Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Mykhaylo Kabalkin

Migration einer Rich Client Applikation auf Code on Demand Technologie

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Mykhaylo Kabalkin

Migration einer Rich Client Applikation auf Code on Demand Technologie

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Kai von Luck
Zweitgutachter : Prof. Dr. Gunter Klemke

Abgegeben am 15. Februar 2008

Mykhaylo Kabalkin

Thema der Masterarbeit

Migration einer Rich Client Applikation auf Code on Demand Technologie

Stichworte

Code on Demand, Web 2.0, AJAX, Rich Internet Application

Kurzzusammenfassung

Im Rahmen dieser Arbeit beschäftigt sich der Autor mit der Migration einer Rich Client Applikation auf Code on Demand Technologie. Eine Migration auf Code on Demand Technologie bedeutet hier nicht nur die Entwicklung einer Web Anwendung, sondern auch die Untersuchung der folgenden Fragen: Mit welcher Technologie und inwiefern kann die Migration einer bestehenden Rich Client Geschäftsanwendung durchgeführt werden? Welche der existierenden Frameworks sind für die Migration geeignet? Was hat ein Entwickler mit dem aktuellen Kenntnisstand über Softwareentwicklung und Design zu erledigen, um Geschäftsanwendungen auf Code on Demand Technologie mit dem geringeren Aufwand migrieren zu können? Was ändert sich für einen Nutzer durch den Einsatz einer Web Anwendung anstelle der existierenden Rich Client Anwendung? Welche Vor- bzw. Nachteile bringt eine Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie für ein Unternehmen? Wie kann das „Change and Configuration Management“ Problem reduziert werden?
Die oben aufgelisteten Fragen werden von dem Autor dieser Arbeit basierend auf der durchgeführten Migration einer Beispiel Rich Client Anwendung beantwortet.

Mykhaylo Kabalkin

Title of the paper

Migration of a Rich Client Application to the Code on Demand technology

Keywords

Code on Demand, Web 2.0, AJAX, Rich Internet Application

Abstract

In context of this thesis the author carried out a migration of a Rich Client Application to the Code on Demand technology. The migration means not only a development of a new web application, but also performing an investigation to answer the following questions. Using which technology and to what extent can the migration of an existing Rich Client business application be carried out? What does a developer with current state of knowledge about software development and design have to do to migrate the business applications with lower effort to the Code on Demand Technology? What changes will a user perceive by using a web application instead of the existing rich client application? Which advantages or disadvantages does a migration of an existing Rich Client Application on code on Demand technology bring for a company? How can the "'Change and Configuration Management"' problem be reduced?
Based on an executed migration of a sample Rich Clien application the author of this thesis is answering the questions listed above.

Danksagung

Mein herzlicher Dank gilt Herrn Prof. Dr. Kai von Luck und Herrn Prof. Dr. Gunter Klemke für ihre hervorragende Unterstützung und vielseitigen fachlichen Rat.

Mein besonderer Dank gilt Herrn Dr. Thomas Koch und in seinem Namen der Firma Atlantec Enterprise Solutions GmbH für die interessante Aufgabenstellung und Ratschläge.

Abschließend möchte ich mich bei meiner Familie und meinen Freunden für ihre Unterstützung während dieser Arbeit und meines ganzen Studiums bedanken.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation und Problematik

Seit längerer Zeit wird die Software von Menschen für das Erledigen unterschiedlicher Aufgaben benutzt. Es existieren verschiedene Software-Typen. Einige Software ist für einfache Endbenutzer entworfen. Solche Software kann man bei einem oder anderem Laden oder einem Softwarehersteller kaufen oder sogar als Freeware-Software herunterladen. Zu dem anderen Software-Typen gehören Geschäftsanwendungen. Die Geschäftsanwendungen, wie z.B. Ingenieur-Software, werden oft entweder erst auf einen Auftrag entworfen oder den individuellen Bedürfnissen der Kunden angepasst.

Meistens sind solche Anwendungen als Desktop Anwendungen entworfen. Sie müssen auf allen Rechnern, auf denen Mitarbeiter mit der Software arbeiten sollen, installiert und dann konfiguriert werden. Dies bedeutet schon zusätzlichen Aufwand.

Die heutige Geschäftswelt setzt hohe Mobilität voraus. Viele Mitarbeiter reisen um die Welt herum und erledigen ihre Aufgabe direkt bei den Kunden vor Ort. Sie haben auf ihren Laptops alle notwendige Software vorinstalliert und können sie natürlich verwenden. Was geschieht aber, wenn der Laptop eines Mitarbeiters während seiner Geschäftsreise wegen eines Hardware-, Software-Fehlers oder eines Virus funktionsunfähig wird? Wie kann der Mitarbeiter seine Arbeit erledigen? Heute ist es gar kein Problem, einen neuen Laptop zu bekommen, wird wahrscheinlich gesagt. Entweder wird der Mitarbeiter einen Rechner von dem Kunden zur Verfügung gestellt bekommen oder bei einem Computerhersteller einen Neuen kaufen. Der Autor dieser Arbeit stimmt hier auch zu. Der Rechner selbst ist kein Problem. Wie bekommt aber unser Ingenieur die für seine Arbeit notwendige Software. Es würde wahrscheinlich wieder gesagt werden, kein Problem, die Software kann heruntergeladen und installiert werden. An dieser Stelle muss der Autor dieser Arbeit dieser Aussage teilweise widersprechen. Das Herunterladen der Software ist kein Problem, wenn sie im Netz zu Verfügung steht oder für den Mitarbeiter gestellt wird. Da wir hier über mächtige Client-Software (z.B. für Ingenieur-Anwendungen) sprechen, kann die Installation der Anwendung kompliziert sein. Die reine Installation ist jedoch nicht der aufwändigste Teil bei solchen Anwendungen, dies kann von einem Mitarbeiter erledigt werden. Nach der erfolgreichen Installation muss die Anwendung noch konfiguriert werden. Die Konfiguration solcher

Anwendungen ist oft ein sehr komplizierter Prozess, der nur von einem Administrator erledigt werden kann. Die Mitarbeiter können mit der Software umgehen, kennen aber oft nur Funktionalität, die für das Erledigen ihrer Arbeit notwendig ist.

Die im Beispiel beschriebenen Probleme werden in der Theorie als das „Change and Configuration Management“ Problem genannt. Die Installation, die Konfiguration und das Update der Software auf eine neue Version sind wichtige Bestandteile des „Change and Configuration Management“ Problems. Seit mehreren Jahren beschäftigen sich viele Wissenschaftler mit dieser Problematik. Noch im Jahr 1997 bezeichnet Dr. Gregor Joeris das Change Management als das Kernproblem der Softwareentwicklung [vgl. Joeris (1997)].

Der Softwareentwicklungsprozess bleibt nicht stehen. Es werden neue Methoden und Konzepte ausgearbeitet. Trotzdem ist der Autor dieser Arbeit überzeugt, dass viele Mitarbeiter/Geschäftsleute ohne zusätzliche Hilfe die Installation und die Konfiguration der Geschäftsanwendungen nicht erledigen können. Dies bedeutet für ein Unternehmen Zeit- und Geldverschwendung, da der Mitarbeiter nicht weiter bei dem Kunden vor Ort arbeiten kann. Beim Auftreten von solchen Problemen muss man auch in seinem Büro auch auf die Unterstützung von Administratoren oder Support-Mitarbeiter erst warten.

Die Geschichte der Softwareentwicklung lief von Mainframes mit einfachen Clients bis zu intelligenten Clients, die vollständige Business Logik enthalten und installiert und konfiguriert werden müssen. Heutzutage gibt es eine Bewegung, so wenig wie möglich auf der Client-Seite zu installieren. Die Renaissance der Rechenzentren ermöglicht, die Anwendungen mit einfacher Wartbarkeit und hoher Verfügbarkeit serverseitig zur Verfügung zu stellen. Die Tendenz geht auch die Richtung der Rich Internet Application (RIA) und der Code on Demand Technologien. Mittels der Code on Demand Technologie kann hohe Verfügbarkeit der Anwendungen clientseitig erreicht werden. Dadurch soll auch die oben beschriebene Problematik des Change and Configuration Managements verringert werden. Diese Problematik gilt als das Metapher für diese Arbeit.

1.2 Zielsetzung

Als Lösung der oben beschriebenen Probleme [1.1] sieht der Autor die Möglichkeit, den Mitarbeitern die Geschäftsanwendungen über das Web zur Verfügung zu stellen. Eine Web-Anwendung muss auch konfiguriert werden. Da die Installation und die Konfiguration der Geschäftsanwendungen von einem Server-Administrator (Experten) einmalig erledigt werden kann, kann man auf den unnötigen Aufwand durch Mitarbeiter in diesem Fall verzichten. Dadurch wird ein Unternehmen Zeit, somit auch Geld, gewinnen, weil die Mitarbeiter beim Auftreten von oben beschriebenen Problemen die Geschäftsanwendung nicht neu installieren und konfigurieren brauchen. Dabei ist es gleichgültig, ob ein Mitarbeiter in seinem Büro

oder bei einem Kunden auf der anderen Seite der Welt mit der Geschäftsanwendung arbeitet soll.

Ziel dieser Arbeit ist, zu untersuchen, mit welcher Technologie und in wie fern bestehende Geschäftsanwendungen auf Code on Demand Technologie migriert werden können. Die Ergebnisse der Untersuchungen sollen anhand eines Beispiels bestätigt werden, an dem die Migration einer bestehenden Geschäftsanwendung durchgeführt werden soll. Dabei soll die Funktionalität der Geschäftsanwendung bestehen bleiben und heutiger Stand von Softwareentwicklung und Design verwendet werden. Während der Untersuchung ist es eine weitere Frage zu beantworten, die der Autor für wichtig hält. Welche der existierenden Frameworks sind für solche Aufgaben geeignet? Zu dieser Frage gehört nicht nur die Frage der Funktionalität, sondern auch die Erlernbarkeit der Frameworks. Mit anderen Worten ist es sehr wichtig, festzustellen, was ein Entwickler mit dem aktuellen Kenntnisstand über Softwareentwicklung und Design zu erledigen hat, um Geschäftsanwendungen auf Code on Demand Technologie mit dem geringeren Aufwand migrieren zu können.

1.3 Inhaltlicher Aufbau der Arbeit

In dieser Arbeit wird das Thema der Migration einer Rich Client Applikation auf Code on Demand Technologie ausgearbeitet. In diesem Kapitel [1] wurden die Motivation des Autors und die Problematik vorgestellt und das Ziel dieser Arbeit definiert.

Im Anschluss an diese Einleitung werden im Kapitel [2] die für das Verstehen der Arbeit erforderlichen Grundlagen erläutert. Das Kapitel behandelt die Themen: Rich Client Application, Rich Internet Application und Code on Demand Technologie. Der Begriff Web 2.0 und das Asynchronous JavaScript and XML Konzept werden aufgezeigt. Es werden einige Java-basierte AJAX Frameworks (Google Web Toolkit, Echo Framework) vorgestellt.

Im Kapitel [3] werden anhand der Vorgaben des Unternehmens die Anforderungen definiert. Es wird zwischen den funktionalen und nicht funktionalen Anforderungen unterschieden. Es werden Anwendungsfälle für das nach außen sichtbare Verhalten der Anwendung eingeführt und einige detailliert beschrieben. Als nicht funktionale Anforderungen werden solche wie die Performance, die Fehlertoleranz, die Portierbarkeit der Anwendung behandelt. Die Anforderungen bezüglich des Entwicklungsaufwandes werden festgelegt.

Das Design und der Entwurf des Softwaresystems sowie getroffene Entscheidungen werden im Kapitel [4] besprochen. Es werden unterschiedliche Architekturstile gezeigt. Der Einsatz von Web Deployment wird diskutiert. Die Architektur des gesamten Systems und die mögliche Trennung der Funktionalität zwischen der Client Seite und der Server Seite werden vorgestellt.

Im Kapitel [5] werden die gefundenen Ergebnisse bewertet. Es wird die Akzeptanz des Einsatzes durch Unternehmen, Entwickler und Nutzer diskutiert. Was ändert sich für einen Nutzer durch den Einsatz einer Web Anwendung statt der existierenden Rich Client Anwendung? Worauf soll ein Entwickler bei dem Entwurf und der Entwicklung einer auf Code on Demand Technologie basierten Web Anwendung achten? Welche Vor- und/oder Nachteile bringt die Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie für ein Unternehmen? Solche Fragen werden im Evaluierungskapitel beantwortet.

Zum Schluss erfolgt im Kapitel [6] eine Zusammenfassung der Arbeit und ein Ausblick in die Zukunft.

2 Grundlagen

In diesem Kapitel werden die Grundlagen vorgestellt, die zum weiteren Verständnis der Arbeit notwendig sind. Zunächst werden Rich Client Applikationen kurz erwähnt. Danach werden die Eigenschaften der Rich Internet Applikationen skizziert. Die für die Migration der existierenden Anwendung notwendige Grundlagen der Code on Demand Technologie, inklusive den Begriff Web 2.0 und das Asynchronous JavaScript and XML (AJAX) Konzept, werden aufgezeigt. Anschließend werden einige Java-basierte Ajax Framework dargestellt und anhand von Vor- und Nachteilen verglichen.

2.1 Rich Client Applikation

Rich Client Applikationen werden als Desktop-Anwendungen bezeichnet. Eine Desktop-Anwendung wird immer lokal auf dem Client-Rechner installiert und verfügt über ein eigenes User Interface. Die Architekturarten einer Desktop-Anwendung werden im Kapitel [4.1] beschrieben. Rich Client Anwendungen sind meistens schwergewichtig und beinhalten den größten Teil der Anwendungslogik auf der Client Seite.

Eine Rich Client Anwendung, wie schon oben erwähnt wurde, setzt eine lokale Installation voraus. Für die Nutzung müssen Rich Client Anwendungen oft konfiguriert werden. Die Installations- und Konfigurationsprozesse sind in größeren Unternehmen technisch problematisch und teuer. Sie führen oft zu „Change and Configuration Management“ Problemen. Dies ist ein großer Nachteil der Rich Client Anwendungen.

Der Autor dieser Arbeit geht davon aus, dass der Leser eigene Erfahrung in Design und Entwicklung von Rich Client Anwendungen hat. Aus diesem Grund werden solche Anwendungen an dieser Stelle nicht weiter diskutiert.

2.2 Rich Internet Applikation

Rich Internet Applikationen (RIA) sind Web-basierte Anwendungen, die eine interaktive Benutzeroberfläche bieten und deren Charakteristiken und Funktionalität mit Rich Desktop Anwendungen vergleichbar sind.

