien persistent konserviert werden sollen, wird im Abstract Persistence Schema bestimmt. Außerdem werden auch einzelne Relationsmodelle zwischen den Attributen der Materialien sowie EJB Query Methoden (Finder/Select) definiert.

3.3.2.2 Anwendung in dem Fallbeispiel „Briefwahlantrag“

Im Unterschied zu dem alten Web-basierten SfM Prototyp eines Web-basierten Servicepoints, setzt der neue Web-basierte SfM Prototyp voraus, dass der Bürger eine Personalisierung in dem Bürgerprozessportal (z.B. http://mein.hamburg.de) vornimmt. Über diese Vorgehensweise lassen sich Allgemeine-Bürgerinformationen, z.B. alles über Bildung, Veranstaltungen, Hotels, etc. in einer Stadt/Bezirk mit direkten Bürgerinformationsdiensten und -vorgängen, wie z.B. Anträge an Bezirksämter oder Behörden stellen, verbinden und für den Bürger gemeinsam und personalisiert präsentieren.

Das Interaktionsservice-Web-Frontend des alten Web-basierten SfM Prototyps (siehe Kap. 3.2.3.3) wurde vom Aussehen und der Handhabung in dem neuen Web-basierten SfM Prototyp übernommen. Anstatt der TclBlend-Schnittstelle zum Servicepointmanager wird diese Schnittstelle in dem neuen Web-basierten SfM Prototyp mit JSP/Servlets realisiert. Der Servicepointmanager bleibt weiterhin die erste Anlaufstelle für jede Anfrage über den Interaktionsservice. Die Aufgabe des Servicepointmanagers beschränkt sich jedoch bei dem neuen Web-basierten SfM Prototyp auf die Bürger-Sitzungs-Zuordnung und das Delegieren von Anfragen an die einzelnen spezialisierten zustandslosen Fachlichen Services. Dadurch hat der Servicepointmanager das Wissen von allen zustandslosen Fachlichen Services. Diese werden über die Servicepointmanager-Schnittstelle dem Interaktionsservice angeboten. Die einzelnen Fachlichen Services haben andererseits den direkten Zugriff auf den Integrationsservice (sie greifen nicht über den Servicepointmanager zu).

Über den Interaktionsservice wird dem Bürger im personalisierten Bereich durch die JSP/Servlets-Schnittstelle `javax.servlet.http.HttpSession` eine Sitzungskennung über den gesamten Verlauf seiner Interaktion mit dem Bürgerprozessportal zugewiesen. Das Sitzungs-Objekt wird an die EJB-Anwendungslogik übergeben bzw. von dem Servicepointmanager in die zustandsbehaftete Session Bean (siehe Kap. 2.2.4.1) aufgenommen (vgl. [JEWELL01]). Dadurch wird einem Bürger eine Session Bean Instanz durch den Servicepointmanager zugeordnet, die über mehrere Anfragen desselben Bürgers den internen Zustand, auch bekannt als *conversational state*, speichert. Dem Bürger werden über den Servicepointmanager Konstrukte zur Verfügung gestellt, die den

internen Zustand der Session Bean Instanz verändern und während seiner Sitzung auf diesen zugreifen können. Der interne Zustand wird nicht persistent gespeichert, wie im Falle einer Entity Bean, sondern ist für den Bürger nur während seiner Sitzung vorhanden.

Bei einer Anfrage des Bürgers werden über den Servicepointmanager mindestens die Objekte "Servicefloat" und "Servicepointscript" erzeugt (ggf. auch die Objekte "citizen", "editor", "provider", etc. aus angegebenen Quellen) und in ein Sitzungs-Objekt reingetan. In dem Sitzungs-Objekt des Bürgers befindet sich u.a. der eindeutige Bürger-Identifikator (Citizen-Primärschlüssel) `citizen_id`. Von dem Servicepointmanager veranlasst, sendet der zustandslose Fachliche Service für das Lesen, Ändern, Schreiben und Finden/Suchen als erstes eine Anfrage über die Methode `findByCitizenPrimaryKey(Integer citizen_id)` an die Servicefloat Entity Bean. In der EJB QL Abfragesprache bzw. in dem Deployment descriptor werden für diese Finder-Methode folgende Angaben wie in Abbildung 19 gemacht:

```
<query>
	<query-method>
		<method-name>findByCitizenPrimaryKey</method-name>
		<method-params>
			<method-param>java.lang.Integer</method-param>
		</method-params>
	</query-method>
	<ejb-ql>
		SELECT OBJECT(sf) FROM Servicefloat sf
		WHERE sf.citizen_id = ?1
	</ejb-ql>
</query>
```

Abbildung 19 - Ausschnitt aus dem Deployment Descriptor des EJB-Prototyps

Hat der Bürger schon den Vorgang „Briefwahlunterlagen beantragen“ angestoßen, so wird das entsprechende Servicefloat-Objekt zurückgegeben. Andernfalls wird über die `createServicefloat(String sf_name, String sf_type, Integer customer_id)` ein neues Servicefloat-Objekt aus einer Vorlage erzeugt. Das Servicefloat wird danach von dem Servicepointmanager in das Sitzungs-Objekt des Bürgers aufgenommen. Über die Remote (Local) und Remote home (Local home) Interfaces des Servicefloats können entsprechend Getter-/Setter-Methoden (*accessor methods*) für einzelne Attribute des Servicefloats gesetzt bzw. abgerufen werden.

Als nächster Schritt wird für den im Servicefloat gesetzten aktuellen Servicepoint das zugehörige Material-Servicepointscript ausfindig gemacht oder neu aus der Vorlage erzeugt. Die Angaben über den aktuellen sowie geplanten Servicepoint werden im Servicefloat als Fremdschlüssel geführt. Für das Finden des aktuellen Servicepoints (oder auch derer, die als geplant markiert sind) wird die Finder-Methode des Servicepointscripts Entity Bean `findByPrimaryKey(Integer key)` verwendet. Danach wird das Servicpointscript-Objekt über den Servicepointmanager in das Sitzungs-Objekt des Bürgers aufgenommen. Für den sondierenden und verändernden Umgang mit den einzelnen Attributen des Servicepointscripts werden auch hier über die Remote (Local) und Remote home (Local home) entsprechend Setter-/Getter-Methoden bereitgestellt.

Der Servicepointmanager setzt jeden dazugehörigen Aufruf von einzelnen zustandslosen Fachlichen Services in einen impliziten Transaktionskontext (siehe Kap. 2.2.5). Dadurch ist die Ausführung eines solchen Fachlichen Services immer vom Erfolg abhängig. Ansonsten wird ein automatischer Rollback des letzten Aufrufs seitens des EJB-Containers ausgelöst. Darüber hinaus werden jeweils einzelne Methoden der zustandslosen Fachlichen Services in deren Transaktionskontext gesetzt. Somit bleibt die gesamte Anwendungslogik des neuen Web-basierten SfM Prototyps während der ganzen Interaktion mit dem Bürger im konsistenten Zustand.

Der Austausch von Servicefloats zwischen den einzelnen Servicepoints findet über die Hilfsmaterialien XML-In und XML-Out statt. In dem neuen Web-basierten SfM Prototyp werden ankommende und ausgehende Servicefloats in die Datenbank bzw. Datenbanktabelle importiert bzw. exportiert. XML-In und XML-Out werden nicht über den Servicepointmanager verwaltet, weil hierfür keine Bürger-Sitzungs-Zuordnung benötigt wird. Die Bearbeitung von Servicefloats in XML-In und XML-Out verläuft nach dem Prinzip der Stapelverarbeitung, die von dem zustandlosen Fachlichen Service „XML-Converter“ in ein DOM-Objekt umgewandelt bzw. in eine XML-Datei ausgeschrieben wird. Für das Einlesen der Servicefloats in die Datenbank (in XML-In) wird der vom EJB-Container bereitgestellte JAXP-Dienst genutzt. Für das Ausschreiben der Servicefloats in XML-Dateien aus der Datenbank (aus XML-Out) wird ebenfalls der JAXP-Dienst verwendet. Für das Versenden und Empfangen der Anwendungsmaterialien im XML-Format ist der zustandslose Fachliche Service „Versenden/Empfangen“, der als Session Bean `SendReceiveBean implements javax.ejb.SessionBean` realisiert wurde, zuständig.

Des weiteren werden Reporting Prozesse und einzelne Statusabfragen (tracking), die den Benutzer und den Dienstleister über den Verlauf des Vorgangs informieren, jeweils in den personalisierten Bereichen angezeigt. Für den Dienstleister steht zudem der Back-Office-Bereich, in dem er verwaltungstechnische Aufgaben erledigen kann, zur Verfügung. Ferner werden die Anwendungs- und Hilfsmaterialien des gesamten Prozesses des Dienstleistungsflusses in den jeweiligen Servicepoints archiviert.

3.3.3 Bewertung der mit EJB realisierten SfM Anwendungslogik

In diesem Abschnitt wird eine Bewertung der mit EJB realisierten SfM Anwendungslogik des Web-basierten SfM Prototyps eines Web-basierten Servicepoints durchgeführt. Hier werden sowohl innere Entwurfsprinzipien als auch äußere Qualitätskriterien des Prototyps bezüglich der im Kapitel 2.1 als bedeutsam ausgearbeiteten Anforderungen und Kriterien für die Entwicklung Web-basierter Anwendungen untersucht. Darüber hinaus werden einige Kriterien für den Einsatz der EJB in der Realisierung von eGovernment-Bürgerprozessportalen kurz besprochen.

3.3.3.1 Nach softwaretechnischen Entwurfsprinzipien

In diesem Abschnitt wird der Web-basierte SfM Prototyp und die Anwendungslogik nach folgenden softwaretechnischen Entwurfsprinzipien bewertet.

Kapselung

Bei der Realisierung des neuen Web-basierten SfM Prototyps wurde das Entwurfskriterium der Kapselung eingehalten. Der Web-basierte SfM Prototyp wurde nach der EJB-Vorgehensweise implementiert, die auf die Beachtung des Geheimnisprinzips und der Trennung von Schnittstelle und Implementation setzt. Die Möglichkeit einer Kapselung der Anwendungslogik in Session Beans sowie der Anwendungsdaten in Entity Beans nach der EJB-Vorgehensweise wurde vorgenommen. Zu jeder verwendeten Komponente existiert durch das EJB-Komponentenmodell mindestens eine Schnittstelle, welche die angebotenen Leistungen spezifiziert. Durch diese Vorgehensweise kann jede Komponente der Anwendungslogik: Fachliche Services oder Integrationsservice problemlos gegen eine andere Implementation ausgetauscht werden.