Macromedia[1] definiert Rich Internet Application als eine Vereinigung der besten Funktionalität der Benutzerschnittstelle der Desktop Software mit der breiten Reichweite und preisgünstiger Verteilung von Web Anwendungen und der besten interaktiven, multimedialen Kommunikation [vgl. Duhl (2003)]. Die Abbildung [2.1] veranschaulicht diese Definition.

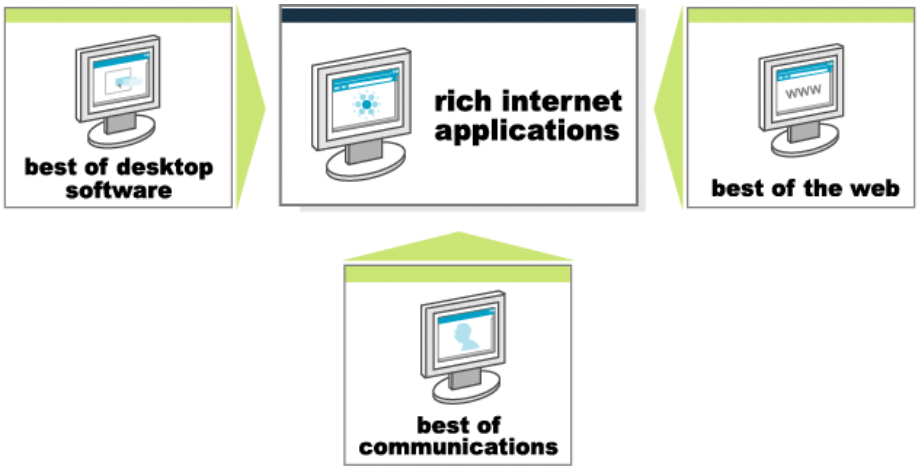

Abbildung 2.1: Rich Internet Application verknüpft das Beste aus Desktop, Web, und Kommunikation. Quelle: [Duhl (2003)]

Typische Rich Internet Applikationen laufen innerhalb eines Web Browsers. Es kann auf sie über bestimmte Internet-Techniken zugegriffen werden. Sie fordern keine lokale Installation und laufen in einer gesicherten Umgebung (Sandbox genannt) ab. Zusätzlich funktionieren RIAs bei Bedarf auch eigenständig auf portablen Rechnern und können von Außendienst-Mitarbeitern im offline-Betrieb eingesetzt werden. Rich Internet Applikationen interagieren mit dem Benutzer.

Rich Internet Applikationen sind zwar viel größer als HTML-Anwendungen, durch ihre effiziente Übertragung von Layoutinforamtionen sowie Streaming- und Komprimierungsmethoden sind sie aber weitaus leistungsfähiger. Außerdem können sie interaktive Verhaltensweisen und Multimedia-Funktionen bieten, wie sie bislang nur von Desktop Anwendungen bekannt waren [vgl. Eytorff u. a. (2007)].

[1] Macromedia wurde von Adobe System übernommen. www.adobe.com ; Zugriffsdatum: 30.01.2008

Durch die zentrale Datenhaltung werden Backup und Distribution neuer Software oder neuer Version von existierender Software erleichtert. Die RIAs sorgen für eine bessere Balance zwischen dem Server und einem Client. Die schnellere Antwortzeiten werden durch die asynchrone Kommunikation und die reduzierte Anzahl der Serveranfragen erreicht. Dadurch wird auch die Netzwerkbelastung reduziert.

2.3 Code on Demand Technologie

Unter Code on Demand versteht man Technologien, die ausführbare Software (Code) von einem Server zu einem Client auf der Client Anfrage übermitteln. In diesem Fall wird der Code auf der Client Seite ausgeführt.

Code on Demand Technologie ist eine spezielle Art von Mobile Code, der in [Carzaniga (17–23 May 1997)], wie folgt definiert ist:

> The capability to reconfigure dynamically, at run-time, the binding between the software components of the application and their physical location within a computer network.

An dieser Stelle werden der Begriff Web 2.0 und das Asynchronous JavaScript and XML (AJAX) Konzept vorgestellt.

2.3.1 Web 2.0

Der Begriff Web 2.0 kennzeichnet die Anwendungen, die das Internet als technische Plattform nutzen, auf der die Programme und die benutzergenerierten Inhalte zur Verfügung gestellt werden können. Das Konzept „Web 2.0“, das mit einem Brainstorming zwischen Tim O´Reilly und MediaLive International begann, ist ein Begriff einer Reihe interaktiver und kollaborativer Phänomene des World Wide Webs, den durch Tim O´Reilly im Jahr 2005 geprägt und dokumentiert wurde [vgl. O'Reilly (2005)]. Zu diesem Zeitpunkt war die Bedeutung des Konzeptes noch unklar. Der Begriff „Web 2.0“ deutete zunächst einfach an, dass es neue Anwendungen gibt (oder geben wird), die sich von der ersten Generation der Web-Anwendungen unterscheiden.

Auf der ersten Web 2.0-Konferenz[2] im Oktober 2004 nannten Tim O´Reilly und John Battelle eine erste Liste von Charakteristiken des „Web 2.0“ Konzeptes. Basierend auf das Dokument [vgl. O'Reilly (2005)] von Tim O´Reilly werden diese Charakteristiken hier kurz vorgestellt.

[2] http://www.web2con.com/web2con ; Zugriffsdatum: 30.01.2008

Das Web als Plattform

Das Web wird im „Web 2.0“-Konzept als eine Plattform anstatt des lokalen Rechners dargestellt. Das bedeutet, dass auf die Anwendungen von jedem Endgerät, das an das Internet angeschlossen ist und über einen Web Browser verfügt, zugegriffen werden kann und eine Installation der Anwendungen auf dem Endgerät nicht notwendig ist. Dadurch steht auch jede neue Version automatisch zur Verfügung. Das „Change and Configuration Management“ Problem kann dadurch reduziert werden.

Nutzung kollektiver Intelligenz

Das zentrale Prinzip hinter dem Erfolg der Web 2.0 Ära ist die Nutzung der kollektiven Intelligenz. Den Anwendern wird die Möglichkeit geboten, eigene Inhalte zu erzeugen oder bestehende Inhalte zu erweitern. Die gemeinsame Nutzung der auf der Plattform vorhandenen Inhalte und ihre gegenseitige Bezüge begründen Beziehungen zwischen den Benutzern.

Basierend auf der ungewöhnlichen Idee, dass jeder Eintrag von jedem Internet-User bearbeitet werden kann, baut Wikipedia[3] eine Online-Enzyklopädie. Viele Unternehmen setzen die Wikipedia-Idee ein, indem eine interne Wissensdatenbank gebaut wird, in die alle Mitarbeiter ihres Wissen publizieren können. Amazon[4] dagegen baut so eine Wissensdatenbank basierend auf dem Wissen der Kunden. Den Amazon-Kunden wird die Möglichkeit angeboten, Erfahrungsberichte zu den angebotenen Produkten zu erstellen. Auch weitere Dienstanbieter wie del.icio.us social bookmarking[5] und Flickr[6] basieren ihre Anwendungen auf die Nutzung der kollektiven Intelligenz der Kunden.

Durch Nutzerbeteiligung sind Netzwerk-Effekte der Schlüssel zur Marktdominanz in der Web 2.0 Ära [vgl. O’Reilly (2005)].

Daten als nächstes „Intel inside“

Aufgebaute Wissensdatenbanken beinhalten einzigartige Daten der Unternehmen. Ein wichtiges Merkmal der Web 2.0 Anwendungen ist nicht nur das Vorhalten solcher Daten, sondern auch die Möglichkeit diese zu verknüpfen und zu erweitern. Da Web 2.0 und die meisten Anwendungen auf Open Source-Software basieren, und durch offene Schnittstellen (APIs) definiert sind, können die Angebote und Daten miteinander mittels so genannten Mashups kombiniert werden. Mashups sind Web Anwendungen, die Informationen von unterschiedlichen

[3] http://www.wikipedia.org ; Zugriffsdatum: 30.01.2008
[4] http://www.amazon.com ; Zugriffsdatum: 30.01.2008
[5] http://del.icio.us ; Zugriffsdatum: 30.01.2008
[6] http://www.flickr.com ; Zugriffsdatum: 30.01.2008

Quellen mittels einfaches Web APIs (z.B. Web Services) verknüpfen können [vgl. Tatemura u. a. (2007)].

Software ohne Lebenszyklus

Die Web 2.0 Anwendungen werden nicht als Produkt (wie Desktop Anwendungen) ausgeliefert, sondern als ein Dienst angeboten, der ständig weiterentwickelt werden kann. Sobald die Entwicklung neuer Funktionalität abgeschlossen wird, steht sie umgehend den Nutzern zur Verfügung. Beseitigte Fehler sind umgehend nicht mehr im System vorhanden. Teile der Anwendung können live auf ihre Akzeptanz untersucht werden. Sollte Funktionalität keine Akzeptanz von den Nutzern gewinnen, kann sie entfernt werden, ohne dass eine neue Version der Anwendung ausgeliefert werden muss. Als Schlagwort ist in diesem Zusammenhang das Perpetual Beta Konzept zu nennen.

Lightweight Programming Models

In diesem Zusammenhang vergleicht Tim O´Reilly in[O'Reilly (2005)] die Amazon Web Services, die in zwei Varianten angeboten werden: Eine unter strenger Einhaltung der Formalien von SOAP, die andere nutzt dagegen die einfache Verbreitung von XML über HTTP mittels eines leichtgewichtigeren Ansatzes. Dieser Ansatz ist als Representational State Transfer (REST) bekannt[7]. Laut Tim O´Reilly verwenden die wenigen hochwertigen B2B-Verbindungen (z.B. zwischen Amazon und den großen Vertriebspartnern wie Toys'R'Us) die SOAP-Variante, dagegen wählen 95 Prozent der sonstigen Nutzer den REST-Service.

Die Unterstützung von Lightweight Programming Models, die lose gekoppelte Systeme ermöglichen, erleichtert die Verknüpfung von solchen Technologien und deren Einbindung in andere Anwendungen.

Software über Gerätegrenzen hinweg

Web 2.0 bestehen aus verschiedenen Diensten von völlig unterschiedlichen Rechnern. Solche Anwendungen sind nicht länger auf PC-Plattformen beschränkt. Dadurch besteht keine Notwendigkeit, ein spezielles Betriebssystem oder spezielle Software zur Nutzung von Diensten einzusetzen, die von Web 2.0 Anwendungen zur Verfügung gestellt werden.

Da mehr und mehr Gerätetypen an die neue Plattform angebunden werden, werden in diesem Bereich von Web 2.0 immer noch große Änderungen erwartet [vgl. O'Reilly (2005)].

[7] Für weitere Informationen siehe u.a [Fielding (2000)] und [Khare und Taylor (2004)]

Rich User Experiences

JavaScript und DynamicHTML (DHTML) waren schon längst als leichtgewichtigere Ansätze bekannt, um clientseitige Programmierung und bessere Benutzerführung, genannt RUEs (Rich User Experiences), zu ermöglichen. Einige Jahre zuvor prägte Macromedia den Ausdruck „Rich Internet Applications“ (siehe Kapitel [2.2]), um hervorzuheben, dass Flash nicht nur Multimedia-Inhalte ausliefern, sondern auch GUI-artige Anwendungsoberflächen ermöglicht.

Das wahre Potential des Webs in Bezug auf die Bereitstellung vollwertiger Web Anwendungen, die sich hinsichtlich Interface und Benutzerinteraktion kaum von einer Desktop Anwendung unterscheiden, wurde erst mit Entwicklung von Gmail[8] bekannt. Die dabei von Google verwandte Technologie wurde ursprünglich von Jesse James Garrett,dem Präsident und Gründer von Adaptive Pathder Webdesign-Firma Adaptive Path[9] als „AJAX“ genannt. Mittlerweile ist AJAX eine Schlüsselkomponente von Web 2.0 Anwendungen. AJAX wird in dem folgenden Unterkapitel [2.3.2] vorgestellt.

2.3.2 Asynchronous JavaScript and XML (AJAX)

Die Idee hinter Asynchronous JavaScript and XML ist, den Austausch von redundanten Informationen zu vermeiden. Mit AJAX soll sich die Kommunikation von Client und Server nur auf die tatsächlich zu verändernden Teile einer Seite beschränken. In [Garrett (2005)] wird Asynchronous JavaScript and XML von Jesse James Garret nicht als eine, sondern als mehrere einzelne Technologien definiert, die zusammenspielend die neuen mächtigen Wege eröffnen. Asynchronous JavaScript and XML (AJAX) vereint folgende Technologien:

- Standard-basierte Präsentation [vgl. Veen (2003)] durch XHTML und CSS
- Dynamische Repräsentation und Interaktion durch Document Object Model (DOM)
- Datenaustausch und Manipulation durch XML und XSLT
- Asynchrone Abrufung der Daten durch XMLHttpRequest
- JavaScript bindet alle Technologien zusammen

Die Abbildung [2.2] (links) zeigt das klassische Modell einer Web Anwendung. In den klassischen Web Anwendungen lösen die meisten Benutzer-Aktionen eine HTTP-Abfrage zu einem Web Server aus. Der Server verarbeitet die Anfrage, indem er weitere Daten abruft und/oder berechnet, und mit verschiedenen Legacy Systemen kommuniziert. Dann sendet

[8] http://gmail.google.com ; Zugriffsdatum: 30.01.2008
[9] http://www.adaptivepath.com ; Zugriffsdatum: 30.01.2008

er eine HTML-Seite als eine Antwort an den Client. Während der Server eine Anfrage verarbeitet, muss der Benutzer auf die Antwort warten. Dies ist natürlich ein Nachteil dieses Modells.

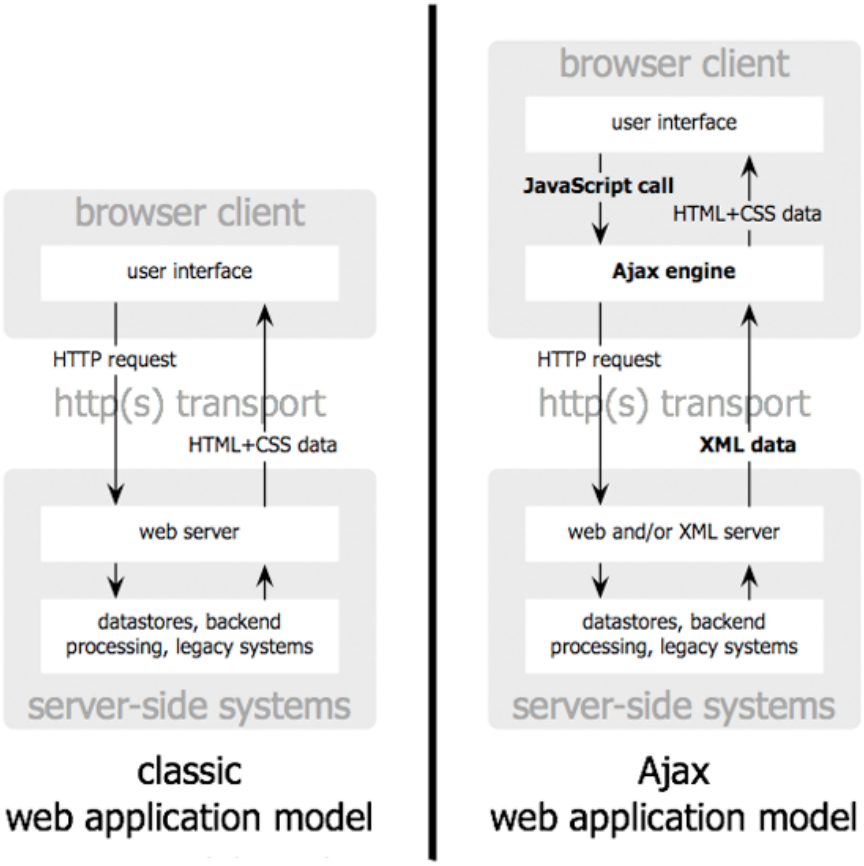

Abbildung 2.2: Das Modell einer traditionellen Web Anwendung (links) im direkten Vergleich mit einer AJAX Anwendung (rechts). Quelle: [Garrett (2005)]

Eine AJAX Anwendung bricht das starre Interaktionsmuster „Anfrage/Antwort“ zwischen Browser und Web Server auf. Wie die Abbildung [2.2] (rechts) zeigt, wird eine AJAX Engine zwischen dem Benutzerinterface und dem Server eingeführt. Dadurch wird ermöglicht, auf Benutzereingaben sofort zu reagieren und den Inhalt der angezeigten Web Seite dynamisch anzupassen. Während die Seite beim Benutzer angezeigt wird, können neue Daten dynamisch geladen und in die Seite eingebaut werden, ohne das diese Seite komplett neu geladen werden muss. Dadurch wird dem Nutzer die Wartezeit erspart. Ferner schafft AJAX neue Möglichkeiten, Web Anwendungen gut zu strukturieren und die bisherige unübersichtliche Mixtur aus den Client- und Server-seitigen Skripten und den Programmfragmenten abzulösen.

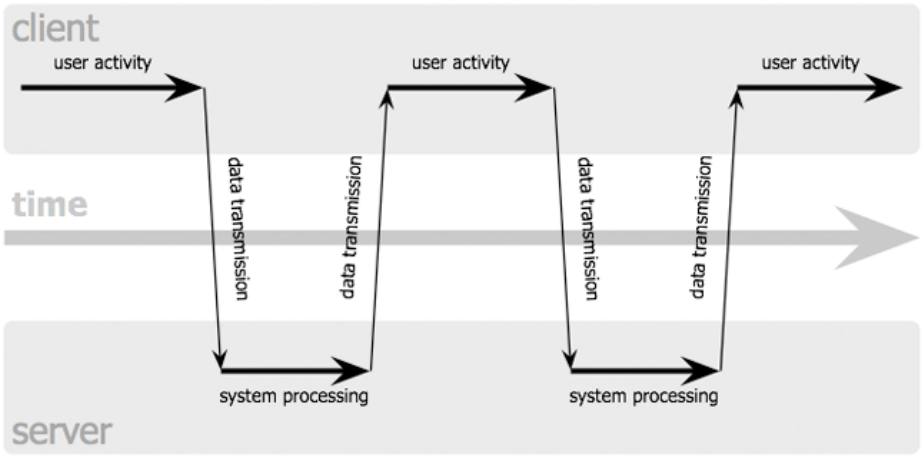

Abbildung 2.3: Der Prozessfluss einer traditionellen Web Anwendung. Quelle: [Garrett (2005)]

Anstatt des Ladens einer Web Seite zu Beginn der Sitzung, lädt der Browser eine in JavaScript geschriebene AJAX Engine. Diese Engine ist für die Darstellung des Benutzerinterfaces und für die Kommunikation mit dem Server zuständig. Die AJAX Engine erlaubt den Benutzern unabhängig von der Kommunikation mit dem Server, mit der Anwendung asynchron zu kommunizieren (siehe Abbildung [2.4]). In diesem Fall wird ein Benutzer nie ein leeres Browser Fenster und ein Sanduhr-Symbol während des Wartens auf eine Antwort des Servers sehen.

Jede Aktion des Benutzers, die in den klassischen Web Anwendungen mittels HTTP Anfragen/Antworten erfolgt (siehe Abbildung [2.3]), findet jetzt in Form eines JavaScript Aufrufes an die AJAX Engine statt. Eine Aktion des Benutzers, sei es die Daten-Validierung, die Änderung der Daten im Speicher oder sogar einige Navigationen im Browser, fordert keine Anfrage an den Server. Um die Serveranfragen kümmert sich die AJAX Engine. Sollte die AJAX Engine etwas von dem Server benötigen, sei es die Daten für die Bearbeitung oder das Laden des zusätzlichen Codes, oder das Finden der neuen Daten, startet es meistens mittels XML die Serveranfragen asynchron, ohne die Interaktion des Nutzers mit der Anwendung zu beeinflussen.

Die größten Herausforderungen bei der Entwicklung von AJAX Anwendungen sind nicht technisch. Der Kern der AJAX Technologien sind keine neuen aber ausgereiften, stabilen und gut verstandenen Technologien. Stattdessen sind die Herausforderungen für Designer der Anwendungen wichtig. Man muss vergessen, über die Einschränkungen von Web zu denken. Es sollten neue breitere und reichere Bereiche von AJAX Möglichkeiten verwendet werden [vgl. Garrett (2005)].

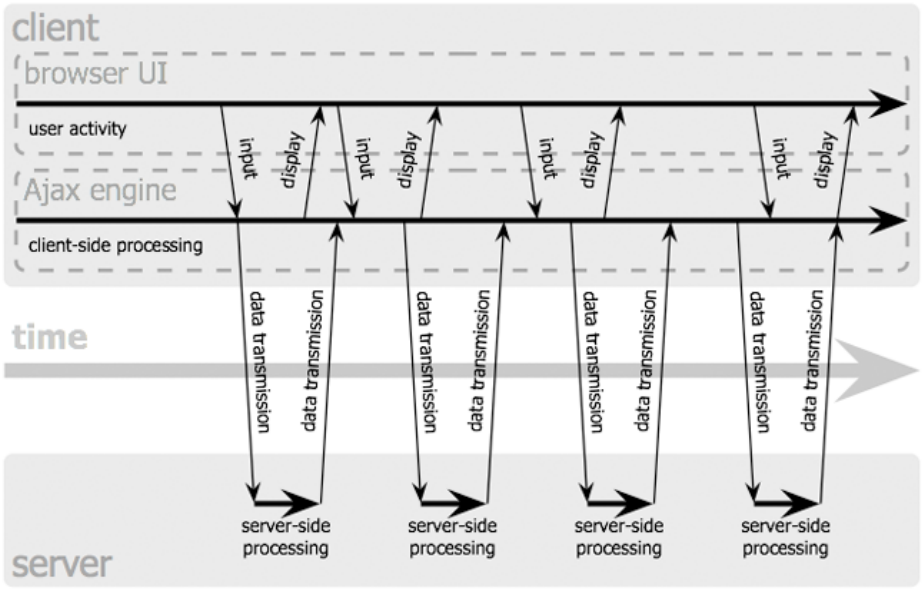

Abbildung 2.4: Der Prozessfluss einer Ajax Anwendung. Quelle: [Garrett (2005)]

Im Hinblick auf Rich Internet Applications stellt AJAX gewissermaßen die natürliche Weiterentwicklung von Web Anwendungen dar, um Desktop Anwendungen hinsichtlich Komfort und Produktivität näher zu kommen.

Pipes & Filter

Aus architektonischer Sicht implementiert AJAX das Pipes und Filters - Muster [vgl. Vlissides (1996)], das an dieser Stelle kurz vorgestellt wird.

Wie die Abbildung [2.5] darstellt, hat jede Komponente dieses Musters eine Menge von Eingängen und Ausgängen. Ein Datenstrom wird an den Eingängen einer Komponente eingelesen, von ihr bearbeitet und an den Ausgängen als Ausgabe weitergeleitet. Solche Komponenten heißen Filters. Die Verbindungen zwischen Filtern, die die Ausgaben eines Filters zu den Eingängen des Anderen transportieren, sind sog. Pipes. Das bekannteste Beispiel für diesen Stil sind Unix-Skripte oder Programme (die Pipe ist hier direkter Bestandteil des Betriebssystems).

Die Filterbausteine sind weitgehend von einander unabhängig und arbeiten meist asynchron. Die konkreten Filterimlementierungen können Einschränkungen über den Eingabestrom setzen oder Garantien über die Ausgabe geben. Die Filter sollen keine Komponenten aus ihrer Nachbarschaft kennen. Weiterhin soll die Korrektheit der Ausgabe eines Pipes&Filters - Netzwerkes nicht von der Reihenfolge der Filter abhängen.

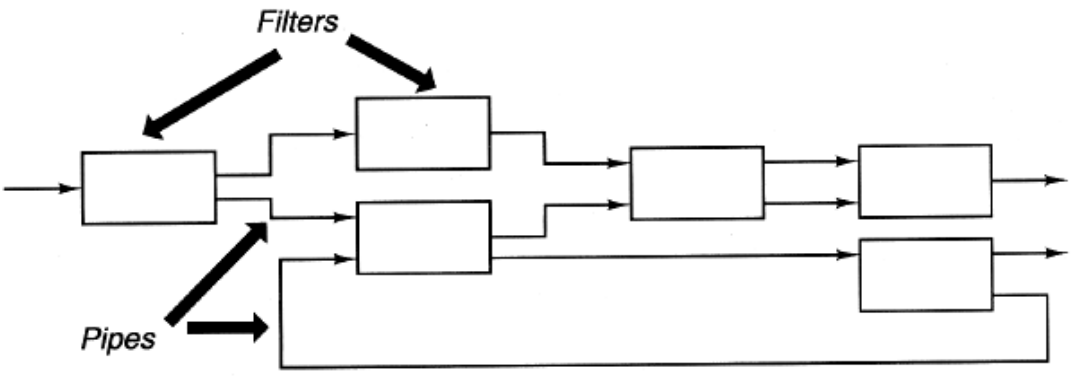

Abbildung 2.5: Pipes&Filters, Quelle: [Shaw und Garlan (1996)]

2.4 Java-basierte AJAX Frameworks

Da diese Arbeit die Migration einer Rich Client Applikation auf Code on Demand Technologie behandelt, wird erster Prototyp der auf Code on Demand Technologie basierten Web Anwendung entwickelt. Für die Entwicklung wird ein Framework benötigt. Da die existierende Rich Client Applikation in der Programmiersprache Java entwickelt wurde, sollen die Frameworks so ausgewählt werden, dass die Entwickler kaum Einarbeitungszeit brauchen. Aus diesem Grund kommen nur Java-basierte AJAX Frameworks in Frage. Durch den Einsatz von Java-basierten Frameworks sollen sich die Entwickler nicht mit den Eigenarten der Browser beschäftigen müssen. Außerdem soll die fehlende Modularität von JavaScript umgangen werden, und die Wiederverwendung von AJAX-Komponenten erleichtert werden. An dieser Stelle werden zwei Java-basierte AJAX Frameworks vorgestellt.

2.4.1 Google Web Tookit

Google Web Toolkit (GWT) ist ein Open-Source Framework zur Erstellung der AJAX-fähigen Web Anwendungen mittels der Programmiersprache Java. Das Framework wurde im Mai 2006 auf der JavaOne Konferenz[10] vorgestellt. Möchte man seine Anwendung ausliefern, übersetzt der GWT-Compiler die Java-Anwendung in Browser-konforme HTML und JavaScript.

[10] http://java.sun.com/javaone/sf/2006/index.jsp ; Zugriffsdatum: 30.01.2008

In seiner Diplomarbeit evaluierte Virginio Carfagno zwei folgende Fragen [vgl. Carfagno (2007)]:

1. Können mit dem Google Web Toolkit 2D-Ajax-Anwendungen nach den State-of-the-Art-Entwurfsprinzipien gebaut werden? (Entwicklersicht)
2. Können mit dem Google Web Toolkit die Anforderungen an eine grafisch mittelmäßig anspruchsvolle 2D-Ajax-Anwendung umgesetzt werden? (Benutzersicht)

Die erste Frage konnte Virginio Carfagno mit „Ja“ beantworten, weil der Systementwurf fast 1:1 und sehr direkt implementiert werden konnte [vgl. Carfagno (2007)].

Da die von Virginio Carfagno entwickelte Mind-Mapping-Anwendung nur in dem Browser Firefox[11] korrekt funktioniert, konnte er die zweite Frage leider nur mit „Nein“ beantworten [vgl. Carfagno (2007)].

An dieser Stelle werden die Architektur und die wichtigsten Bestandteile des Toolkits vorgestellt.