Das Geheimnisprinzip wird sowohl auf der Ebene des Servicepointmanagers und der einzelnen zustandslosen Fachlichen Services, als auch auf der Ebene des Integrationsservice eingehalten. Dies wurde erreicht indem die Fachlichen Services und der Integrationsservice durch die fachlich motivierten modellierten Klassen (und Methoden) implementiert wurden, welche die innere Datenstruktur bzw. implementierungsbezogenen Interna verbergen und nur Fachwerte liefern. Benutzende Komponenten haben dadurch keine Kenntnisse darüber, mit welchen Mitteln sie ihre Dienstleistungen erbringen.

Trennung von Zuständigkeiten

Durch die Aufteilung des Web-basierten SfM Prototyps in mehrere Schichten, die jeweils klare Zuständigkeiten besitzen, wurde diesem Entwurfskriterium in hohem Maße nachgegangen. Die Trennung von Zuständigkeiten orientierte sich an den Vorgaben des EJB-Komponentenmodells, das diese in seiner Architektur auf verschiedenen Ebenen (EJB-Architektur: EJB-System, EJB-Rollen, EJB-Bean, Deployment Descriptor, etc.) weitestgehend unterstützt. Auch bei der Realisierung der inneren Bestandteile der Anwendungslogik: Fachliche Services oder Integrationsservice ist eine klare Zuordnung von Aufgaben sowie Trennung von Zuständigkeiten unter Verwendung der EJB-Vorgehensweise und der Entwurfsmetapher der Fachlichen Services umgesetzt worden.

Bei näherer Betrachtung des Servicepointmanagers bezüglich des Aufgabendelegierens an die einzelnen zustandslosen Fachlichen Services, deren Einzelleistungen der Servicepointmanager zur Erbringung seiner eigenen Leistung verwendet, ist festzustellen, dass hier ebenfalls eine Trennung der Verantwortlichkeiten durchgeführt worden ist. Die Hauptaufgabe des Servicepointmanagers ist somit die Koordination zwischen den einzelnen zustandslosen Fachlichen Services sowie die Zuordnung von verschiedenen Anwendungsmaterialien an die Bürger.

Neben der Anwendungslogik-Komponente Fachliche Services erfüllt die Integrationsservice-Komponente im Wesentlichen auch das Entwurfskriterium der Trennung von Zuständigkeiten. Durch die Implementierung der Anwendungs- und

Hilfsmaterialien mit Entity Beans sowie der Festlegung der Laufzeitangaben im Deployment descriptor, insbesondere das Abstract Persistence Schema hinsichtlich der Container Managed Persistence, findet eine klare Abgrenzung der Aufgaben statt. Entity Beans (Materialien) bilden die Attribute einer Datenbanktabelle im Anwendungskontext ab. Im Abstract Persistence Schema wird das Persistenzverhalten von einzelnen Attributen definiert. Hinzu kommt noch das Festlegen der Relationen für eine oder mehrere Entity Beans bzw. Relationsmodelle zwischen den jeweiligen Anwendungsmaterialien.

Die Trennung von Zuständigkeiten auf der Ebene des Interaktionsservices wurde durch Nutzung der JSP/Servlets teilweise realisiert. Jedoch ist nahezu in jedem JSP/Servlet eine Ablauflogik eingebaut. Nur über die Iteration von Ergebniswerten und einfacheren Verzweigungsoperationen ist die Ablauflogik der Anwendungslogik in der Interaktionslogik vertreten.

Minimierte Kopplung

Durch die fachlich motivierte Modellierung und Realisierung der Anwendungslogik bzw. der einfachen einseitigen Benutzbeziehung, Delegation und Aggregation zwischen den einzelnen Komponenten ist die Kopplung innerhalb der Anwendungslogik als recht gering zu bezeichnen. Der Servicepointmanager ist die einzige Komponente in der Anwendungslogik, die das Wissen über einzelne zustandslose Fachliche Services besitzt. Er nimmt Anfragen vom Interaktionsservice entgegen und delegiert sie entsprechend an die zustandslosen Fachlichen Services. Im Integrationsservice wurden Anwendungs- und Hilfsmaterialien in Entity Beans aggregiert.

Wie bei den vorigen Entwurfsprinzipien sollte auch hier erwähnt werden, dass das EJB-Komponentenmodell die Realisierung der n-Tier-Anwendungen durchgehend unterstützt. Auf der Ebene der Enterprise JavaBean (siehe Kap. 2.2.2.3) bietet es mehrere kleine Schnittstellen an. Jede einzelne EJBean besitzt somit mindestens eine Schnittstelle. Die EJBeans, die die einzelnen Komponenten der Anwendungslogik darstellen, stehen in einfachen einseitigen Benutzbeziehungen zueinander und tragen dazu bei die Kopplung zu vermindern. Auch im Gesamtkontext des Web-basierten SfM Prototyps wurde darauf geachtet, dass die Kommunikation nur über eine minimierte Kopplung der einzelnen Schichten stattfindet.

Verwendung von Architektur- und Entwurfsmuster

Aus der Vorstellung des neuen Web-basierten SfM Prototyps wird ersichtlich, dass sowohl die Grobstrukturierung als auch einzelne Schichten des Entwurfs sehr stark an das EJB-Architekturmuster angelehnt wurde. Auf der Grobstrukturierungsebene des Entwurfs wurde das schichtenbasierte Architekturmuster verwendet. Jeder einzelnen Schicht ist eine Aufgabe zugewiesen worden, die für ein Dienstleistungsspektrum verantwortlich ist.

Auf der Anwendungslogikebene sind zunächst die Fachlichen Services und der Integrationsservice unterschieden worden. In den Fachlichen Services wurde das gesamte

Anwendungswissen bzw. die Anwendungslogik in mehrere fachlich motivierte Einheiten unterteilt. Der Servicepointmanager ist eine stateful Session Bean und verwendet das implizite Transaktionsmanagement. Durch das Delegieren der Aufgaben an die einzelnen zustandslosen Fachlichen Services verkörpert er die zentrale Einheit der Anwendungslogik. Zusammen mit einzelnen zustandslosen Fachlichen Services verwirklicht der Servicepointmanager die Fachliche Services Entwurfsmetapher in der Anwendungslogik. Auf der Ebene des Integrationsservices wurde darauf geachtet, dass dieser unabhängig von dem Persistenzmedium bleibt.

Es stellt sich die Frage, ob weitere Entwurfsmuster aus dem etablierten Desktopbereich angewandt werden könnten, anstatt der von der EJB-Architektur vorgegebenen Vorgehensweise für die Realisierung des server-seitigen Web-basierten SfM Prototyps?

3.3.3.2 Nach softwaretechnischen Qualitätskriterien

In diesem Abschnitt wird vergleichend zu der im Kapitel 3.2.4 durchgeführten Schwachstellenanalyse des alten Web-basierten SfM Prototyp die Anwendungslogik nach denselben softwaretechnischen Qualitätskriterien untersucht und bewertet.

Skalierbarkeit

Einer der Werbeslogans von EJB ist, dass eine EJB-Anwendung in einer J2EE Umgebung beliebig skalierbar ist. Dies bezieht sich auf die mögliche Erweiterung von Betriebsmitteln und Ressourcen des EJB-Servers und des EJB-Containers. Einer EJB-Anwendung kommt dies dann zugute, wenn sie nach der EJB-Vorgehensweise entwickelt wurde. Bei dem Web-basierten SfM Prototyp ist das der Fall, er wurde nach der EJB-Vorgehensweise entwickelt. Dadurch stehen ihm die bereitgestellten Mittel des EJB-Systems und EJB-Containers zur Verfügung, und er ist auf dieser Ebene somit skalierbar. Bei der Entwicklung der einzelnen Schichten des Web-basierten Prototyps wurde besonders auf die Skalierbarkeit und die Handhabung in der Praxis geachtet. Jedoch ist das Qualitätskriterium der Skalierbarkeit stärker auf der Anwendungslogik- und auf der Persistenzebene berücksichtigt worden, als auf der Interaktionsserviceebene.

Der Servicepointmanager ist nicht nach dem Singleton Entwurfsmuster (vgl. [GHJV96] S. 139) realisiert worden, sondern als stateful Session Bean. Dadurch lässt sich der Servicepointmanager in verteilten Umgebungen und J2EE-Architekturen beliebig einsetzen, weil er als einzige Instanz verfügbar ist. Dies wurde im Deployment descriptor festgelegt. Der Servicepointmanager braucht kein Wissen über das Vorhandensein der anderen Servicepointmanager zu haben und sich nicht mit den anderen zu koordinieren oder synchronisieren. Zudem kann der Servicepointmanager auf verschiedenen verteilten Rechnern (Cluster-Infrastruktur), auf denen jeweils eine eigene Java Virtual Machine läuft, betrieben werden. Der EJB-System und EJB-Container sorgen dafür, dass der Servicepointmanager über mehrere Rechner und Container die Bürger-Sitzungs-Zuordnung beibehält.

Auch in der Teilung der Anwendungslogik in Fachliche Services und Integrationsservice wird dem softwaretechnischen Qualitätskriterium nach Skalierbarkeit nachgegangen, vor allem durch die Möglichkeit im Deployment descriptor die Anzahl der instantiierten Exemplare von EJBeans (außer für den Servicepointmanager) der einzelnen zustandslosen Fachlichen Services und der Anwendungs- und Hilfsmaterialien festzulegen. Ferner wird zur Laufzeit die Möglichkeit eingeräumt, jegliche Einstellungen für die einzelnen Komponenten oder Methoden hinsichtlich des Ressourcen-, Zugriffs-, Transaktions- und Persistenzmanagments zu ändern oder anzupassen.

Robustheit

Der EJB Web-basierte SfM Prototyp schließt die Gefahr inkonsistenter Zustände, die in dem Fallbeispiel durch Verbindungsabbrüche oder konkurrierende Zugriffe vorkommen können, durch die Verwendung der EJB Container Managed Transaktions- und Persistenzverwaltung aus. Die kritischen Weiterschaltungen bzw. Übergaben von Zuständigkeiten von einem Fachlichen Service zum anderen, werden durch das Setzen des Transaktionskontexts für die einzelnen EJBeans und Methoden abgesichert. Tritt ein Fehler auf der Kommunikationsebene auf, wird vom EJB-Container unmittelbar ein Rollback ausgeführt, das den alten Zustand wiederherstellt. Da jede einzelne (Teil-) Aktion rückgängig gemacht werden kann, wird somit eine kritische Ausgangslage für die Anwendungslogik verhindert. Einziger Nachteil hierbei ist, dass die einzelnen Aktionen von den Bürgern wiederholt werden müssen. Darüber hinaus kann der Datenbank-Manager jedoch auf der Persistenzebene im Falle eines Systemabsturzes den letzten konsistenten Zustand der Anwendungs- und Hilfsmaterialien wiederherstellen.

Durch die impliziten Transaktions- und Persistenzverwaltungsdienste der EJB-Architektur wird die Robustheit des Web-basierten SfM Prototyps gewährleistet. Die EJB stellen ein sehr ausgereiftes Komponentenmodell zur Verfügung, das sich besonders für die Entwicklung robuster und skalierbarer Web-basierter Anwendungen eignet.

Portabilität

Eine der server-seitigen Vorteile der J2EE Technologie in Verbindung mit XML ist die Portabilität. Dieses Qualitätskriterium wurde durch die Verwendung und die Anlehnung an die EJB-Vorgehensweise bei der Entwicklung des EJB Web-basierten SfM Prototyps vollständig erfüllt. Eine in EJB entwickelte Anwendung läuft in jeder J2EE Umgebung.

Im Gegensatz zu dem alten Web-basierten SfM Prototyp sind die Anwendungs- und Hilfsmaterialien in einer relationalen Datenbank modelliert. Die Materialien liegen somit nicht auf dem File-System oder Web-Server. Außer für vereinbarte FTP-Austauschplätze der Materialien auf dem Web-Server wird im gesamten EJB-Code keine festverdrahtete Festlegung von jeglichen Datei-, Ordner-, EJB-System-, EJB-Container-Pfaden, etc. gemacht. Für die Austauschplätze werden zwei FTP-Accounts: XML-In und XML-Out benötigt, die für jeden Zugangsberechtigten von dem Systemadministrator (siehe Kap. 2.2.3) einzeln konfiguriert werden können. Des weiteren werden alle nötigen

Eintragungen für den EJB Web-basierten SfM Prototyp im Deployment descriptor vorgenommen.

Eine der weiteren Schwächen des alten Prototyps war, dass dieser keine zentrale oder portable Verwaltung von Benutzer- bzw. Bürgerdaten besaß. Dies wurde in dem neuen Prototyp durch die zentrale Aufnahme und Verwaltung von Bürger-, Sachbearbeiter- und Serviceerbringerdaten gelöst. Über die Anwendungslogik (als Middelware) werden die entsprechenden Remote- und Local-Schnittstellen für den Zugriff auf Benutzer- bzw. Bürgerdaten für anderweitige Anwendungen aus dem Desktop- oder Internet-Bereich bereitgestellt.

Sicherheit

Sicherheitsaspekte sind in der Entwicklung des EJB Web-basierten SfM Prototyps mitberücksichtigt worden. Hierfür wurden die von EJB angebotenen Sicherheitsmaßnahmen: Authentifizierung und Access control (siehe Kap. 2.2.2.2) angewandt.

```
...
<security-role>
	<description>
		This role is allowed to execute any method on the bean and to
		read and change any servicefloat bean data.
	</description>
	<role-name>
		Administrator
	</role-name>
</security-role>
<security-role>
	<description>
		This role is allowed to locate and read servicefloat info. This
		role is not allowed to change servicefloat bean data.
	</description>
	<role-name>
		ReadOnly
	</role-name>
</security-role>
<method-permission>
	<role-name>Administrator</role-name>
	<method>
		<ejb-name>ServicefloatBean</ejb-name>
		<method-name>*</method-name>
	</method>
</method-permission>
<method-permission>
	<role-name>ReadOnly</role-name>
	<method>
		<ejb-name>ServicefloatBean</ejb-name>
		<method-name>getServicefloatName</method-name>
	</method>
	<method>
		<ejb-name>ServicefloatBean</ejb-name>
		<method-name>getServicefloatType</method-name>
	</method>
</method-permission>
...
```

Abbildung 20 - Beispiel: Access control

Um das Dienstleistungsangebot in Anspruch zu nehmen, muss der Bürger als erstes in den personalisierten Bereich gelangen. Zu diesem Zweck stellt der EJB-Container die Zugriffskontrolle über „Role-driven access control“ (s. Abbildung 20) zur Verfügung.

Für jeden Zugangsberechtigten können mehrere Rollen definiert werden, die mitsamt den Zugangsberechtigungsdaten im Deployment descriptor festgelegt wurden. Durch den Deployer werden diese Zugangsberechtigungsdaten untersucht und entsprechend der reellen Gruppen in einem EJB-System abgebildet und eingesetzt. So kann z.B. die Rolle „ReadOnly“ für die Gruppe der Bürger eingesetzt werden, die sich über den Interaktionsservice in dem Bürgerprozessportal identifizieren. Diese Vorgehensweise ist in der EJB-Umgebung als „*Mapping roles in the operational environment to logical roles in the deployment descriptor*“ bekannt.

Über den JNDI-Kontext sind danach die Zugangsberechtigungsdaten für einen EJBean-Client zugänglich. Über die `java.security.Principal` Schnittstelle werden diese Angaben an den JNDI-Kontext bzw. EJBean-Client (s. Abbildung 21) übermittelt.

```
...

properties.put(Context.SECURITY_PRINCIPAL, userName);
properties.put(Context.SECURITY_CREDENTIALS, userPassword);

javax.naming.Context jndiContext = new
javax.naming.InitialContext(properties);
Object ref = jndiContext.lookup(“ServicefloatHomeRemote“);
ServicefloatHomeRemote servicefloatHome = (ServicefloatHome)
PortableRemoteObject.narrow(ref, ServicefloatHomeRemote.class);

...
```

Abbildung 21 - Abfragen von Zugangsberechtigung

Außer der vom EJB-Container bereitgestellten Zugriffskontrolle wurden weitere Dienste und Mechanismen des EJB-Komponentenmodells zur Gewährleistung von Sicherheitsaspekten in Anspruch genommen. Diese werden in der Anwendungslogik auf der Ebene der internen Kommunikation als auch auf der Ebene der korrekten und vollständigen Verfügbarkeit von einzelnen Fachlichen Services verwendet.

Darüber hinaus werden Servicefloats mitsamt der angehängten Anwendungsmaterialien vor dem Versenden an den nächsten Servicepoint, direkt in einem FTP-Postfach, nach den gängigen Sicherheitsmethoden verschlüsselt. Dieser Vorgang findet vor dem Ablegen in XML-Out statt. Die eingehenden Servicefloats kommen verschlüsselt in XML-In an und werden von einem Fachlichen Service entschlüsselt. Hierzu wird vorausgesetzt, dass eine Absprache und Spezifizierung über die Art der Verschlüsselung zwischen den Beteiligten in dem gesamten Dienstleistungsflussaustausch erreicht wurde.

Ein weiterer Sicherheitsaspekt des neuen EJB Web-basierten SfM Prototyps ist, dass auf der Persistenzebene seitens des Datanbank-Managers sichergestellt wird, dass nur die Zugangsberechtigten auf die Anwendungsmaterialien zugreifen können.

Wiederverwendung

Auch bei dem neuen EJB Web-basierten SfM Prototyp wurde der Schwerpunkt auf die Wiederverwendung gesetzt. Dies ist größtenteils durch die EJB-Architektur bzw. EJB Vorgehensweise zur Erstellung von J2EE konformen Anwendungen bedingt gewesen, aber auch bewusst gewählt. Auf jeder Ebene des Web-basierten SfM Prototyp Schichtenmodells wurde auf deren jeweilige Besonderheiten weitestgehend abstrahiert.

Bei der näheren Betrachtung der Anwendungslogik, speziell der Fachlichen Services und Integrationsservices, sind einzelne Bestandteile in jeder J2EE wiederverwendbar. Damit diese in eine andere J2EE-Umgebung eingesetzt werden können, sollen nur die Angaben über das verwendete Persistenzmedium im Deployment descriptor sowie im EJB- oder Application-Server angepasst werden. Einer der größten Vorteile von EJB basiert auf den wiederverwendbaren EJB-Komponententeilen, die die Entwicklung transaktionsfähiger, skalierbarer und portabler Web-Anwendungen unterstützen.

Integration

Während der Re-Implementierung des Web-basierten SfM Prototyps wurde insbesondere in der Anwendungslogik auf dieses Qualitätskriterium geachtet. Einzige Voraussetzung für die Integration der Anwendungslogik in verschiedenen Infrastrukturen und Umgebungen (s. Abbildung 2) ist die Bereitstellung einer Java Virtual Machine für das Ausführen der Anwendungslogik.

- Web-Frontend: HTML
- Interaktionsservice: JSP/Servlets
- Anwendungslogik:
  - Fachliche Services: Verwendung des server-seitigen EJB-Komponentenmodells sowie Anlehnung an die Fachliche Services Entwurfsmetapher. Servicepointmanager ist als stateful Session Bean implementiert worden. Einzelne zustandslose Fachliche Services sind als zustandslose Session Beans realisiert worden.
  - Integrationsservice: Die objektorientierte Sicht auf die Anwendungs- und Hilfsmaterialien wurde durch die Verwendung der Entity Beans umgesetzt. Jede Entity Bean bildet die Materialien der Anwendungswelt ab.
- Persistenzmedium: Relationale Datenbank

Dadurch ist die Integration der oben einzeln aufgeführten Punkte aber auch des gesamten EJB Web-basierten SfM Prototyps in beliebigen J2EE konformen EJB- und Application-Servern sowohl in homogenen und als auch heterogenen Infrastruktur-Konstellationen schnell realisierbar.

3.3.3.3 Nach Kriterien bezogen auf den Anwendungskontext

In diesem Abschnitt wird eine Eignung von EJB hinsichtlich des Anwendungskontexts Web-basierte eGovernment-Bürgerprozessportale besprochen. Folgende Kriterien, die als

Besonderheiten eines eGovernment-Bürgerprozessportal in Kapitel 3.1 ausgearbeitet wurden und für die Entscheidungsfindung eines Unternehmens (oder Regierung, Verwaltung, etc.) für die Realisierung eines Web-basierten eGovernment-Bürgerprozessportals mit EJB bedeutsam sein können, werden besprochen.