Google Web Toolkit Architektur

Die Abbildung [2.6] stellt die Architektur des Google Web Toolkits dar. Sie besteht aus folgenden Komponenten [vgl. Google (2006)]:

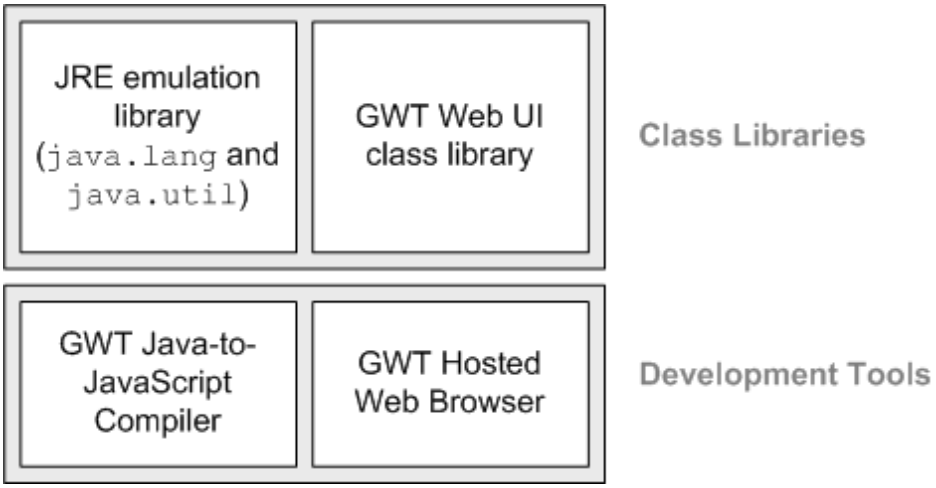

Abbildung 2.6: Google Web Toolkit. Architektur. Quelle: [Google (2006)]

[11] http://www.mozilla.com/firefox/ ; Zugriffsdatum: 30.01.2008

- **Java-to-JavaScript Compiler**
  Der GWT Java-to-JavaScript Compiler übersetzt die Programmiersprache Java in JavaScript. Der Compiler wird für die Ausführung der Anwendungen in dem Web Modus verwendet, in dem die Anwendungen als pure HTML und JavaScript Anwendungen laufen. Durch die Übertragung nach JavaScript ist die volle AJAX (Code on Demand) Funktionalität gegeben, der Browser kann asynchron Objekte nachladen.
- **Hosted Web Browser**
  Mit dem GWT Hosted Web Browser können die Anwendungen in dem Hosted Modus ausgeführt werden. In diesem Modus läuft die Anwendung innerhalb von Java Virtual Machine (JVM). Dieser Modus ermöglicht die Ausnutzung der Vorteile der Java-Entwicklung (z.B. Debuggen).
- **JRE emulation library**
  GWT beinhaltet JavaScript Implementierung einiger Klassen der Java Standard Class Library. An dieser Stelle sieht der Autor dieser Arbeit einen großen Nachteil von Google Web Toolkit, weil das Framework nur die Mehrheit der Klassen aus dem java.lang Package und einen Subset der Klassen aus dem java.util Package unterstützen. Der Rest des Java APIs wird aus dem einen oder dem anderen Grund nicht unterstützt [vgl. Google (2006)]. Die Programmiersprache Java wird nur in der Version 1.4 oder den früheren Versionen unterstützt.
- **Web UI class library**
  Die GWT Web UI-Klassenbibliothek ist eine Reihe von den benutzerdefinierten Interfaces und Klassen, die es ermöglichen, Web Browser „Widgets“ (z.B. Buttons, Text boxes, etc.) zu erzeugen. Diese Klassenbibliothek ist der Kern der User Interface Library zum Erstellen von Anwendungen GWT.

Remote Procedure Call Mechanismus

GWT Remote Procedure Call (RPC) ist ein Mechanismus für die Interaktion einer GWT Anwendung mit einem entfernten Server. Für Client und Server vereinfacht GWT RPC den Transport der Java Objekte hin und her über HTTP. Bei der Verwendung von RPCs hat man die Möglichkeit, die UI-Logik an den Client zu transportieren. Als Konsequenz daraus wird die Leistung der Anwendung verbessert, und die Bandbreite und die Web-Server Belastung reduziert.

Die Abbildung [2.7] veranschaulicht die Klassen und Interfaces, die implementiert oder generiert werden müssen, damit ein Service aufgerufen werden kann.

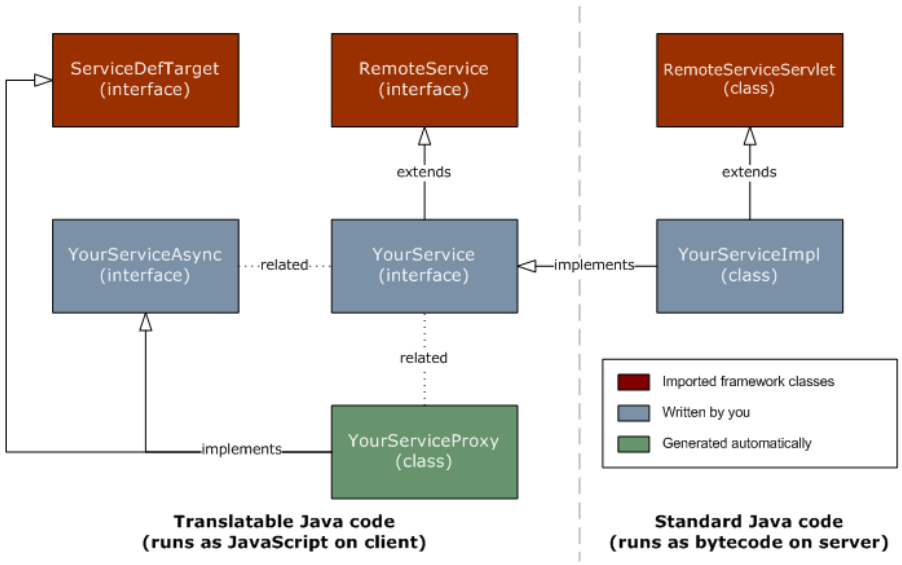

Abbildung 2.7: Google Web Toolkit. Remote Procedure Call Mechanismus. Quelle: [Google (2006)]

Einige dieser Klassen, wie der Proxy-Service (YourServiceProxy (class)), werden automatisch generiert. Der Entwickler bekommt nie mit, dass die Klasse existiert. Das Muster von Helper-Klassen ist identisch für jeden Service, der entwickelt werden soll.

JavaScript Native Interface (JSNI)

Die GWT Anwendungen werden in der Programmiersprache Java geschrieben. Der GWT Compiler übersetzt den Java Code in JavaScript. Manchmal ist es praktisch, die Möglichkeit zu haben, den handgeschriebenen JavaScript Code in Java Code zu mischen. Über das JavaScript Native Interface (JSNI) lässt sich JavaScript direkt in den Java Code einbinden, so dass auch spezifische Erweiterungen möglich sind, die sich nicht durch den Umfang der Java Bibliotheken aus dem GWT realisieren lassen. Durch die Verwendung von JSNI ist die Anbindung von verschiedenen existierenden JavaScript Toolkits (z.B. Dojo Toolkit[12]) möglich.

Die JSNI Methode ist eine leistungsfähige Technologie. Da JSNI Code nicht vollständig Browser-unabhängig ist, sollte sie vorsichtig verwendet werden.

[12] http://www.dojotoolkit.org ; Zugriffsdatum: 30.01.2008

2.4.2 Echo Framework

NextApp[13] hat mit Echo2 ein AJAX-basiertes Framework für die Entwicklung von Rich Web Applications als Open Source veröffentlicht. Mit EchoStudio2[14] bietet NextApp zudem ein kommerzielles Eclipse-Plug-In an, das die visuelle Entwicklung von Echo2-Applikationen erlaubt. Aus der Perspektive eines Entwicklers verhält sich das Echo Framework wie ein User Interface Toolkit (z.B. Java Swing). Die AJAX Technologie wird eingesetzt, um die Funktionalität der Web Anwendungen an der Funktionalität der Desktop Anwendungen anzunähern. Echo Anwendungen können als Server-seitigen Java-Code mit Hilfe von einem Komponenten-orientierten und Ereignis-gesteuerten API entwickelt werden.

Da das Echo Framework ein Java API zur Verfügung stellt, können die Entwickler die eine oder die andere Java Entwicklungsumgebung für die Erstellung der Anwendungen verwenden. Die Abbildung [2.8] stellt eine Java Entwicklungsumgebung NetBeans[15] dar, in der die Anwendungen mit der Verwendung von Echo Framework aber wie gewöhnt entworfen, entwickelt und getestet werden können.

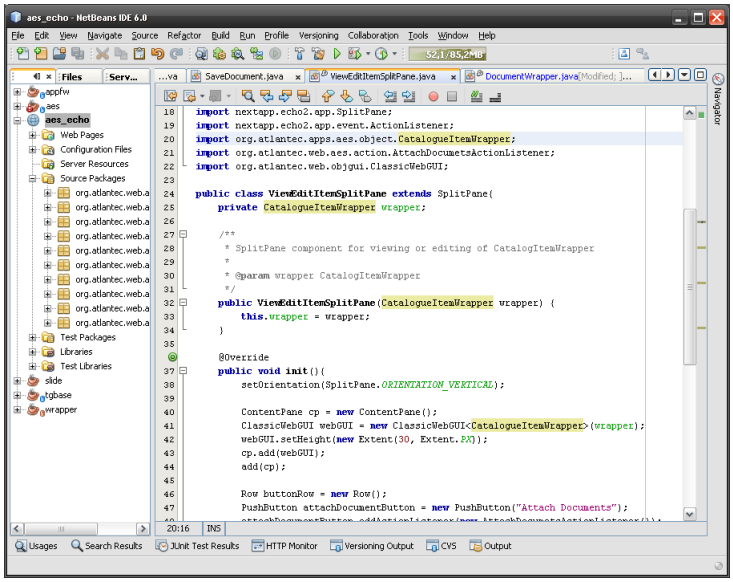

Abbildung 2.8: Verwendung von Echo Framework in NetBeans IDE

[13] http://www.nextapp.com/ ; Zugriffsdatum: 30.01.2008

[14] http://www.nextapp.com/products/echostudio/ ; Zugriffsdatum: 30.01.2008

[15] Bei der Verwendung von Echo Framework können Sie mit Ihrer lieblings Entwicklungsumgebung arbeiten und Ihre Anwendungen entwickeln, sei es NetBeans (http://www.netbeans.org/ ; Zugriffsdatum: 30.01.2008), Eclipse (http://www.eclipse.org/ ; Zugriffsdatum: 30.01.2008), IntelliJ IDEA (http://www.jetbrains.com/idea/ ; Zugriffsdatum: 30.01.2008), etc.

Echo2 Framework. Architektur.

Die Abbildung [2.6] stellt die Architektur einer Echo Anwendung dar. Es ist zu sehen, dass das Echo2 Framework (mittlere Schicht in der Abbildung [2.6]) in drei getrennte Module aufgeteilt ist:

- Application Framework
- Web Rendering Engine
- Web Application Container

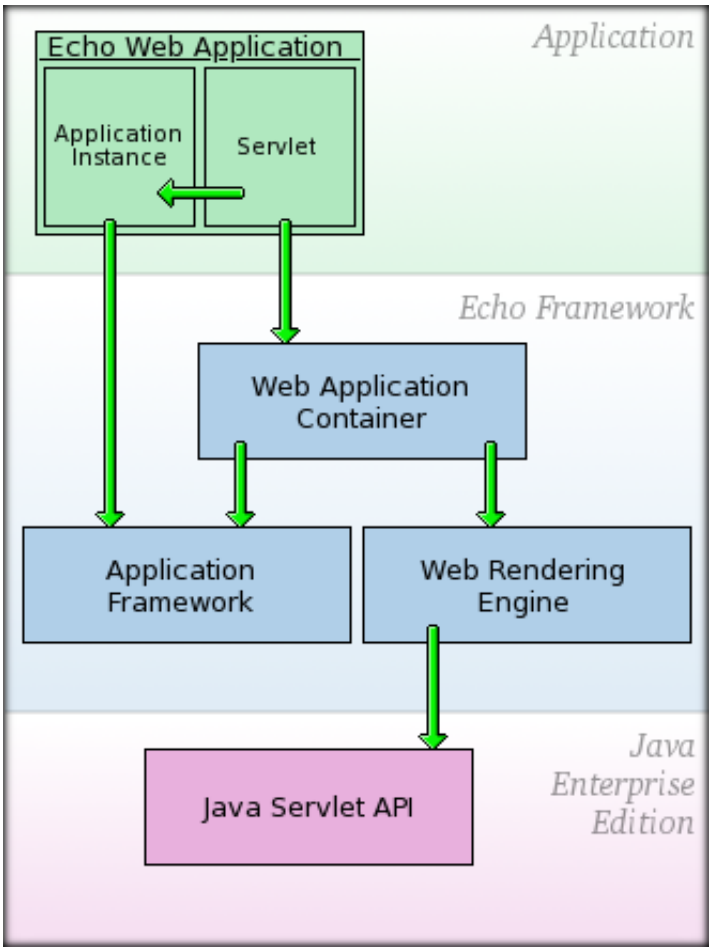

Abbildung 2.9: Echo2 Framework. Architektur. Quelle: [NextApp (2007)]

Die Abhängigkeiten zwischen den Modulen werden auf das Minimum reduziert. Die einzige Abhängigkeit besteht darin, dass das Web Application Container Modul von beiden anderen

Application Framework und Web Rendering Engine Modulen abhängig ist. Diese Abhängigkeit ist aber nicht bidirektional. An dieser Stelle werden die wichtigen Module des Echo Frameworks vorgestellt.

Application Framework
Das Application Framework Modul stellt ein API zur Verfügung, das zur Verwaltung des Zustandes der Anwendungen und zur Darstellung des User Interfaces dient. Die Interaktion mit dem Echo Framework findet durch die Verwendung des Application Frameworks[16] (ApplicationInstance, Component's und Property Objekte) und der Event-/Listener APIs statt.

Das Application Framework ist weder für das Rendering des User Interface nach HTML noch für die Kommunikation mit dem Web Browser zuständig. Darüber hinaus hat es keine Abhängigkeit zu einem solchen Code.

- **Update Manager**
  Während das Application Framework keine Randering Fähigkeiten zur Verfügung stellt, bietet es eine Infrastruktur für den Rendering Agent, um mit ihm kommunizieren zu können. Der Update-Manager wird verwendet, um die Aktualisierungen der angezeigten Hierarchie der Komponenten einer Echo Anwendung zu verfolgen, die von dem Rendering Agent empfangenen Eingaben zu bearbeiten und mit der Anwendung und ihre Komponente zu kommunizieren.

Web Rendering Engine
Das Web Rendering Engine Modul ist ein Satz von Werkzeugen, der die Grundlage für die Client / Server - Interaktion zur Verfügung stellt. Die Engine besteht aus einem in der Programmiersprache Java mit der Verwendung des Java Servlet API geschrieben serverseitigen Teil und einem in JavaScript mit der Verwendung von XMLHttpRequest-Objekt geschrieben clientseitigen Teil.

Die Web Rendering Engine ist eine Basisschicht. Sie ist von anderen Modulen völlig unabhängig. Die Web Rendering Engine ist ein generisches API für die Synchronisierung eines Web Browser mit einer serverseitigen Anwendung. Sie hat keine direkte Verbindung zu dem Application Framework Modul.