Prozessportal

EJB bietet ein server-seitiges Komponentenmodell für die Realisierung von eGovernment-Bürgerprozessportalen. Die Entwicklung von eGovernment-Bürgerprozessportalen, die eine Personalisierung anbieten, wird durch die impliziten und expliziten Dienste der EJB-Architektur unterstützt. Somit lassen sich eGovernment-Bürgerprozessportale, die organisationsübergreifend in verteilten Umgebungen laufen, schnell errichten. Weiterhin ermöglicht die EJB-Architektur das Kapseln des fachlichen Wissens in der Anwendungslogik eines eGovernment-Bürgerprozessportals. Durch die EJB empfohlene Vorgehensweise bleibt die gekapselte Anwendungslogik somit handhabungs- und präsentationsunabhängig. Hierfür stellt das EJB-Komponentenmodell eine Reihe von Dienstleistungen zur klaren Trennung von Interaktions-, Anwendungs- und Datenlogik zur Verfügung.

Authentifizierungs- und Zugriffskontrolle

Die Empfindlichkeit der Daten, insbesondere in eGovernment-Bürgerprozessportalen hinsichtlich der Sicherheitsvorkehrungen, wird durch die impliziten und expliziten EJB-Container-Dienste zur Gewährung und Sicherung der Authentifizierung- und Zugriffskontrolle geschützt. EJB stellt zentrale Einstellungsmöglichkeiten von Rollen und Zugriffsrechten im Deployment descriptor zur Verfügung, die die Erfüllung dieser Aufgaben weitestgehend unterstützen.

Skalierbarkeit und Robustheit

Das eGovernment-Bürgerprozessportal ist jederzeit durch Erweiterung bzw. Hinzufügen von weiteren Ressourcen und Mitteln seitens der EJB-Server bzw. EJB-Container skalierbar. Dadurch können während der Laufzeit immer wieder neue Komponenten hinzukommen, die das eGovernment-Bürgerprozessportal in verteilten Umgebungen ebenfalls unterstützen würden. Darüber hinaus sorgt das Container Managed Transaktionskonzept (siehe Kap. 2.2.5) dafür, dass das eGovernment-Bürgerprozessportal während und nach der getätigten Bürgerinteraktion im konsistenten Zustand bleibt.

Prozesspooling und Nebenläufigkeit

Ein instantiierter Prozess, der mehrere Servicepoints innerhalb des eGovernment-Bürgerprozessportals durchläuft, kann jederzeit mit Hilfe des EJB-Container-Ressourcen-Managements zur Ruhe gesetzt bzw. für spätere Aktionen reaktiviert werden (siehe Kap. 2.2.2.2). Des weiteren stellt der EJB-Container Dienste bereit, die das Prozesspooling und die Nebenläufigkeit unterstützen.

Mehrbenutzerfähigkeit

Die EJB-Architektur unterstützt durch zahlreiche implizite und explizite Mechanismen die Realisierung von mehrbenutzerfähigen eGovernment-Bürgerprozessportalen in Cluster-Infrastrukturen. Auf dieser Ebene und insbesondere bei der Bearbeitung von großen Anfragevolumendaten können eGovernment-Bürgerprozessportale sich auf die technischen Eigenschaften des EJB-Komponentenmodells stützen.

Dynamische Modellierung und situative Anpassung

Der Deployment descriptor ermöglicht Laufzeitveränderungen und -anpassungen der eGovernment-Bürgerprozessportale und somit auch die dynamische Modellierung sowie situative Verhaltensanpassung und -veränderung von EJBeans (siehe Kap. 2.2.2.3) bzw. deren Methoden. Zudem ermöglicht die EJB-Architektur auch auf der Transaktions- sowie Persistenzsteuerunsgebene Laufzeitveränderungen und -anpassungen vorzunehmen.

Wiederverwendbarkeit und Integration

Die in EJB realisierten Komponententeile sind auf weiteren J2EE-Systemen wiederverwendbar. Durch die Plattformunabhängigkeit sowie durch die Verwendung der Java Programmiersprache und XML stellt EJB Schnittstellen und Dienste bereit, die von anderen Anwendungen bzw. Systemen verwendet werden können. Ein nach der EJB-Vorgehensweise implementiertes eGovernment-Bürgerprozessportal kann ohne große Veränderungen schnell in eine andere J2EE-Umgebung integriert werden.

Im Allgemeinen ist zu sagen, dass die EJB-Architektur durch das gesamte implizite und explizite Dienstleistungsspektrum gemeinsam mit Web-Komponenten (s. Kap. 2.3) bzw. J2EE-Architektur eine gute Plattform für die Realisierung von Web-basierten eGovernment-Bürgerprozessportalen anbietet.

3.3.4 Zusammenfassung

In diesem Kapitel wurde die Architektur des mit EJB realisierten Web-basierten SfM Prototyps mit dem Schwerpunkt auf der Anwendungslogik untersucht.

Als erstes wurde meine Vorgehensweise bei der Re-Implementierung des Web-basierten SfM Prototops dargestellt. Eine Optimierung des Web-basierten SfM Prototyps durch den Einsatz der EJB wurde vorgeschlagen und implementiert. Eine neue Anwendungslogik-Architektur wurde in der neuen Aufteilung: Fachliche Services und Integrationsservice vorgestellt und besprochen. Als Bestandteil der Fachlichen Services wurde der Servicepointmanager, der ein stateful Session Bean repräsentiert, in dem Fallbeispiel „Briefwahlantrag“ mit seinen neuen Zuständigkeiten: Bürger-Sitzungs-Zuordnung und Aufgabendelegierung an die einzelnen Fachlichen Services, erläutert. Die Aufgaben und Zuständigkeiten der einzelnen zustandslosen Fachlichen Services wurden benannt. Im Anschluss daran wurde die Architektur des Integrationsservices vorgestellt. Einen

wichtigen Bestandteil dieser Architektur stellen die Entity Beans dar, deren Aufgabe die Aggregation sowie die Umhüllung der Anwendungs- und Hilfsmaterialien in einzelne Objekte mit klar definierten Schnittstellen ist. Die Funktionalität des Abstract Persistence Schema zur Realisierung der impliziten Persistenzverwaltung wurde erläutert.

Danach folgte eine Bewertung der mit EJB realisierten SfM Anwendungslogik nach softwaretechnischen Entwurfsprinzipien (in Tabelle 8) und Qualitätskriterien (in Tabelle 9). Diese heben die Verbesserung der Schwachstellen des alten Web-basierten SfM Prototyps, durch den Einsatz der EJB-Technologie, hervor.

Entwurfsprinzip	***Erfüllt***
Kapselung	durch Trennung von Schnittstelle und Implementation sowie Einhaltung des Geheimnisprinzips
Trennung von Zuständigkeiten	durch Vorgaben der EJB-Architektur; klare Zuordnung von Aufgaben und Zuständigkeiten
Minimierte Kopplung	durch viele lose Einheiten mit spezialisierten Aufgaben
Verwendung von Architektur- und Entwurfsmustern	durch die EJB-Vorgehensweise; keine Verwendung von Desktop-Entwurfsmustern

Tabelle 8 - Erfüllung allgemeiner softwaretechnischer Entwurfsprinzipien durch den EJB Web-basierten SfM Prototyp

Qualitätskriterium	***Erfüllt***
Skalierbarkeit	durch die EJB-Vorgehensweise sowie die Realisierung des Servicepointmanagers als stateful Session Bean
Robustheit	durch das implizite Transaktions- und Persistenzmanagement des EJB-Containers
Portabilität	durch die J2EE-Architektur sowie Java + XML
Sicherheit	durch die implizite Authentifizierung- und Zugriffskontrolle des EJB-Containers
Wiederverwendung	durch das EJB-Komponentenmodell und die schichtenbasierte Architektur des Prototyps
Integration	durch die Anlehnung an die J2EE-Architektur; integrierbar in jeder J2EE Umgebung

Tabelle 9 - Erfüllung allgemeiner softwaretechnischer Qualitätskriterien durch den EJB Web-basierten SfM Prototyp

Im Anschluss daran wurde anhand einiger Kriterien eine Eignung der EJB für die Realisierung von eGovernment-Bürgerprozessportalen durchgeführt. Ergebnis dieser Eignung war, dass die EJB-Technologie die geeignete Plattform und Konstrukte für deren Realisierung auf der Ebene der Anwendungslogik zur Verfügung stellt. Vor allem in Bezug auf die Empfindlichkeit der Bürgerdaten innerhalb eines eGovernment-Bürgerprozessportals hinsichtlich der Authentifizierungs- und Zugriffskontrolle bietet EJB Mechanismen zur Gewährleistung dieser Kriterien. Des weiteren ermöglicht das

EJB-Komponentenmodell die Realisierung von robusten und skalierbaren Bürgerprozessportalen. Tabelle 10 fasst die Ergebnisse zusammen.

Kriterium	***Ergebnisse***
Prozessportal	unterstützt durch das breite Dienstleistungsspektrum des EJB-Komponentenmodells
Authentifizierung- und Zugriffskontrolle	unterstützt durch die EJB-Container Authentifizierung und Zugriffskontrolle
Skalierbarkeit und Robustheit	unterstützt durch die EJB empfohlene automatische Persistenz- und Transaktionsverwaltung
Prozesspooling und Nebenläufigkeit	unterstützt durch das automatische Ressourcen- und Lebenszyklusmanagement seitens des EJB-Container
Mehrbenutzerfähigkeit	unterstützt durch die impliziten Dienste des EJB-Containers
Dynamische Modellierung und situative Anpassung	unterstützt durch die zentralen Einstellungsmöglichkeiten im Deployment descriptor
Wiederverwendbarkeit und Integration	unterstützt durch den Einsatz der J2EE-Architektur

Tabelle 10 - Eignung von EJB für die Realisierung eines eGovernment-Bürgerprozessportals

4 Resümee

In diesem Kapitel werden die Ergebnisse dieser Arbeit vorgelegt und die Antworten auf die Fragestellung gegeben. Die wesentlichen Schritte zur Beantwortung der Fragestellung werden dabei dargelegt. Der Ausblick schließt am Ende des Kapitels diese Arbeit ab.

4.1 Ergebnis/Antworten

In diesem Abschnitt werden Ergebnisse und Antworten auf die Fragestellung dieser Arbeit vorgelegt.

Einsatz der EJB in der Entwicklung Web-basierter Anwendungen

Die allgemeine Frage meiner Arbeit war: „*Welche Vorteile bietet der Einsatz der Enterprise JavaBeans (EJB) in der Praxis der Web-Entwicklung? (z.B. In welchem Kontext bzw. Projektumfang empfiehlt sich der Einsatz der EJB?)*“

Das EJB-Komponentenmodell stellt für die Web-Entwicklung eine gute Plattform zur Realisierung von robusten, skalierbaren und mehrbenutzerfähigen server-seitigen Web-basierten Anwendungen zur Verfügung. Die Anlehnung an die EJB-Architektur bzw. EJB-Vorgehensweise sichert eine zügige Entwicklung der Web-basierten Anwendungen bzw. Anwendungslogik. Auf der Modellierungsebene stellt die EJB-Architektur Abstraktionsmöglichkeiten für die Realisierung der n-Tier Web-basierten Anwendungen bereit. Sie unterstützt das Erstellen der Anwendung in mehreren Schichten und insbesondere eine klare Trennung der Schichten in Anwendungs-, Daten- und Benachrichtigungslogik. Ein weiterer Aspekt für die Web-Entwicklung ist, dass EJB die Implementierung der Thinner Clients (ultra-lightweight client interface) durch die Integration von J2EE Technologien, wie Java Servlets/JSP, unterstützen.

Ein anderer Vorteil der EJB sind die impliziten Dienste bezüglich der Ressourcen-, Zugriffs-, Sicherheits-, Transaktions- und Persistenzverwaltung. Diese bringen enorme Erleichterung bei der Realisierung der Web-basierten Anwendungen, weil sie einen großen Anteil an Implementierungsaufgaben (z.T. wiederkehrende) durch die Eintragung im Deployment descriptor übernehmen. Außerdem ermöglichen die impliziten Dienste den Entwicklern von Web-basierten Anwendungen sich voll auf die Implementierung der Anwendungslogik zu konzentrieren. Jenseits der technischen Umsetzung von Web-basierten Anwendungen unterstützt EJB auf der Organisationsebene im Entwicklungsprozess die Aufgabenverteilung durch einige vordefinierten Rollen.

Eine große Erleichterung bei der Erstellung von Web-basierten Anwendungen, insbesondere bei der Durchführung von einfachen (z.T. nachträglichen) Laufzeiteinstellungen und -anpassungen einer Anwendung, die mit geringem Aufwand im Deployment descriptor vorgenommen werden können, stellt der zentrale Laufzeiteinstellungsmechanismus von EJB dar. Eine neue Kompilierung der gesamten Anwendung wird dadurch vermieden. Darüber hinaus unterstützt EJB durch die Einbettung der EJB QL Abfragesprache die SQL-92 Spezifikation, trennt die Datenbank-

Implementierungslogik von der Anwendungslogik und abstrahiert dazu von dem eingesetztem Persistenzmedium.

Darüber hinaus ist zu sagen, dass die in EJB implementierte Anwendungslogik als Middelware in homogenen und heterogenen Umgebungen nicht nur für Internet-/Intranet-Lösungen, sondern auch für den Einsatz in verschiedenen spezifischen Produkt- und System-Lösungen, wie z.B. Großrechnereinsatz im Regierungswesen, Bankwesen, Versicherungswesen, etc., integrierbar ist. Die Tabelle 11 gibt zusammenfassend die wichtigsten Vorteile und Nachteile der EJB für die Entwicklung Web-basierter Anwendungen wieder.

Vorteile	***Nachteile***
- Kapselung - Trennung von Zuständigkeiten - Komponentenbasiert - Zentrales Management - Laufzeiteinstellungen - Implizite Dienste - Thinner Clients - Herstellerunabhängig	- Performanzkosten - Zerbrechlichkeit der EJB-Architektur - Erhöhte Komplexität - Höhere Lernkurve - Technology for Technology's Sake - Hohe Investitionskosten

Tabelle 11 - Resümee: Vorteile und Nachteile der EJB für die Entwicklung Web-basierter Anwendungen

Die markantesten Nachteile der EJB-Technologie sind das komplexe Dienstleistungsspektrum, Zerbrechlichkeit der EJB-Architektur sowie die hohe Lernkurve. Daher sollte man die EJB-Technologie in einem Kontext und Projektumfang einsetzten, wo das komplexe Dienstleistungsspektrum der EJB tatsächlich in Anspruch genommen werden kann. In der Web-Entwicklung sollten die EJB dort zum Einsatz kommen, wo die Vorteile effektiv genutzt werden können und die Nachteile überschaubar bleiben.

Eine Alternative zu EJB innerhalb der J2EE-Architektur wäre die Verwendung von Servlets/JSP für die Entwicklung Web-basierter Anwendungen. Aus meiner jetzigen Erfahrung mit der EJB-Spezifikation 2.0 haben folgende Fragen, die bei der Entscheidungsfindung für oder gegen EJB in einem bestimmten Kontext oder Projektumfang hilfreich sein können, einen besonderen Stellenwert:

- Wird die EJB Transaktions- und Persistenzunterstützung in einem Projekt gebraucht? – Können vorhandene Systeme (Legacy Systeme, Eigenentwicklungen, Datenbanken, etc.) oder Anwendungen sich in das implizite oder explizite Transaktions- und/oder Persistenzkonzept des EJB Containers einbinden?

- Wird das implizite Zugriffs- und Sicherheitsmanagement gebraucht? – Reichen die impliziten Sicherheitsmaßnahmen basierend auf „Role-driven access control“ aus?

- Sind die Performanzkosten nach der EJB-Vorgehensweise über mehrere Stellen oder Schichten berechenbar? – Ist ein einfache „round trip“ Netzwerkkommunikation zwischen Proxy, Webserver, Servlet-Engine (in der Interaktionslogik) zu der EJB-Anwendungslogik überschaubar?

- Sind die Entwickler befähigt mit der Komplexität der EJB-Spezifikation umzugehen? – Haben sie den Überblick beim Beschreiben des umfangreichen Deployment descriptors einer EJB Anwendung?

Fallen die Antworten auf diese Fragen teilweise negativ aus, sollte man die Servlets/JSP Technologie für die Entwicklung Web-basierter Anwendungen vorziehen. Aber auch hierbei muss darauf geachtet werden, dass eine Trennung in Interaktions-, Anwendungs- und Datenlogik vorgenommen wird.

Qualität der Re-Implementierung der Anwendungslogik des Web-basierten SfM Prototyps

Die Hauptfrage meiner Arbeit war: *„Inwiefern verbessert eine Re-Implementierung auf Basis von EJB die Qualität der Anwendungslogik des Web-basierten Serviceflow Management Prototyps im Hinblick auf ihren robusten und skalierbaren Einsatz?“*

Zur Beantwortung der Hauptfrage dieser Arbeit wurde als erstes die Schwachstellenuntersuchung des alten Web-basierten SfM Prototyps hinsichtlich einiger bedeutsamer Anforderungen und Kriterien an die Entwicklung Web-basierter Anwendungen (siehe Kapitel 2.1) durchgeführt. Diese beziehen sich auf die sechs softwaretechnischen Qualitätskriterien (s. Tabelle 12), die heutzutage immer größeren Einfluss auf die Web-basierte Anwendungsentwicklung haben.

Schwachstellen	***Einschränkungen durch***
Skalierbarkeit	Singleton Entwurfsmuster im Servicepointmanager
Robustheit	Umgang mit Original-Anwendungsmaterialien sowie hohe Gefahr inkonsistenter Zustände
Portabilität	Festlegung der Pfade im Java-Code sowie fehlende zentrale oder portable Verwaltung von Benutzerdaten
Sicherheit	Keine Gewährleistung von Authentifizierung sowie Zugriffskontrolle
Wiederverwendung	Keine Einschränkungen; Verwendung von Architektur- und Entwurfsmustern
Integration	Wenig Einschränkungen; Abhängig von der Hard- und Software Infrastruktur-Konstellation

Tabelle 12 - Resümee: Übersicht der Schwachstellenanalyse

Nächster Schritt war die Auseinandersetzung mit der EJB-Technologie. Die wichtigsten Bestandteile der EJB-Architektur wurden dabei vorgestellt und erläutert. Eine Re-

Implementierung der Anwendungslogik des Web-basierten SfM Prototyps mit EJB wurde anhand eines Fallbeispiels vorgenommen. Nach der Re-Implementierung wurde eine Bewertung nach softwaretechnischen Entwurfsprinzipien und Qualitätskriterien, die die Anforderungen und Kriterien für die Entwicklung Web-basierter Anwendungen in dieser Arbeit ausmachen, durchgeführt.

Die Tabelle 13 gibt einen Überblick über die an den EJB Web-basierten SfM Prototyp gestellten softwaretechnischen Entwurfsprinzipien und deren Erfüllung.

Entwurfsprinzip	***Erfüllt***
Kapselung	durch Trennung von Schnittstelle und Implementation sowie Einhaltung des Geheimnisprinzips
Trennung von Zuständigkeiten	durch Vorgaben der EJB-Architektur; klare Zuordnung von Aufgaben und Zuständigkeiten
Minimierte Kopplung	durch viele lose Einheiten mit spezialisierten Aufgaben
Verwendung von Architektur- und Entwurfsmustern	durch die EJB-Vorgehensweise; keine Verwendung von Desktop-Entwurfsmustern

Tabelle 13 - Resümee: Erfüllung allgemeiner softwaretechnischer Entwurfsprinzipien durch den EJB Web-basierten SfM Prototyp

- Das Prinzip der Kapselung bezieht sich auf die Beachtung des Geheimnisprinzips sowie die Trennung von Schnittstelle und Implementation. Das Geheimnisprinzip wurde durch die Implementierung der Anwendungslogik-Komponente in den fachlich motivierten Fachlichen Services, welche die innere Datenstruktur verbergen, eingehalten. Weiterhin wurde die Kapselung der Anwendungsdaten in den Entity Beans und in der Anwendungslogik bzw. Geschäftslogik in den Session Beans vorgenommen.

- Die Trennung von Zuständigkeiten innerhalb des EJB Web-basierten SfM Prototyps ist nach den Vorgaben der EJB-Architektur, die auf verschiedenen Ebenen eine klare Zuordnung von Aufgaben und Zuständigkeiten aufweist, implementiert worden. Auf der Ebene des Web-basierten SfM Prototyps wurde dieses Entwurfsprinzip durch die Unterteilung in mehrere Schichten eingehalten. In der Anwendungslogik wurde dies durch die Aufteilung der Verantwortlichkeit zwischen den Fachlichen Services und dem Integrationsservice erreicht.