- **Server Engine**
  Der serverseitige Teil der Web Rendering Engine stellt einen HttpServlet als sein Kern zur Verfügung, der für die Bearbeitung der Clients-Anfragen zuständig ist. Wenn der Servlet eine HTTP-Anfrage erhält, delegiert er sie an einen geeigneten Service, der sich auf dem in der Anfrage definierten Service-Identifier basiert. Das Echo Framework stellt zahlreiche Service-Implementierungen zur Verfügung, die viele unterschiedlichen

[16] http://echo.nextapp.com/content/echo2/doc/api/2.1/public/ ; Zugriffsdatum: 30.01.2008

Aufgaben, so wie das Rendering des User-Interfaces als HTML-Dokument, die Unterstützung der JavaScript - Module und das Rendering von XML-Nachrichten, erledigen können.

- **Client Engine**
  Der clientseitige Teil der Web Rendering Engine ist eine JavaScript-Anwendung, die clientseitig in dem Web Browser läuft und mit der Server-Engine interagiert, um ein Remote User-Interface für die serverseitige Anwendung zur Verfügung zu stellen. Die Client Engine ist für die Initialisierung der Verbindungen (via XMLHttpRequeset) zu der Server Engine verantwortlich, um den Client / Server Zustand zu synchronisieren, wenn der Anwender Operationen ausführt. Es stellt darüber hinaus eine Reihe von Plattform-unabhängigen APIs für die Interaktion mit Web-Browser zur Verfügung.

Web Application Container
Der Web Application Container ist eine Erweiterung der Web Rendering Engine, die dazu dient, den Zustand eines User-Interface, das mit dem Application Framework entwickelt wurde, mit den Remote Web Clients zu rendern und zu synchronisieren. Als Erweiterung der Web Rendering Engine, bietet er Erweiterungen sowohl für die Server Engine als auch die Client Engine, d.h. es enthält den serverseitigen Java-und clientseitigen JavaScript-Code.

2.5 Fazit

In diesem Kapitel wurden die zum weiteren Verständnis der Arbeit notwendigen Grundlagen vorgestellt. Zunächst wurden Rich Client Applikationen kurz erwähnt und die Eigenschaften der Rich Internet Applikationen skizziert. Danach wurden die notwendigen Grundlagen der Code on Demand Technologie vorgestellt, inklusive den Begriff Web 2.0 und das Asynchronous JavaScript and XML (AJAX) Konzept. Anschließend wurden das Google Web Toolkit und das Echo Framework anhand der Vor- und Nachteilen verglichen. Das Google Web Toolkit unterstützt clientseitig nur ein Subset (java.lang und java.util Pakete) des Java APIs. Bei der Entwicklung eines Rich Internet Clients ist es ein großer Nachteil. Aus diesem Vergleich wurde entschieden, bei der Migration der existierenden Rich Client Anwendung auf Code on Demand Technologie, d.h. bei der Entwicklung der Web Anwendung, das Echo Framework zu verwenden.

3 Analyse

In diesem Kapitel werden anhand der Vorgaben des Unternehmens Atlantec Enterprise Solutions GmbH[1] die Anforderungen des Systems definiert.

Das Business des Unternehmens liegt in der Schiffbaubranche mit dem Ziel, Software-Lösungen zur Verfügung zu stellen, die den Kunden ermöglichen, Technik- und Produktionsprozesse im Schiffbau zu verbessern und zu vereinfachen. Atlantec Enterprise Solutions betreibt Produkte unter dem Markennamen Topgallant®.

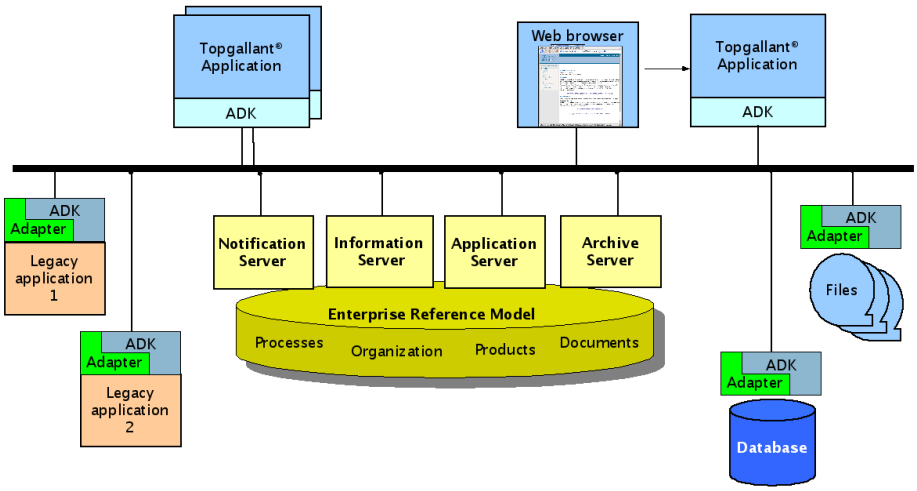

Abbildung 3.1: Topgallant®Infrastruktur. Quelle [Atlantec Enterprise Solutions (2006)]

Die Abbildung [3.1] stellt die Topgallant®Infrastruktur mit ihren wichtigen Komponenten dar. Ein der wichtigsten Komponenten der Topgallant®Infrastruktur ist der Information Server [vgl. Atlantec Enterprise Solutions (2001a)]. Er besteht aus einem potentiell verteilten Directory Server und adapter client APIs[2] für verschiedene Operationen. Der Information Server ist gleichzeitig ein Authentifizierungsserver, der eine Benutzer-Datenbank und ein spezifisches Topgallant®Rechtemodell verwaltet. Das Datenmodell aller signifikanten Typen leitet von dem Information Object ab und ist im Enterprise Reference Model (ERM) definiert [vgl.

[1] http://www.atlantec-es.com ; Zugriffsdatum: 30.01.2008
[2] Bestandteil von Topgallant®Adapter Developer Kit

Atlantec Enterprise Solutions (2003a)]. Weitere Komponenten der Topgallant®Infrastruktur sind in [Atlantec Enterprise Solutions (2006)] beschrieben.

Für die Migration auf Code on Demand Technologie wurde Part Catalog Editor als eine existierende Rich Client Anwendung ausgewählt. Viele unterschiedliche Unternehmen bieten Katalog-basierte Verwaltungssysteme. Der Part Catalog Editor ist ein Informationssystem, das ein Werkzeug zum Erzeugen, Ändern, Suchen und Anzeigen etc. von den klassifizierten Katalogeinträgen zur Verfügung stellt. In Topgallant®werden Schiffbauteile als Katalogeinträge betrachtet.

Die Analyse eines Softwaresystems wird durch funktionale und nicht funktionale Anforderungen beschrieben. Als funktionale Anforderungen wird die Funktionalität des Softwaresystems verstanden. Für die Analyse der Funktionalität des Softwaresystems werden Anwendungsfälle (use cases) eingefügt.

Durch nicht funktionale Anforderungen werden weitere wichtige Eigenschaften eines Softwaresystems definiert. Als solche zählt man die Erweiterbarkeit, die Performanz, die Fehlertoleranz und die Portierbarkeit eines Softwaresystems.

3.1 Funktionale Anforderungen (use cases)

In diesem Abschnitt werden Anwendungsfälle beschrieben. Da die Funktionalität der Web Anwendung der Funktionalität der bestehenden Desktop Rich Client Anwendung, so nahe wie möglich, entsprechen sollte, sind die Anwendungsfälle aus der Spezifikation [Atlantec Enterprise Solutions (2003b)] der Desktop Rich Client Anwendung übernommen worden.

Zu jedem der folgenden Anwendungsfälle werden jeweils Screenshots der bestehenden Desktop Anwendung und ein Sequenzdiagramm dargestellt, das den genauen Ablauf einzelnes Falles verdeutlichen. Gleichzeitig wird die Funktionalität vorgestellt, die in der Rich Internet Applikation auf der Client Seite implementiert werden kann. Allgemein kann man gesagt werden, dass mindestens die Überprüfungen der Vorbedingungen der jeweiligen Anwendungsfälle clientseitig zu implementieren sind.

3.1.1 Herstellen einer Verbindung zu einem Authentifizierung Server

Zusammenfassung
Der Anwendungsfall beschreibt den Ablauf des Herstellens einer Verbindung zu einem Information Server, der auch Authentifizierungsaufgaben übernimmt. In der Abbildung [3.2] ist ein Screenshot der bestehenden Desktop Anwendung gezeigt, und die Abbildung [3.3] stellt den genauen Ablauf dieses Anwendungsfalls mittels eines Sequenzdiagramms dar.

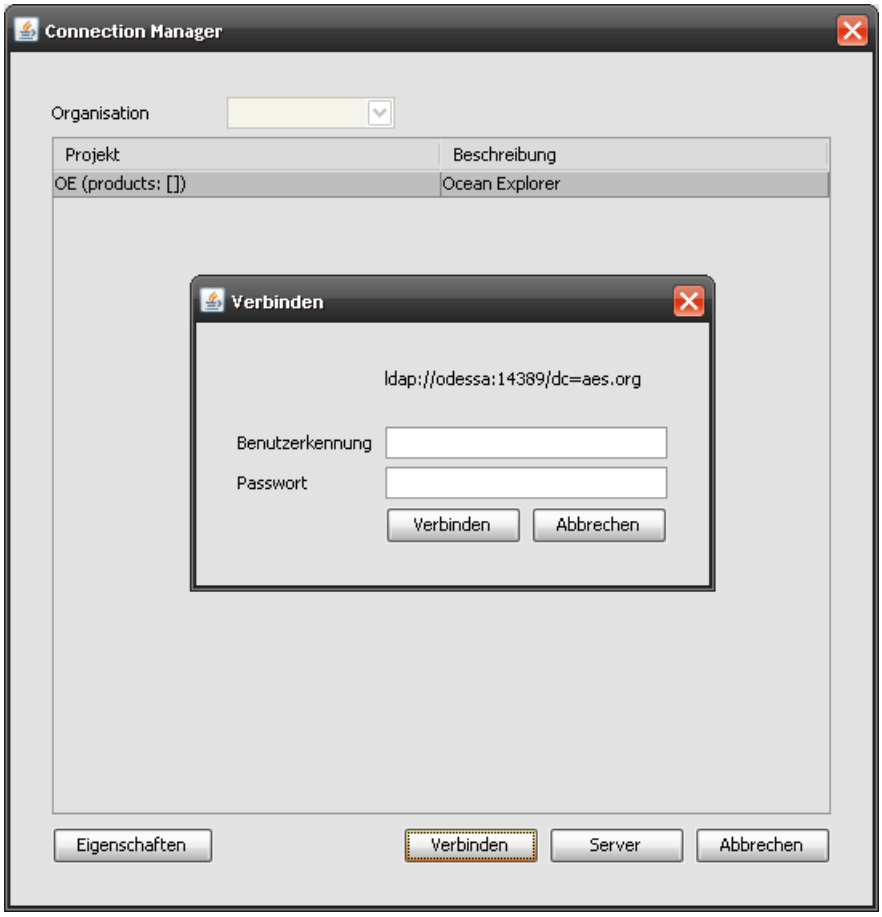

Abbildung 3.2: Herstellen einer Verbindung. Screenshot.

Vorbedingungen

Ein Topgallant®Information Server wurde vom Systemadministrator konfiguriert. Der Ingenieur hat die Zugangsdaten für den Topgallant®Information Server. Die Zugangsdaten sind Server-URL, Benutzername, Passwort und das Projekt, in dem der Ingenieur arbeiten möchte.

Beschreibung des Szenarios

Der Ingenieur sucht in der Liste vorhandener Server/Projekte das Projekt, in dem er arbeiten möchte. Ist das Projekt vorhanden, wählt er es aus und stellt eine Verbindung zu diesem Server / Projekt her. Dabei gibt er seinen Benutzernamen und Passwort ein. Sollte das Projekt nicht vorhanden sein, fügt der Ingenieur einen neuen Topgallant®Information Server in die List der bekannten Server hinzu und dann stellt zu dem Server eine Verbindung wie oben beschrieben her.

Ausnahmen
Die Verbindung kann zu einem Topgallant®Information Server nicht hergestellt werden, wenn der Server nicht gestartet ist oder die Zugangsdaten falsch eingegeben wurden.

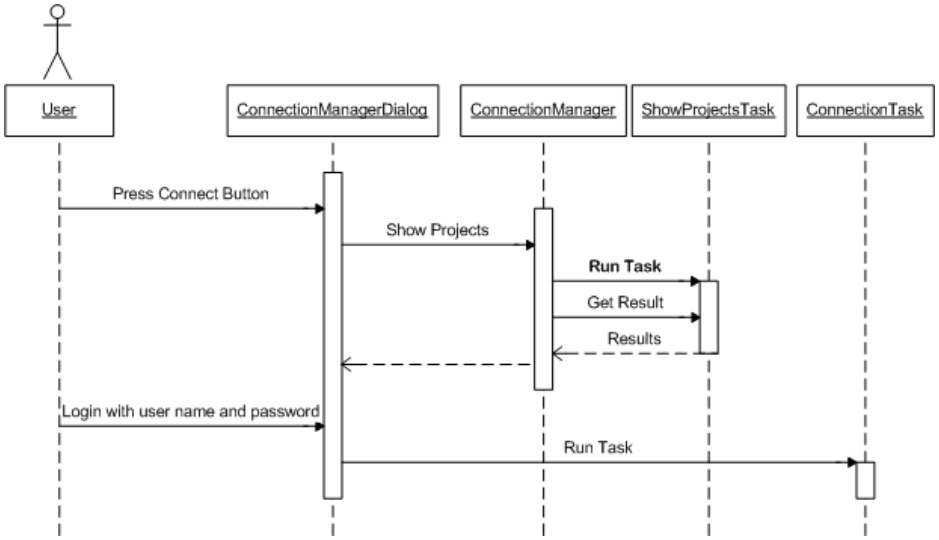

Abbildung 3.3: Herstellen einer Verbindung. Sequenzdiagramm.

Clientseitige Funktionalität
In Topgallant®Systemen können Verbindungen zu den bekannten Topgallant®Information Servers lokal benutzerspezifisch verwaltet werden. Unter einer Verbindung sind hier auch bekannte Projekte zu verstehen. Die Verwaltung der bekannten Verbindungen, dazu gehört auch die Überprüfung der Benutzereingaben, ist Funktionalität, die clientseitig implementiert werden kann.

3.1.2 Laden eines Kataloges

Zusammenfassung
Der Anwendungsfall beschreibt den Ablauf des Ladens eines Kataloges. In der Abbildung [3.4] ist ein Screenshot der bestehenden Desktop Anwendung gezeigt, und die Abbildung [3.5] stellt den genauen Ablauf dieses Anwendungsfalls mittels eines Sequenzdiagramms dar.

Vorbedingungen
Die Verbindung zu einem Topgallant®Information Server / Projekt ist hergestellt. Der Server enthält mindestens einen Katalog.

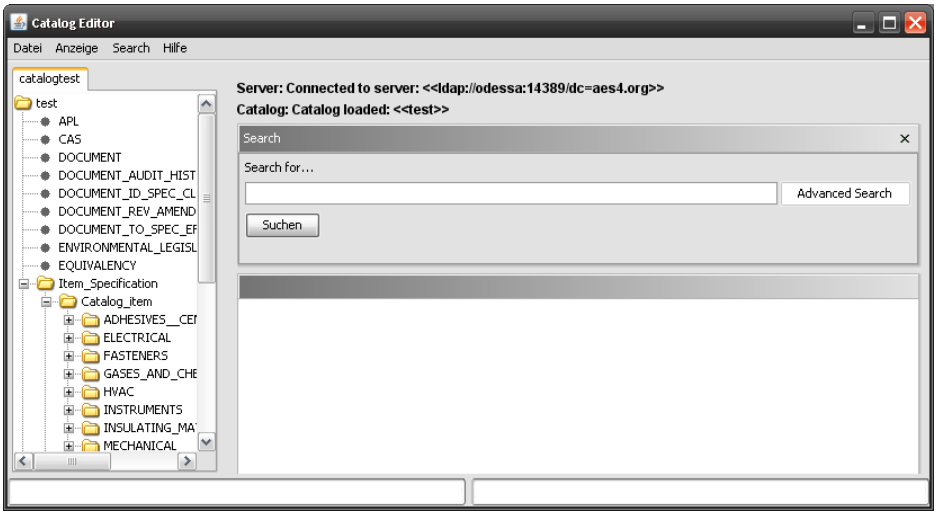

Abbildung 3.4: Laden eines Kataloges. Screenshot.

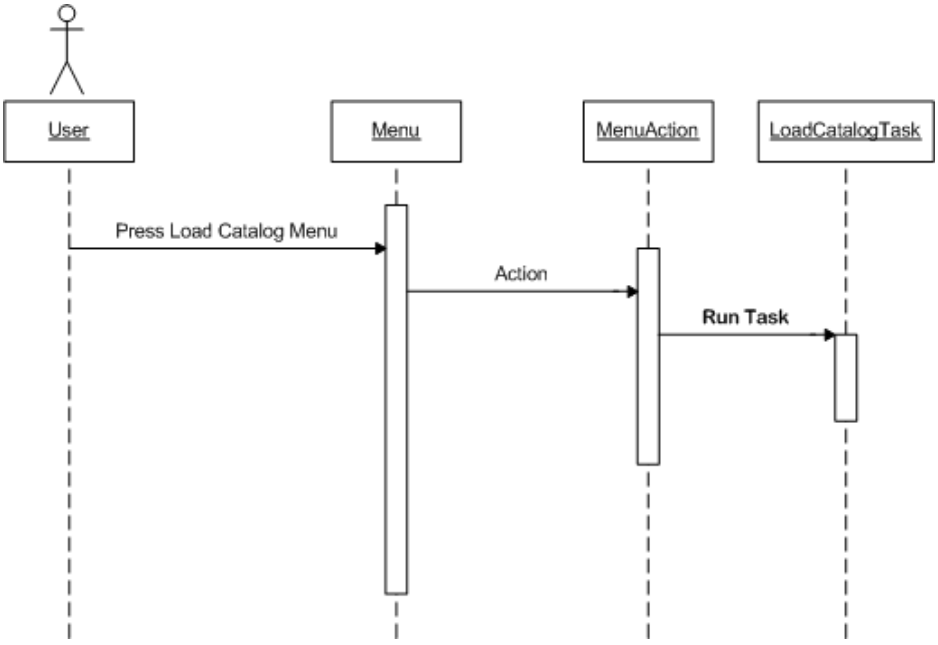

Abbildung 3.5: Laden eines Kataloges. Sequenzdiagramm.

Beschreibung des Szenarios
Der Ingenieur lädt einen Katalog. Sollte mehr als ein Katalog vorhanden sein, wählt er einen aus, mit dem arbeiten möchte.

Ausnahmen
Es ist kein Katalog im Information Server vorhanden.

Clientseitige Funktionalität
Sollte mehr als ein Katalog auf einem TopgallantⓇInformation Server vorhanden sein, so hat ein Benutzer die Möglichkeit, aus einer Liste vorhandener Kataloge eine Auswahl zu treffen, welcher Katalog geladen werden soll. Diese Interaktion mit dem Benutzer ist die Funktionalität, die clientseitig implementiert werden kann.

3.1.3 Suchen nach Katalogeinträgen, die zu einer bestimmten Kategorie gehören

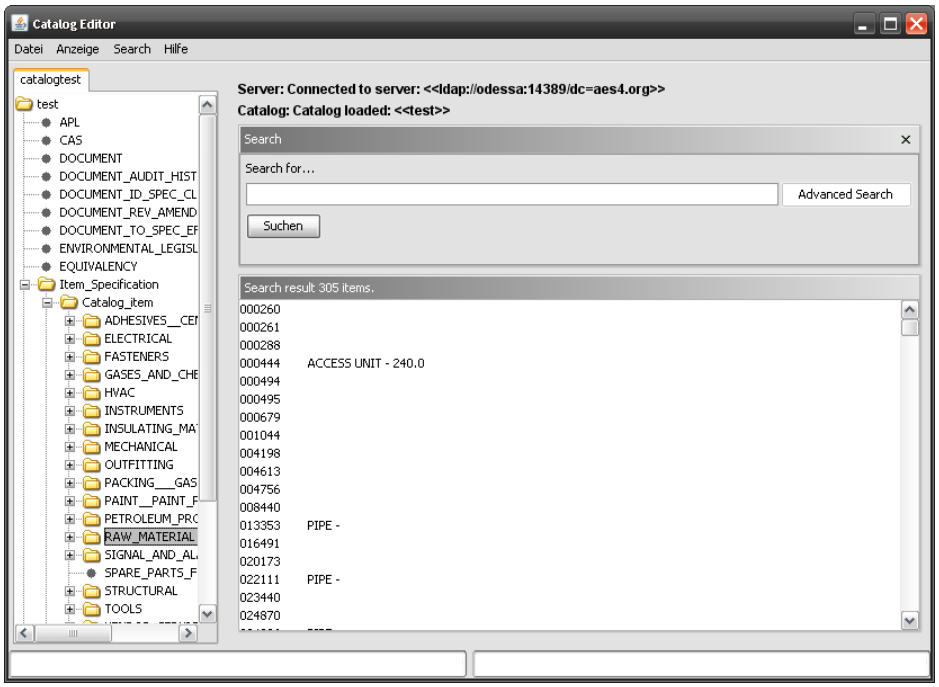

Abbildung 3.6: Suchen in einer bestimmten Kategorie. Screenshot.

Zusammenfassung
Der Anwendungsfall beschreibt den Ablauf einer Suche nach Teilen in einem Katalog, die zu

einer bestimmten Kategorie gehören. In der Abbildung [3.6] ist ein Screenshot der bestehenden Desktop Anwendung gezeigt, und die Abbildung [3.7] stellt den genauen Ablauf dieses Anwendungsfalls mittels eines Sequenzdiagramms dar.

Vorbedingungen
Die Verbindung zu einem Topgallant®Information Server / Projekt ist hergestellt. Ein Katalog ist geladen. Die Kategorie der gesuchten Teile ist eine gültige Kategorie in der Klassifikation des Katalogs.

Beschreibung des Szenarios
Der Ingenieur braucht für seine Konstruktionen verschiedene Rohre. Er sucht in einem bestimmten Katalog nach Bauteilen, die zu der Kategorie „Rohre“ gehören. Er bestellt diese dann für die Fertigung im Lager bzw. bei einem Lieferanten, wenn er die Rohre im Katalog gefunden hat.

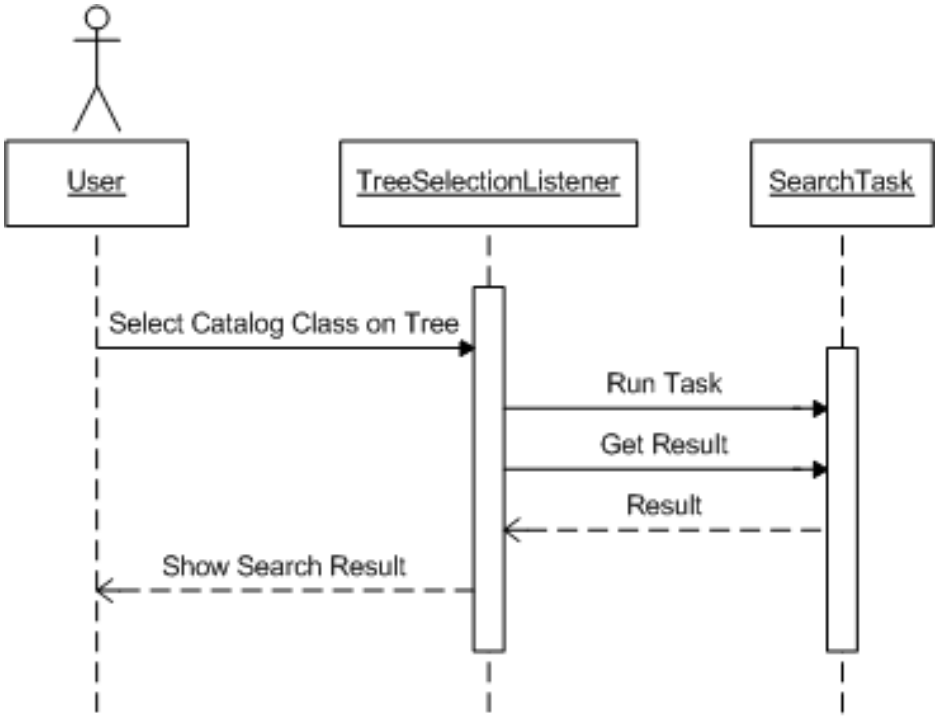

Abbildung 3.7: Suchen in einer bestimmten Kategorie. Sequenzdiagramm.

Clientseitige Funktionalität
In diesem Anwendungsfall ist die Überprüfung, ob eine Kategorie vom Benutzer ausgewählt ist, als clientseitige Funktionalität vorstellbar.

3.1.4 Suchen nach Katalogeinträgen, die einer Bedingung entsprechen

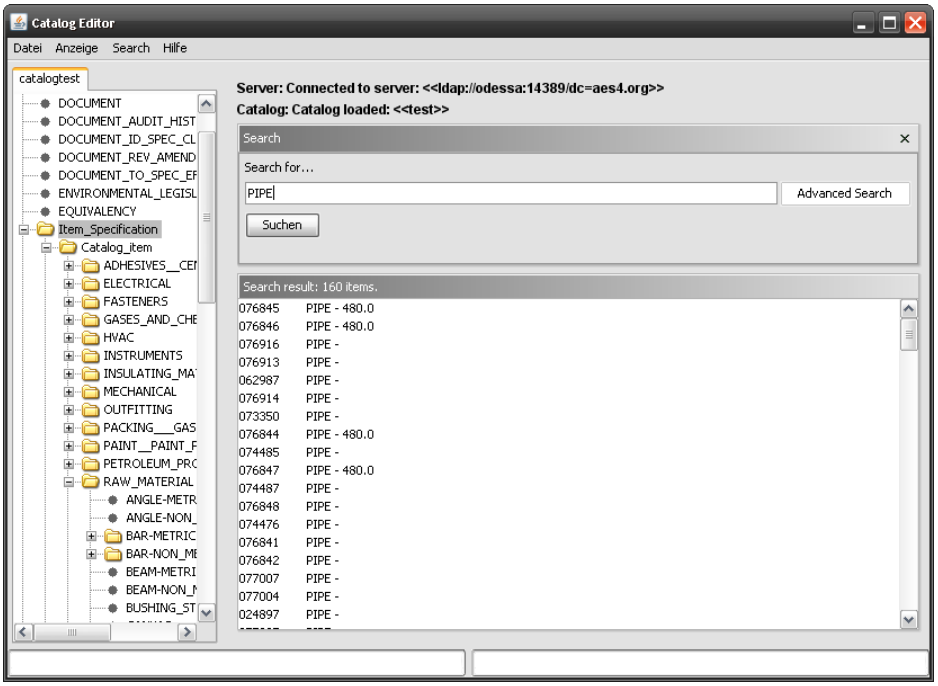

Abbildung 3.8: Suchen mit einer Bedingung. Screenshot.

Zusammenfassung
Der Anwendungsfall beschreibt den Ablauf einer Suche nach Teilen, deren Eigenschaften einer bestimmten Bedingung entsprechen. In der Abbildung [3.8] ist ein Screenshot der bestehenden Desktop Anwendung gezeigt, und die Abbildung [3.9] stellt den genauen Ablauf dieses Anwendungsfalls mittels eines Sequenzdiagramms dar.

Vorbedingungen
Die Verbindung zu einem Topgallant®Information Server / Projekt ist hergestellt. Ein Katalog ist geladen. Die Eigenschaften und deren Werte, die in der Suchanfrage auftreten, sind in der Klassifikation des Katalogs als gültig für Bauteile eingetragen. Die Suche nach einem Rohr mit der Eigenschaft „Geschwindigkeit“ ist unmöglich, da die „Geschwindigkeit“- Eigenschaft für Bauteile in der Klassifikation „Rohre“ keine gültige Eigenschaft darstellt.

Beschreibung des Szenarios
Der Ingenieur braucht für seine Konstruktionen verschiedene Rohre, deren Länge nicht größer als 200 Meter sein darf. Er sucht in einem bestimmten Katalog nach Bauteilen der Klassifikation „Rohre“, deren Länge kleiner oder gleich 200 Meter ist.