- Die Kopplung der Komponenten der Anwendungslogik wurde durch die EJB-Vorgehensweise minimiert gehalten. Der Servicepointmanager ist die einzige Komponente in der Anwendungslogik, die das Wissen über einzelne zustandslose Fachliche Services besitzt. Im Integrationsservice wurden Anwendungs- und Hilfsmaterialien in Entity Beans aggregiert.

- Durch die konsequente Anwendung der EJB-Vorgehensweise, die auf eine Art und Weise ein Architekturmuster darstellt, wurde dem Entwurfsprinzip der Verwendung von Architektur- und Entwurfsmustern nachgegangen. Es bleibt die Frage offen, ob weitere Entwurfsmuster aus dem etablierten Desktopbereich angewandt werden könnten? (z.B. Business Delegate, Session Façade, Value Object oder Aggregate Entity nach [ADAT01], S. 501)

Die Tabelle 14 stellt Bewertungsergebnisse der Qualitätskriterien des EJB Web-basierten SfM Prototyps, mit dem Schwerpunkt auf der Anwendungslogik, dar.

Qualitätskriterium	***Erfüllt***
Skalierbarkeit	durch die EJB-Vorgehensweise sowie die Realisierung des Servicepointmanagers als stateful Session Bean
Robustheit	durch das implizite Transaktions- und Persistenzmanagement des EJB-Containers
Portabilität	durch die J2EE-Architektur sowie Java + XML
Sicherheit	durch die implizite Authentifizierung- und Zugriffskontrolle des EJB-Containers
Wiederverwendung	durch das EJB-Komponentenmodell und die schichtenbasierte Architektur des Prototyps
Integration	durch die Anlehnung an die J2EE-Architektur; integrierbar in jeder J2EE Umgebung

Tabelle 14 - Resümee: Erfüllung allgemeiner softwaretechnischer Qualitätskriterien durch den EJB Web-basierten SfM Prototyp

- Das Qualitätskriterium der Skalierbarkeit ist in dem Web-basierten SfM Prototyp durch die EJB-Vorgehensweise bei der Realisierung der einzelnen Schichten, insbesondere der Anwendungslogik, erfüllt worden. In der Anwendungslogik bzw. in den Fachlichen Services wurde der Servicepointmanager, der nicht von der Infrastruktur-Konstellation des Einsatzortes und der Anzahl der Java Virtual Machines abhängig ist, als stateful Session Bean implementiert,.

- Durch die Verwendung des impliziten Transaktions- und Persistenzmanagements in der Anwendungslogik wurde sichergestellt, dass der Web-basierte SfM Prototyp die Gefahr inkonsistenter Zustände ausschließt. Weitere implizite Dienste des EJB-Containers, wie Ressourcen-, Zugriffs- und Sicherheitsmanagement stellen dazu sicher, dass das Qualitätskriterium der Robustheit erfüllt wird.

- Die Anwendungslogik des Web-basierten SfM Prototyps ist in jede J2EE Umgebung portabel. Die Anwendungs- und Hilfsmaterialien des SfM Prototyps sind für den Einsatz in einer relationalen Datenbank modelliert worden. Das Austauschen von Materialien, vor allem der Servicefloats, zwischen den einzelnen Servicepoints bzw. Dienstleistungserbringern findet über zwei festgelegte FTP-Accounts: XML-In und XML-Out, in dem Servicefloats im XML-Format empfangen und versandt werden,

statt. Somit ist dem Qualitätskriterium der Portabilität im hohen Maße nachgegangen worden.

- Die Sicherheitsaspekte in der Realisierung des Web-basierten SfM Prototyps wurden durch die EJB-Vorgehensweise erfüllt. Die vom EJB-Container bereitgestellten Dienste hinsichtlich der Authentifizierung- und Zugriffskontrolle sowie die Möglichkeit der Zugriffsberechtigungsfestlegung für jede Gruppe durch den „Role-driven access control“ in dem Deployment descriptor haben dazu beigetragen, dass das Qualitätskriterium der Sicherheit vollständig umgesetzt wurde. Des weiteren wurde die Verschlüsselung der Servicefloats für den Beteiligten in dem gesamten Dienstleistungsflussaustausch eingeführt. Auf der Persistenzebene ist seitens des Datenbank-Managers sichergestellt worden, dass der Zugriff auf die Materialien nur den Zugangsberechtigten erlaubt wird.

- Durch die EJB-Vorgehensweise bei der Realisierung des Web-basierten SfM Prototyps wird dem Wiederverwendungs-Qualitätskriterium in hohem Maße nachgegangen. Die schichtenbasierte Architektur ermöglicht die Wiederverwendung und den Einsatz der einzelnen Schichten in jeder J2EE-Umgebung.

- Dem Qualitätskriterium der Integration des Web-basierten SfM Prototyps, besonders der Anwendungslogik, wurde durch die Anlehnung an die J2EE-Architektur umgesetzt. Einzige Voraussetzung für die Erfüllung dieses Kriteriums ist die Verwendung der J2EE-Architektur sowie die Bereitstellung einer relationalen Datenbank auf der Persistenzebene.

Das Ergebnis der Untersuchungen des EJB Web-basierten SfM Prototyps nach den softwaretechnischen Entwurfsprinzipien und Qualitätskriterien zeigt eine deutliche Verbesserung gegenüber dem alten Web-basierten SfM Prototyp. Dies ist vor allem durch die von EJB empfohlene Vorgehensweise erzielt worden. Die Qualität der Anwendungslogik wurde durch die Verwendung des impliziten Transaktions- und Persistenzmanagements des EJB-Containers im Hinblick auf ihren robusten und skalierbaren Einsatz signifikant gesteigert.

Das Testergebnis des gesamten EJB Web-basierten SfM Prototyps auf meiner Hard- und Software-Konstellation fiel für die implementierten Teile des Prototyps recht positiv aus. Es konnten keine besonderen Performanzeinbußen der EJB-Technologie sowie des Prototyps festgestellt werden. Das lag vor allem daran, dass der Prototyp auf einem Rechner, auf dem ein lokaler Application-Server mit lokaler Datenbank läuft, getestet wurde. Hierbei möchte ich noch erwähnen, dass neben der Implementierung der Fachlichen Services und des Integrationsservices auch die sorgfältige Beschreibung des Deployment descriptors einige Zeit in Anspruch nahm. Diese Aufgabe sollte in der Praxis nicht unterschätzt werden, weil eine mangelhafte Beschreibung des Deployment descriptors die Funktionstüchtigkeit der EJB-Anwendung stark einschränkt.

Ein weiteres Ergebnis dieser Arbeit bezieht sich auf die Untersuchung der Realisierung eines eGovernment-Bürgerprozessportals mit dem EJB-Komponentenmodell nach

einigen bedeutsamen Kriterien für den eGovernment-Bereich. Welche Unterstützung die EJB-Technologie für die Realisierung eines eGovernment-Bürgerprozessportals anbietet, ist in Tabelle 15 zusammengefasst.

Kriterium	***Ergebnisse***
Prozessportal	unterstützt durch das breite Dienstleistungsspektrum des EJB-Komponentenmodells
Authentifizierung- und Zugriffskontrolle	unterstützt durch die EJB-Container Authentifizierung und Zugriffskontrolle
Skalierbarkeit und Robustheit	unterstützt durch die EJB empfohlene automatische Persistenz- und Transaktionsverwaltung
Prozesspooling und Nebenläufigkeit	unterstützt durch das automatische Ressourcen- und Lebenszyklusmanagement seitens des EJB-Container
Mehrbenutzerfähigkeit	unterstützt durch die impliziten Dienste des EJB-Containers
Dynamische Modellierung und situative Anpassung	unterstützt durch die zentralen Einstellungsmöglichkeiten im Deployment descriptor
Wiederverwendbarkeit und Integration	unterstützt durch den Einsatz der J2EE-Architektur

Tabelle 15 - Resümee: Eignung von EJB für die Realisierung eines eGovernment-Bürgerprozessportals

Die entscheidenden Vorteile von EJB für die Realisierung eines eGovernment-Bürgerprozessportals sind hierbei die impliziten Dienstleistungen der EJB-Architektur, vor allem hinsichtlich des Transaktions-, Persistenz- und Ressourcenmanagements sowie einer zentralen Zugriffs- und Laufzeiteinstellungskontrolle im Deployment descriptor. Auch hier gelten die selben Einschränkungen und Nachteile von EJB, auf die in der allgemeinen Fragestellung eingegangen wurde.

4.2 Ausblick

In meiner Arbeit habe ich mich stark darauf konzentriert einen möglichst geradlinigen und stimmigen Weg bei der Beantwortung der Fragestellung, beginnend mit der Problemstellung über die Eignung von EJB für die Entwicklung Web-basierter Anwendungen bis hin zu einer konkreten Implementation des Web-basierten SfM Prototyps nach der EJB-Vorgehensweise, mit einer detaillierten Erläuterung und Erörterung der EJB-Technologie aufzuzeigen. Dabei sind einige Fragen aufgetreten, deren Betrachtung und Untersuchung den Rahmen meiner Diplomarbeit sprengen würde, jedoch als Grundlage für weitere Arbeiten in diesen Bereichen dienen könnten.

Die offenen Fragen zur Anwendbarkeit der EJB für die Entwicklung Web-basierter Anwendungen lassen sich wie folgt formulieren:

- Ist eine Entwicklung von neuen wissenschaftlichen Methoden zur Ermittlung verlässlicher Anforderungen und Kriterien an die Entwicklung Web-basierter Anwendungen nötig oder können Leitbilder und Ansätze aus der klassischen Softwareentwicklung übernommen werden? – Welche Auswirkungen können diese auf die von EJB empfohlene Vorgehensweise haben?

- Inwiefern können die Desktop-Entwurfsmuster bei der Entwicklung Web-basierter Anwendungen mit EJB eingesetzt werden? – Welche Auswirkungen können diese auf die EJB empfohlene Vorgehensweise haben bzw. wie hoch ist dabei die Gefahr der Zerbrechlichkeit der EJB-Architektur?

- Ist ein Vergleich der EJB-Technologie mit einer weiteren server-seitigen Technologie für die Entwicklung Web-basierter Anwendungen, insbesondere .NET Technologie von Microsoft hinsichtlich der Einhaltung der Entwurfsprinzipien und Qualitätskriterien notwendig?

- Ist der Einsatz der EJB in heterogenen System- und Anwendungslandschaften mit Clustering, Server Load Balancing, etc. zu empfehlen? – Hier insbesondere im Hinblick auf das Zusammenspiel mit den impliziten Dienstleistungen des EJB-Containers?

- Die Komplexität der Nutzung der impliziten Dienstleistungen des EJB-Containers wird auf die Beschreibung des Deployment descriptors reduziert. Inwiefern verändert oder beschränkt dies den Charakter der Plattform Web? – Welche Auswirkungen könnte dies dabei auf die Web-basierte Anwendungsentwicklung haben?

5 Literaturverzeichnis

[ADAT01] ***Rahim Adatia et al.***
Programmer to programmer: Professional EJB
Wrox Press Ltd. 2001

[BLANK01] ***Timmy Blank***
Prozessunterstützung für eGovernment-Services mit Java und XML am Beispiel von http://www.hamburg.de (Diplomarbeit, 2001)

[BLJEKL02] ***Wolf-Gideon Bleek, Martti Jeenicke, Ralf Klischewski***
e-Prototyping
In: EMISA-Forum, Heft 1, S. 11-18, 2002

[CODAET98] ***Coda, Francesco; Ghezzi, Carlo; Vigna, Goivanni; Garzotto, Franca***
Towards a Software Engineering Approach to Web Site Development. In: Proceedings of the Ninth International Workshop on Software Specification and Design : April 16 - 18, 1998, Ise-Shima, Japan, Seite 8 - 17.

[CONNOL00] ***Dan Connolly***
A little history of the World Wide Web
WWW-Konsortium, http://www.w3.org/History.html

[FIELDIN99] ***R. Fielding, et al.***
Hypertext Transfer Protocol – HTTP/1.1
WWW-Konsortium, http://www.w3.org/Protocols/Specs.html

[FLANAG98] ***David Flanagan***
Javascript: The Definitive Guide
O'Reilly & Associates, Juni 1998

[FLGRMA99] ***C. Floyd, G. Gryczan, J. Mack***
Einführung in die Softwaretechnik. Scriptum zur gleichnamigen Lehrveranstaltung, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, Universität Hamburg, 1999.

[FLOYD92] ***Christiane Floyd***
STEPS Projekthandbuch
Universität Hamburg, 1992

[GAEGRÄ00] ***Martin Gaedke, Guntram Gräf***
Development and Evolution of Web-Applications Using the WebComposition Process Model
LNCS 2016, p. 58 ff.

[GHJV96] ***E. Gamma, R. Helm, R. Johnson, J. Vlissides***
Entwurfsmuster – Elemente wiederverwendbarer objektorientierter
Addison-Wesley, 1996.

[GILESCH96] ***A. Girgensohn, A. Lee, K. Schlueter***
Experiences in Developing Applications Using the World Wide Web "Shell".
In: Hypertext'96, Seventh ACM Conference on Hypertext, pp. 246-255, 1996.

[GISLER00] ***Michael Gisler***
eGovernment – Eine Einführung
http://www.telematiktage.ch/pdf/referate/gisler.pdf

[GISLER01] ***Michael Gisler***
Electronic Government – mehr als eine Website.
In: DISP Nr. 144, S. 35. Zürich, 2001.
http://www.orl.arch.ethz.ch/disp/pdf/144/144_5.pdf

[GISSPA01] ***Michael Gisler, Dieter Spahni***
eGovernment. Eine Standortbestimmung
Bern, 2001

[GRAREU93] ***Jim Gray, Andreas Reuter***
Transaction processing: concepts and techniques
Morgan Kaufmann, 1993

[GRYCZ96] ***Guido Gryczan***
Prozessmuster zur Unterstützung kooperativer Tätigkeit.
Deutscher Universitätsverlag GmbH, 1996.

[HUNTER01] ***Jason Hunter***
Java Servlet Programming (2nd Edition)
O'Reilly 2001

[JEENICK01] ***Martti Jeenicke***
Antizipative Anforderungsermittlung als Voraussetzung für die partizipative Systementwicklung (Diplomarbeit, 2001)

[JEWELL01] ***Tyler Jewell***
Stateful Session EJBs: Beasts of Burden
http://www.oreillynet.com/pub/a/onjava/2001/10/02/ejb.html

[KAHLBR98] ***Bernd Kahlbrandt***
Software-Engineering: Objektorientierte Software-Entwicklung mit der Unified Modelling Language
Springer 1998

[KAOJ01] ***James Kao***
Developer's Guide to Building XML-based Web Services with the Java 2 Platform, Enterprise Edition (J2EE); June 2001
http://www.theserverside.com/resources/pdf/J2EE-WebServices-DevGuide.pdf

[KÄSTN00] ***Kerstin Kästner***
E-Government – Wege zur elektronischen Verwaltung der Zukunft
http://www.z-punkt.de/download/e-gov.pdf

[KLISCHE01] ***Ralf Klischewski***
Infrastructure for an e-Government Process Portal
In: Remenyi, D., Bannister, F. (ed.): European Conference on e-Government. MCIL, Reading, UK, 2001, pp. 233-245

[KLIWET00] ***Ralf Klischewski, Ingrid Wetzel***
Serviceflow Management
Informatik Spektrum, Band 23, Heft 1, Februar 2000

[KLIWETa01] ***Ralf Klischewski, Ingrid Wetzel***
Serviceflow Management für das organisationsübergreifende e-Government
In: Bauknecht, K. u.a. (Hg.): Informatik 2001. Wirtschaft und Wissenschaft in der Network Economy – Visionen und Wirklichkeit. Österreichische Computer Gesellschaft, Wien, 2001, S. 313-319

[KLIWETb01] ***Ralf Klischewski, Ingrid Wetzel***
XML-based Process Representation for e-Government Serviceflows
Schmid, B., et al. (ed.): Towards the E-Society: E-commerce, E-business, and E-government (I3E 2001, IFIP). Dordrecht: Kluwer, 2001, pp. 789-802

[KLIWET02] ***Ralf Klischewski, Ingrid Wetzel***
Serviceflow Management: Caring for the Citizen's Concern in Designing E-Government Transaction Processes
Proceedings of the 35th Hawaii International Conference on System Sciences - 2002

[KLWEBA01] ***Ralf Klischewski, Ingrid Wetzel, Ali Baharami***
Modeling Serviceflow
Godlevsky, M., Mayr, H. (ed.): Information Systems Technology and its Applications. Proceedings ISTA 2001 (June 13-15, 2001, Kharkiv, Ukraine). Bonn: German Informatics Society, Lecture Notes in Informatics, 2001, pp. 261-272

[LENK00] ***Klaus Lenk, Gudrun Klee-Kruse***
Multifunktionale Serviceläden: Ein Modellkonzept für die öffentliche Verwaltung im Internet-Zeitalter.
Rainer Bohn Verlag, Berlin 2000

[LIROWO02] ***Martin Lippert, Stefan Roock, Henning Wolf***
Software entwickeln mit eXtreme Programming
Erfahrungen aus der Praxis
dpunkt.verlag, 2002

[LIWOZÜ01] ***Martin Lippert, Henning Wolf, Heinz Züllighoven***
Domain Services for Multichannel Application Software
Proceedings of Hawaii International Conference on System Sciences 2001, HICSS 34, IEEE Computer Society, 2001

[MATHAP99] ***Vlada Metana & Mark Hapner***
SUN Enterprise Java Beans Specification, v1.1

[McLAU00] ***Brett McLaughlin***
Java and XML
O'Reilly & Associates, 2000.

[MODABO01] ***Rajiv Mordani, James Duncan Davidson, Scott Boag***
Java API for XML Processing
Sun Microsystems, Inc.

[MONSON01] ***Richard Monson-Haefel***
Enterprise Java Beans, Third Edition
O'Reilly & Associates, Inc. 2001

[MONSON00] ***Richard Monson-Haefel***
Enterprise Java Beans, Second Edition
O'Reilly & Associates, Inc. 2000

[MURUG01] ***San Murugesan, Yogesh Deshpande, Steve Hansen and Athula Ginige***
Web-Engineering: A New Descipline for Development of Web-Based Systems
LNCS 2016, p. 3 ff.

[NAWA99] ***Nambisan, S., Wang, Y.-M.***
Roadblocks to Web Technology Adoption? In: Communications of the ACM, 42 (1), January 1999, S. 98-101.

[OTTSCH00] ***Michael Otto, Norbert Schuler***
Fachliche Services: Geschäftslogik als Dienstleistung für verschiedene Benutzungsschnittstellen-Typen (Diplomarbeit, 2000)

[ROMAN02] ***Ed Roman***
Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition; Second Edition, Wiley, 2002

[ROMAN99] ***Ed Roman***
Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition; Wiley, 1999

[SCHEDL00] ***Kuno Schedler***
eGovernment und neue Servicequalität in der Verwaltung?
In: M. Gisler, D. Spahni: eGovernment – Eine Standortbestimmung
Haupt Verlag, November 2000

[SPAH00] ***Dieter Spahni***
eGovernment Begriffe / eGovernment als Standortfaktor
aus http://www.iwv.ch/ccegov/

[SUN99] ***Sun Microsystems Inc., 1999***
Enterprise JavaBeans Specification 1.1
http://java.sun.com/products/ejb/javadoc-1.1/

[SUN01] ***Sun Microsystems Inc., 2001***
Enterprise JavaBeans Specification 2.0
http://java.sun.com/products/ejb/javadoc-2_0-fr/

[SUNJSP01] ***Sun Microsystems Inc., 2001***
Tag Libraries
http://java.sun.com/products/jsp/taglibraries.html

[TURAU99] ***Volker Turau***
Techniken zur Realisierung Web-basierter Anwendungen
In: Informatik Spektrum 22, 3-12, Springer-Verlag, 1999

[VIGN99] ***Vignette Corporation. Version 5.0, 1999***
Vignette StoryServer: Overview

[WOLF00] ***Henning Wolf***
Java Server Pages und Fachliche Services
PJS Service 2000
Apcon Workplace Solutions

[ZÜLLIG98] ***Heinz Züllighoven***
Das objektorientierte Konstruktionshandbuch nach dem Werkzeug- & Material-Ansatz.
dpunkt.verlag, 1998.