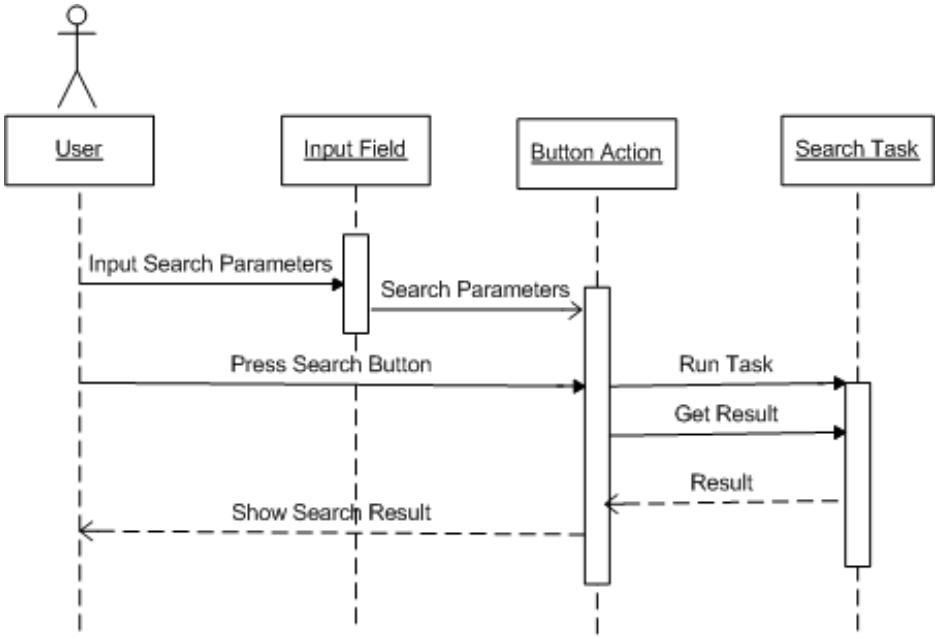

Abbildung 3.9: Suchen mit einer Bedingung. Sequenzdiagramm.

Clientseitige Funktionalität
Während der Eingabe von Suchkriterien können dem Benutzer intelligente Vorschläge, ähnlich zu Google Suggest[3], gemacht werden. Als Beispiel kann man sich hier die Situation vorstellen, bei der innerhalb von einer bestimmten Kategorie nach Einträgen gesucht werden soll, die bestimmte Werte bei bestimmten Eigenschaften haben sollen. Als Vorschläge sind die Namen der Eigenschaften vorstellbar. Diese Funktionalität kann clientseitig implementiert werden.

3.1.5 Katalogeintrag modifizieren

Zusammenfassung
Der Anwendungsfall beschreibt den Ablauf des Änderns eines Eintrages in einem Katalog. In der Abbildung [3.10] ist ein Screenshot der bestehenden Desktop Anwendung gezeigt, und die Abbildung [3.11] stellt den genauen Ablauf dieses Anwendungsfalls mittels eines Sequenzdiagramms dar.

[3] http://www.google.com/webhp?complete=1 ; Zugriffsdatum: 30.01.2008

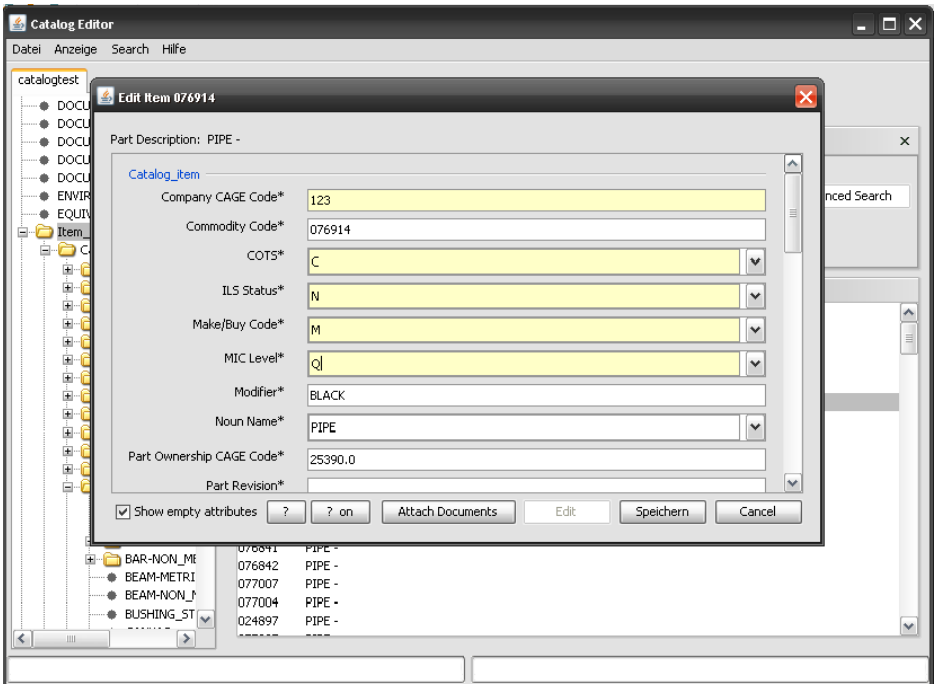

Abbildung 3.10: Modifizieren eines Katalogeintrages. Screenshot.

Vorbedingungen
Die Verbindung zu einem Topgallant®Information Server / Projekt ist hergestellt. Ein Katalog ist geladen. Der Eintrag (Bauteil), der geändert werden sollte, ist im Katalog enthalten.

Beschreibung des Szenarios
Der Ingenieur stellt fest, dass ein Eintrag mit Fehlern in dem Katalog eingetragen wurde. Er sucht nach diesem Eintrag (s. Anwendungsfälle oben). Wenn es ihm erlaubt ist, die fehlerhaften Eigenschaften zu modifizieren (s. Ausnahmen), korrigiert er sie und speichert die Änderungen.

Ausnahmen
Die zu korrigierenden Eigenschaften sind einzigartig und dürfen von Standartbenutzern nicht geändert werden. Der Benutzer wendet sich in diesem Fall an den Administrator des Kataloges, der den fehlerhaften Eintrag modifizieren kann.

Clientseitige Funktionalität
Alle Einträge im Katalog haben durch das Modell bestimmte Eigenschaften, die gesetzt werden müssen (required properties). Sollten solche Eigenschaften eines modifizierten Eintrages nicht gesetzt sein, darf er nicht abgespeichert werden. Zusätzlich könnte man sich vorstellen, dass bei den Eigenschaften, die nur bestimmte Werte besitzen können, die möglichen Werte zur Auswahl angeboten werden. Diese Funktionalität und das Validieren

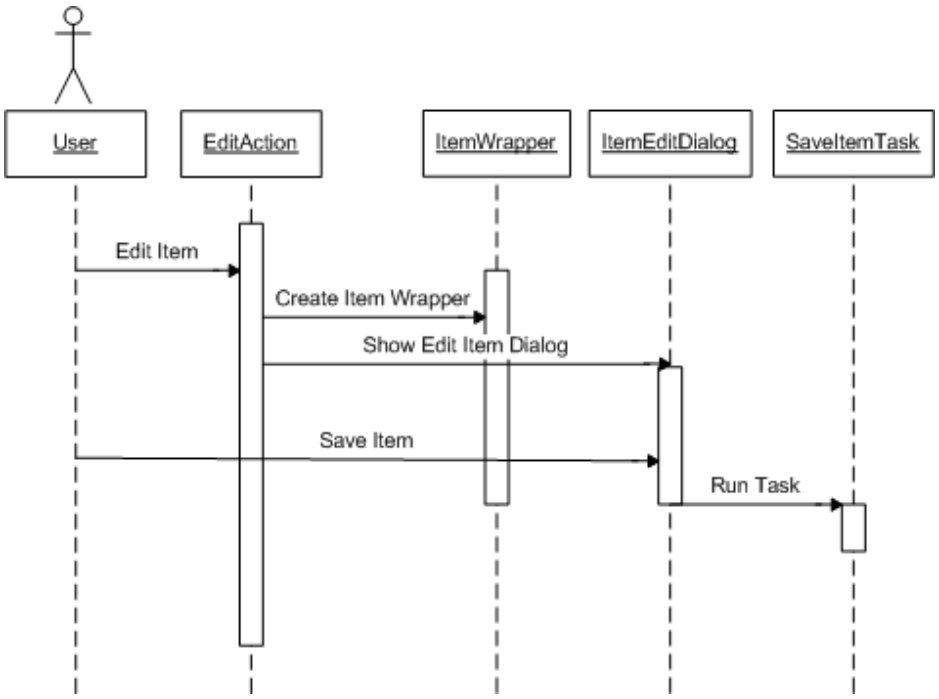

Abbildung 3.11: Modifizieren eines Katalogeintrages. Sequenzdiagramm.

des Eintrages können clientseitig implementiert werden.

3.1.6 Einfügen eines neuen Katalogeintrages in den Katalog

Zusammenfassung
Der Anwendungsfall beschreibt den Ablauf des Einfügens eines neuen Eintrages (Bauteiles) in einen Katalog. In der Abbildung [3.12] ist ein Screenshot der bestehenden Desktop Anwendung gezeigt, und die Abbildung [3.13] stellt den genauen Ablauf dieses Anwendungsfalls mittels eines Sequenzdiagramms dar.

Vorbedingungen
Die Verbindung zu einem Topgallant®Information Server / Projekt ist hergestellt. Ein Katalog ist geladen. Der Ingenieur besitzt Rechte zum Einfügen eines neuen Eintrages in den Katalog. Das Bauteil ist in dem Katalog nicht enthalten.

Beschreibung des Szenarios
Der Ingenieur sucht nach der bestimmten Kategorie, zu welcher der neue Eintrag gehören soll. Dann füllt er die entsprechende Eingabemaske für die Eigenschaften des Eintrages aus und speichert den Eintrag ab. Somit ist der neue Eintrag in dem Katalog eingetragen.

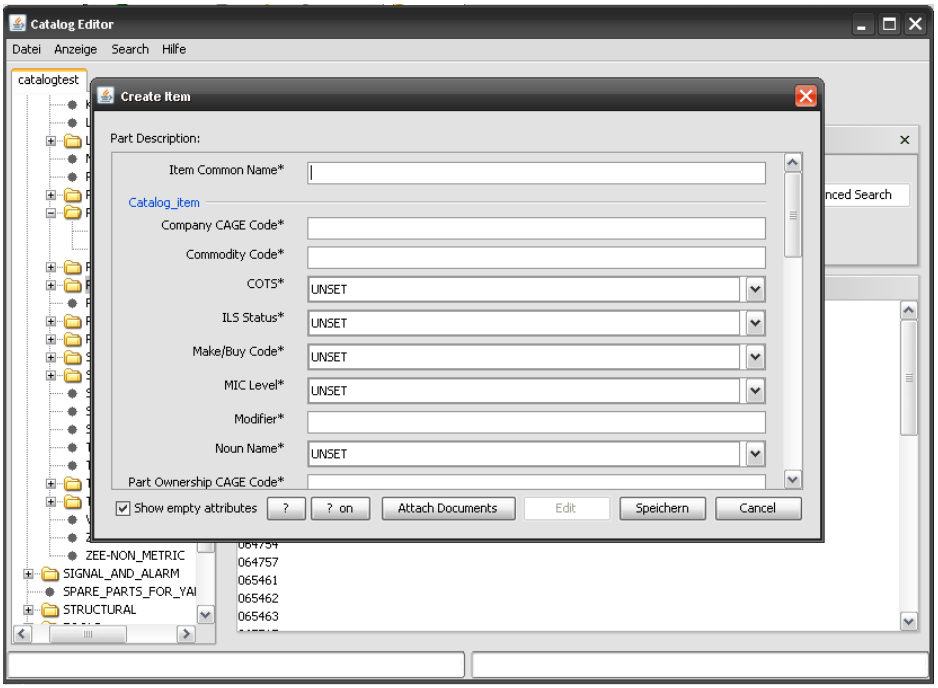

Abbildung 3.12: Erzeugen eines neuen Katalogeintrages. Screenshot.

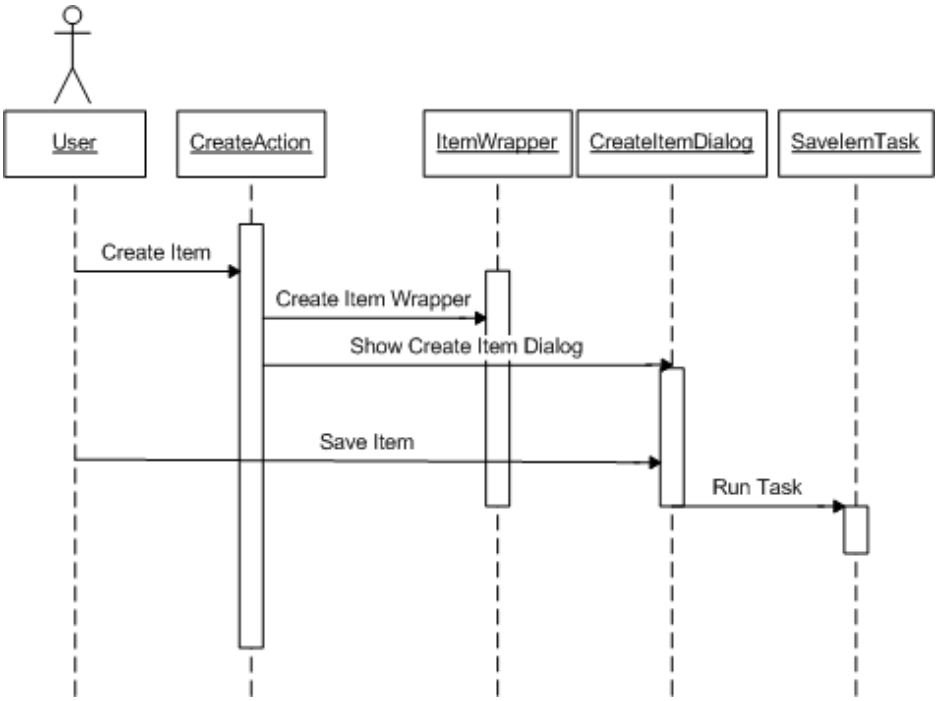

Abbildung 3.13: Erzeugen eines neuen Katalogeintrages. Sequenzdiagramm.

Clientseitige Funktionalität
Beim Einfügen eines neuen Eintrages müssen einige Eigenschaften gesetzt werden, auf deren Basis auf der Persistenzebene eine ID (Identifikationsnummer) für die Datenbank erzeugt wird. Diese Eigenschaften müssen in dem ganzen System eindeutig sein. Alle Einträge im Katalog haben einige durch das Modell bestimmte Eigenschaften, die gesetzt werden müssen (required properties). Sollten solche Eigenschaften eines neuen Eintrages nicht gesetzt werden, darf er nicht abgespeichert werden. Zusätzlich kann man sich vorstellen, dass bei den Eigenschaften, die nur bestimmte Werte haben können, die möglichen Werten zur Auswahl angeboten werden. Diese Funktionalität und das Validieren des Eintrages können clientseitig implementieren werden.