6 Anhang

In diesem Abschnitt wird als erstes eine Übersicht über die EJB-Server bzw. Application-Server gegeben. Danach werden die wichtigsten Anwendungslogik-Komponenten des EJB Web-basierten SfM Prototyps aufgeführt. Dabei werden die Bestandteile der Fachlichen Services und des Integrationsservices präsentiert. Danach wird die Tabellenstruktur der verwendeten Anwendungs- und Hilfsmaterialien des EJB Web-basierten SfM Prototyps dargestellt.

6.1 EJB-Server-Übersicht

EJB-Server gibt es inzwischen von verschiedenen kommerziellen und freien Herstellern. Alle implementieren entweder die EJB-Spezifikation Version 1.1, 2.0 oder beide. Mittlerweile setzt die Softwareindustrie, um die Komplexität des Einsatzes von EJB-Anwendungen zu reduzieren, auf die Application-Server. Sie bestehen oftmals aus einer Kombination von mehreren verschiedenen Diensten und Technologien, wie z.B. Web-Server, Object Request Broker (ORB), Message-Oriented Middleware (MOM), Datenbanken, etc. Enterprise JavaBeans setzt auf Component Transaction Monitors (CTM)[30], welche die meisten Application-Server für unternehmerische Anwendungen einsetzen. CTM stellt die notwendige Infrastruktur für die automatische Verwaltung von Transaktionen, verteilten Objekten, Nebenläufigkeit, Sicherheit, Persistenz und Ressourcen sowie asynchronem Messaging zur Verfügung. In Enterprise JavaBeans wird dies durch die Integration der J2EE Technologie umgesetzt.

Wesentliche Unterschiede zwischen den jeweiligen EJB-Servern bzw. Application-Servern (siehe Abbildung 6) gibt es nicht, da sie allesamt die gleiche EJB-Spezifikation implementieren, aber man differenziert zwischen Zusatzmöglichkeiten, die der eine oder andere Hersteller anbietet. Vor allem achtet man auf die Geschwindigkeit und Skalierbarkeit des EJB-Containers. Darüber hinaus bieten viele Hersteller noch die Entwicklungsumgebung dazu an, was einige Vorteile und Erleichterung für die Entwicklung und den Einsatz von EJB mit sich bringt. Zusätzlich stellen viele Hersteller noch diverse Tools für Administration, Überwachung oder Debugging von EJB-Anwendungen zur Verfügung.

Da der EJB-Server- bzw. Application-Server-Markt ständig wächst, werden hier einige aktuelle freie und kommerzielle Produkte aufgelistet:

Freie:
- J2EE Reference Implementation
- JOnAS (http://www.evidian.com/jonas)
- Jboss (http://www.jboss.org)
- andere ...

[30] CTMs evolved as a hybrid of traditional Transaction Processing (TP) and Object Request Broker (ORB) technologies. They implement robust server-side component models that make it easier for developers to create, use, and deploy business systems. [MONSON01]

Kommerzielle:
- BEA WebLogic (http://www.bea.com)
- Borland Enterprise Server (http://www.borland.com/bes)
- IBM Websphere (http://www-4.ibm.com/software/webservers/appserv)
- Oracle 9i Application Server (http://www.oracle.com/ip/deploy/ias/index.html)
- Pramati (http://www.pramati.com)
- Orion Application Server (http://www.orionserver.com)
- andere ...

Aktuelle Informationen über EJB- und Application-Server findet man unter:
http://www.flashline.com/components/appservermatrix.jsp oder
http://www.techmetrix.com/trendmarkers/techmetrixasd.php

6.2 Komponenten der implementierten Anwendungslogik

Fachliche Services: Einzelne Bestandteile der Fachlichen Services mit deren Schnittstellen.

Session Beans	Bestandteile
Servicepointmanager	ServicepointmanagerBean, ServicepointmanagerRemote, ServicepointmanagerHomeRemote, ServicepointmanagerLocal, ServicepointmanagerHomeLocal
FrontOffice	FrontOfficeBean, FrontOfficeRemote, FrontOfficeHomeRemote, FrontOfficeLocal, FrontOfficeHomeLocal
BackOffice	BackOfficeBean, BackOfficeRemote, BackOfficeHomeRemote, BackOfficeLocal, BackOfficeHomeLocal
ReportingAndTracking	ReportingAndTrackingBean, ReportingAndTrackingRemote, ReportingAndTrackingHomeRemote, ReportingAndTrackingLocal, ReportingAndTrackingHomeLocal
Archive	ArchiveBean, ArchiveRemote, ArchiveHomeRemote, ArchiveLocal, ArchiveHomeLocal
XMLConverter	XMLConverterBean, XMLConverterRemote, XMLConverterHomeRemote, XMLConverterLocal, XMLConverterHomeLocal
SendReceive	SendReceiveBean, SendReceiveRemote, SendReceiveHomeRemote, SendReceiveLocal, SendReceiveHomeLocal

Tabelle 16 - Bestandteile des Fachlichen Services

Integrationsservice: Einzelne Bestandteile der Anwendungs- und Hilfmaterialien mit deren Schnittstellen.

Entity Beans	Bestandteile
Servicefloat	ServicefloatBean, ServicefloatRemote, ServicefloatHomeRemote, ServicefloatLocal, ServicefloatHomeLocal
Servicepointscript	ServicepointscriptBean, ServicepointscriptRemote, ServicepointscriptHomeRemote, ServicepointscriptLocal, ServicepointscriptHomeLocal
Customer	CustomerBean, CustomerRemote, CustomerHomeRemote, CustomerLocal, CustomerHomeLocal
Editor	EditorBean, EditorRemote, EditorHomeRemote, EditorLocal, EditorHomeLocal
Provider	ProviderBean, ProviderRemote, ProviderHomeRemote, ProviderLocal, ProviderHomeLocal
Address	AddressBean, AddressLocal, AddressHomeLocal
XML-In	XMLInBean, XMLInLocal, XMLInHomeLocal
XML-Out	XMLOutBean, XMLOutLocal, XMLOutHomeLocal

Tabelle 17 - Bestandteile der Materialien als Entity Beans

6.3 Die Datenbanktabellenstruktur der Anwendungsmaterialien

Im folgenden werden die Datenbanktabellenstruktur der verwendeten Anwendungs- und Hilfsmaterialien des Web-basierten SfM Prototyps aufgeführt.

SERVICEFLOAT					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
sf_id	Number	Unsigned	No		auto_increment
sf_name	Varchar2		No		
sf_type	Varchar2		No		
citizen_id	Number	Unsigned	No		foreign_key
editor_id	Number	Unsigned	No		foreign_key
sf_create	Date		No		
sf_last_changed	Date		Yes		
editor_last_changed_id	Number	Unsigned	Yes		foreign_key
editor_history_ids	Object		Yes		
current_sp_id	Number	Unsigned	No		foregin_key
Scheduled_sp_ids	Object		Yes		
passed_sp_ids	Object		Yes		
precon_sp_ids	Object		Yes		
postcon_sp_ids	Object		Yes		
sf_documents	Object		Yes		

Tabelle 18 - Anwendungsmaterial: Servicefloat

CITIZEN					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
id	Number	Unsigned	No		auto_increment
firstname	Varchar2		No		
lastname	Varchar2		No		
gender	Varchar2		No		
citizen_account	Varchar2		No		
password	Varchar2		No		
address_id	Number	Unsigned	No		foreign_key

Tabelle 19 - Anwendungsmaterial: Citizen

SERVICEPOINTSCRIPT					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
sp_id	Number	Unsigned	No		auto_increment
sp_name	Varchar2		No		
sp_type	Varchar2		No		
citizen_id	Number	Unsigned	No		foreign_key
editor_id	Number	Unsigned	No		foreign_key
provider_id	Number	Unsigned	No		foreign_key
sp_create	Date		No		
sp_last_changed	Date		Yes		
editor_last_changed_id	Number	Unsigned	Yes		foreign_key
editor _history_ids	Object		Yes		
current_act_id	Number	Unsigned	No		foregin_key
scheduled_act_ids	Object		Yes		
passed_act_ids	Object		Yes		
precon_act_ids	Object		Yes		
postcon_act_ids	Object		Yes		
sp_documents_ids	Object		Yes		

Tabelle 20 - Anwendungsmaterial: Servicefloat

EDITOR					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
id	Number	Unsigned	No		auto_increment
firstname	Varchar2		No		
lastname	Varchar2		No		
gender	Varchar2		No		
editor_account	Varchar2		No		
password	Varchar2		No		
address_id	Number	Unsigned	No		foreign_key
sp_id	Number		No		foreign_key
sp_ids	Number		Yes		

Tabelle 21 - Anwendungsmaterial: Editor

PROVIDER					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
id	Number	Unsigned	No		auto_increment
firstname	Varchar2		No		
lastname	Varchar2		No		
provider_account	Varchar2		No		
password	Varchar2		No		
company	Varchar2		No		
address_id	Number	Unsigned	No		foreign_key

Tabelle 22 - Anwendungsmaterial: Provider

DOCUMENT					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
doc_id	Number	Unsigned	No		auto_increment
doc_type	Number		No		
doc_header	Varchar2		No		
doc_create	Date		No		
doc_editor_id	Number	Unsigned	No		foreign_key
doc_last_changed	Date		Yes		
doc_editor_last_changed_id	Number	Unsigned	Yes		foreign_key
doc_history_ids	Object		Yes		
doc_body	Blob		No		

Tabelle 23 - Anwendungsmaterial: Document

ADDRESS					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
id	Number	Unsigned	No		auto_increment
street	Varchar2		No		
streetnumber	Number	Unsigned	No		
zip	Number		No		
city	Varchar2		No		
province	Varchar2		Yes		
country	Varchar2		No		
email	Varchar2		Yes		
telephone	Number		Yes		
telefax	Number		Yes		

Tabelle 24 - Anwendungsmaterial: Address

XML-In					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
in_id	Number	Unsigned	No		auto_increment
in_timestamp	Date		No		
in_provider_id	Number	Unsigned	Yes		foreign_key
in_body	Blob		No		
in_received	Number		No		

Tabelle 25 - Hilfsmaterial: XML-In

XML-Out					
Field	**Type**	**Attributes**	**Null**	**Default**	**Extra**
out_id	Number	Unsigned	No		auto_increment
out_timestamp	Date		No		
out_provider_id	Number	Unsigned	Yes		foreign_key
out_body	Blob		No		
out_send	Number		No		

Tabelle 26 - Hilfsmaterial: XML-Out