3.2 Nicht funktionale Anforderungen

In diesem Abschnitt werden nicht funktionale Anforderungen des Softwaresystems definiert. Die Definition der Anforderungen und die Qualitätsmerkmale des Systems erfolgen nach den Prinzipien des Software Engineering.

3.2.1 Performance und angemessene Antwortzeiten

Performance-Anforderungen definieren, wie gut oder schlecht das System bestimmte Funktionen ausführen muss. Zu den Performance-Anforderungen zählt man die Geschwindigkeit (z.B. Datenbankreaktionszeiten), den Durchsatz (Transaktionen pro Sekunde), die Kapazität (Auslastung durch gleichzeitige Aufrufe) und das Timing (Echtzeitanforderungen) [vgl. Wiegers (2003)].

Mit dem Softwaresystem können mehrere Benutzer gleichzeitig arbeiten. Es ist bei dem Design des Systems zu beachten, dass sich die Antwortzeiten und die Funktionalität des Systems mit großer Anzahl der Benutzer nicht beeinflussen lassen sollten. Die Kommunikation zwischen einem Client und dem Server muss für einen Nutzer in akzeptabler Zeit erfolgen.

3.2.2 Fehlertoleranz

Als Fehlertoleranz bezeichnet man die Fähigkeit eines Softwaresystems, mögliche Soft- bzw. Hardwarefehler, die im System auftreten können, entsprechend zu verarbeiten. Das Softwaresystem soll weiterhin verfügbar bleiben.

3.2.3 Sicherheit

Der Server darf keinen wahlfreien Zugriff auf die Daten, sondern nur Datenzugriff mit entsprechenden Rechten zulassen. Das heißt, dass nur autorisierte Benutzer auf die für sie bestimmten Daten und Informationen zugreifen und nur diese bearbeiten dürfen. Die Daten müssen immer verfügbar sein, wenn sie benötigt werden, und dürfen von Dritten nicht manipuliert werden.

3.2.4 Wartbarkeit. Erweiterbarkeit. Wiederverwendbarkeit.

Wichtige Anforderungen des Softwaresystems sind die Wartbarkeit und die Erweiterbarkeit. Das gesamte System muss dann und wann korrigiert und/oder an sich ändernden Anforderungen angepasst werden können. Das gesamte System soll so entworfen werden, dass der Aufwand für die Änderungen und/oder Erweiterungen akzeptabel ist [vgl. Ludewig und Lichter (2006)]. Das System muss mit anderen Worten lebendig sein, so dass die Anpassung ohne großen Aufwand nach Kundennachfragen realisiert werden kann.

Die Wiederverwendbarkeit ist ein gefragtes Ziel der Softwareentwicklung. Sie gibt Aufschluss über den relativen Aufwand, der erforderlich ist, um eine Softwarekomponente für den Einsatz in andere Anwendungen zu konvertieren. Die Entwicklung der wiederverwendbaren Software ist kostenspieliger, als die Erstellung einer Komponente, die nur in einer Anwendung benutzt werden kann [Wiegers (2003)].

3.2.5 Portierbarkeit. Heterogene Landschaften

Der Aufwand, der mit der Migration einer Software von einer Betriebsumgebung zu einer anderen verbunden ist, ist eine Maß für die Portierbarkeit. In dem Fall der Web Anwendung ist diese Betriebsumgebung ein Web Browser, der auf dem Client Rechner installiert ist.

Da die Software von unterschiedlichen Kunden benutzt werden kann, ist zu berücksichtigen, dass die Kunden mit unterschiedlichen Web Browser arbeiten können. Aus diesem Grund sollte das Softwaresystem browserunabhängig sein. Alle Betriebssysteme besitzen einen Web Browser. Mit dem Browser soll das gesamte System benutzbar sein. Ein Web Browser soll ein einziges Programm sein, das auf einem Client Rechner als Voraussetzung für die Benutzung des gesamten Systems vorhanden sein sollte.

3.2.6 Entwicklungsaufwand

Nachdem das Design des Systems entworfen wird (darunter versteht man die Anforderungsnalyse und den Entwurf der Architektur etc.), wandert man in die Realisierungsphase, in der das System codiert wird.

Die existierende Rich Client Anwendung ist in der Programmiersprache Java [Sun Microsystems (b)] entworfen. Das darunter liegende Application Programming Interface, das von der Firma Atlantec Enterprise Solutions GmbH[4] zur Verfügung steht, ist auch in der Programmiersprache Java implementiert. Dies bedeutet, dass die Entwicklung standardmäßig in der Programmiersprache Java durchgeführt wird und die Entwickler diese Programmiersprache beherrschen.

Unter einer Migration einer Rich Client Anwendung auf Code on Demand Technologie versteht man unter anderem auch die Entwicklung einer neuen Web-Anwendung, die die Funktionalität der existierenden Rich Client Anwendung wiederspiegelt. Oft werden Web-Anwendungen mit Hilfe von Markup und/oder Scriptsprachen[5] entwickelt. Eine neue Programmiersprache bedeutet für eine Firma einen zusätzlichen Aufwand, somit auch zusätzliche Kosten, die Schulungen für Entwickler verursachen.

Aus den oben genannten Gründen ist es, bei der Analyse ein Framework für die Entwicklung einer Rich Internet Application auszuwählen und bei der Implementierung zu verwenden, das ein Java API zur Verfügung stellt. Mit anderen Worten, es soll eine Rich Internet Application mit einem guten Java-basierten Framework entwickelt werden. Nur in Fällen, in denen man ohne Script- und/oder Markupsprachen nicht auskommen kann, dürfen solche als Notlösung verwendet werden.

3.3 Fazit

In diesem Kapitel wurden die funktionalen und nicht funktionalen Anforderungen für das gesamte System definiert. Das zu entwickelnde System soll für einen Endbenutzer kein wirklich neues System werden. Die Nutzer, die mit der existierenden Rich Client Anwendung arbeiten, sollen mit einem geringeren Aufwand mit der neuen Web-Anwendung ihre Aufgaben erledigen können. Dies bedeutet wiederum für Entwickler, dass das Graphical User Interface (GUI), Menus, Dialoge etc. sehr ähnlich zu der existierenden Anwendung zu entwickeln sind.

[4] http://www.atlantec-es.com ; Zugriffsdatum: 30.01.2008
[5] Z.B. HTML, PHP, JavaScript etc.

Da es möglich ist, in der neuen Rich Internet Application Teile der Business Logik, heißt die Benutzerinteraktion etc., clientseitig ausführen zu lassen, ist kaum Kommunikation zu einem Applikation Server notwendig. Nur wenn echte Daten manipuliert werden sollen, wird die Kommunikation zu einem Applikation Server stattfinden und die notwendige Business Logik auf der Server Seite ausgeführt. Dadurch werden die Antwortzeiten des Systems optimiert.

Die Funktionalität der neuen Rich Internet Application wird teilweise clientseitig ausgeführt, setzt aber keine clientseitige Installation der Anwendung voraus. Als eine einzige Voraussetzung für das neue System gilt ein Web Browser, der durch beliebige Betriebssysteme zur Verfügung gestellt wird. Dadurch wird das „Change and Configuration Management“ Problem reduziert. Die Wartbarkeit und die Benutzerfreundlichkeit des Systems werden wiederum erhöht.

4 Design und Konzeptionelle Entscheidungen

Basierend auf den in dem Analysekapitel beschriebenen Anwendungsfällen [3.1] und Anforderungen [3.1] werden in diesem Kapitel der Entwurf des Systems und die bei dem Entwurf getroffenen Entscheidungen vorgestellt.

Software-Architektur ist eine der wichtigsten Komponenten bei der Entwicklung eines Softwaresystems. In vielen Quellen sind unterschiedliche Definitionen des Begriffes „Software-Architektur“ zu finden[1]. In dem IEEE Standard [vgl. IEEEStd (2000)] ist die Architektur, wie folgt, definiert:

> The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

Die Modellierung von Software-Architekturen zielt primär auf die Betrachtung nicht funktionaler Aspekte von Software-Systemen. Ein zentrales Ziel ist die Bewertung der Qualität der modellierten Software-Systeme, bevor diese realisiert werden. Die Genauigkeit der Bewertungen sollte am später realisierten System überprüft werden, um die zukünftigen Vorhersagen zu verbessern.

In diesem Abschnitt werden mögliche Architekturen für Desktop- und Web-Anwendungen dargestellt und anhand ihrer Vor- und Nachteilen verglichen.

4.1 Schichtenarchitektur

Interaktive Anwendungen basieren auf der Schichtenarchitektur. Unter einer Schichtenarchitektur versteht man die Zerlegung eines Software-Systems in Komponenten, deren Abhängigkeitsbeziehungen untereinander durch die Architektur eingeschränkt sind. Jede Schicht repräsentiert eine bestimmte Abstraktionsebene, die die Dienste der darunterliegenden Schicht benutzt und der nächst höheren Schicht eigene Dienste anbietet. Bei den strikten

[1] Für weitere Informationen siehe u.a. [Hasselbring (2006)]

Schichtenarchitekturen greift jede Schicht nur auf die unmittelbar darunterliegende Schicht zu. Man unterscheidet zwischen Zwei-Schichten- und mehrschichtiger Architekturen.

4.1.1 Zwei-Schichten-Architektur

Bei einer Zwei-Schichten-Architektur, die auch als das grundlegende Client-Server-Modell bekannt ist, werden die Prozesse in einem verteilten System in zwei Gruppen eingeteilt. Ein Server ist ein Prozess, der Anforderungen von anderen Prozessen entgegen nimmt. Die Clients sind Prozesse, die einen Dienst von einem Server anfordern. Sie sind auslösende Elemente, d.h. sie können zu jedem beliebigen Zeitpunkt eine Anforderung senden und warten dann auf die Antwort des Servers. Diese Zusammenarbeit bezeichnet man als Anforderungs-/Antwort-Verhalten [vgl. Tanenbaum und van Steen (2002)]. Sie ist in Abbildung [4.1] dargestellt. Bei der Zwei-Schichten-Architektur kommen ein Fat-Client und ein Fat-Server zum Einsatz. Der Client übernimmt die Präsentation und die Manipulation der Daten (die Logik der Anwendung). Der Server kümmert sich um die Speicherung dieser Daten.

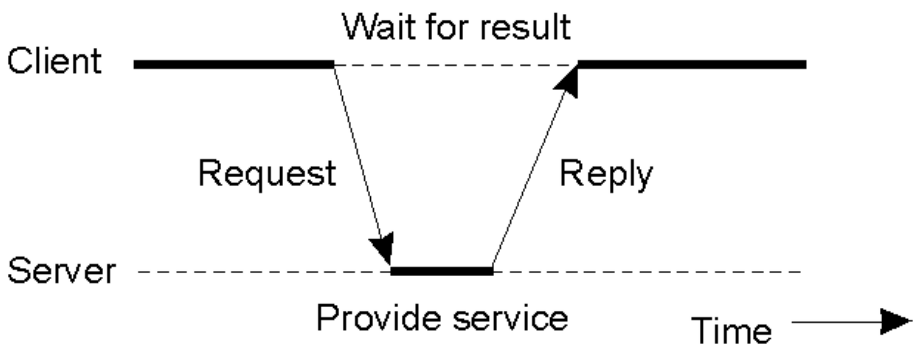

Abbildung 4.1: Allgemeine Zusammenarbeit zwischen einem Client und einem Server. Quelle: [Tanenbaum und van Steen (2002)]

Um das Client-Server-Modell gibt es zahlreiche Debatten und Kontroversen. Eines der größten Probleme bei vielen Anwendungen, die auf dem Client-Server-Modell basieren, ist keine klare Unterscheidung zwischen dem Client und dem Server getroffen werden kann. Da sich ein Server wie ein Client verhalten kann, heißt es, er kann ein Client anderer Server sein, gibt es häufig keine klare Unterscheidung.

4.1.2 Mehrschichtige Architektur

Aus der Betrachtung, dass viele Client-Server-Anwendungen den Zugriff von Benutzern auf Datenbanken unterstützen sollen, wird die Unterscheidung zwischen den drei folgenden logischen Schichten getroffen [vgl. Tanenbaum und van Steen (2002)][2]:

- **Präsentationsschicht**
  Die Präsentationsschicht ist für die Repräsentation der Daten, Benutzereingaben und die Benutzerschnittstelle verantwortlich. Sie ermöglicht den Benutzern, mit Anwendungen zusammenzuarbeiten.
- **Verarbeitungsebene**
  Die Verarbeitungsebene wird auch als die Anwendungslogikschicht bezeichnet. Sie beinhaltet alle Verarbeitungsmechanismen. Mit anderen Worten vereint sie die Intelligenz einer Anwendung. Durch die Verarbeitunsebene kann eine klare Trennung zwischen der Anwendungslogik und der Darstellung erfolgen. Die Anwendungslogikschicht kann auch wiederum in mehrere Schichten unterteilt werden. In diesem Fall spricht man von einer mehrschichtigen Architektur.
- **Datenhaltungsschicht**
  Die Datenhaltungsschicht verwaltet die eigentlichen Daten, mit denen die Anwendungen arbeiten.

Diese Unterscheidung ist als konzeptionelle Drei-Schichten-Software-Architektur bekannt. Die Schichten spiegeln die grundsätzlichen Aufgaben von Software-Systemen wider. Im Gegensatz zu der Zwei-Schichten-Architektur erlaubt die Drei-Schichten-Architektur verschiedene Lastverteilungskonzepte, wie z.B. die Verteilung der Last auf mehrere Anwendungsserver und verfügt über eine höhere Performance und Ausfallsicherheit. Sie ermöglicht eine klare Trennung zwischen der Darstellung und der Anwendungslogik. Die Abbildung [4.2] zeigt, dass ein Server in der Drei-Schichten-Software-Architektur manchmal auch als Client arbeiten kann. In diesem Fall kann die Anwendungslogik über mehrere Clients und Server verteilt sein. Als Weiteres sind folgende Vorteile der Drei-Schichten-Architektur zu nennen:

- Wiederverwendbarkeit der bestehenden Dienste
- Entkopplung zwischen dem Datenspeicher und einem Client
- Die Komponentenbauweise ermöglicht das Ersetzen oder das Ändern einzelner Komponenten. Wiederum können Komponenten wie im Baukasten zusammengesetzt werden.
- Bei der Entwicklung können die einzelnen Schichten von verschiedenen Entwicklerteams unabhängig erstellt werden.

[2] Für weitere Informationen siehe u.a. [Coulouris u. a. (2002)] und [Bengel (2000)]

- Veränderungen haben Auswirkungen nur in einer Schicht.

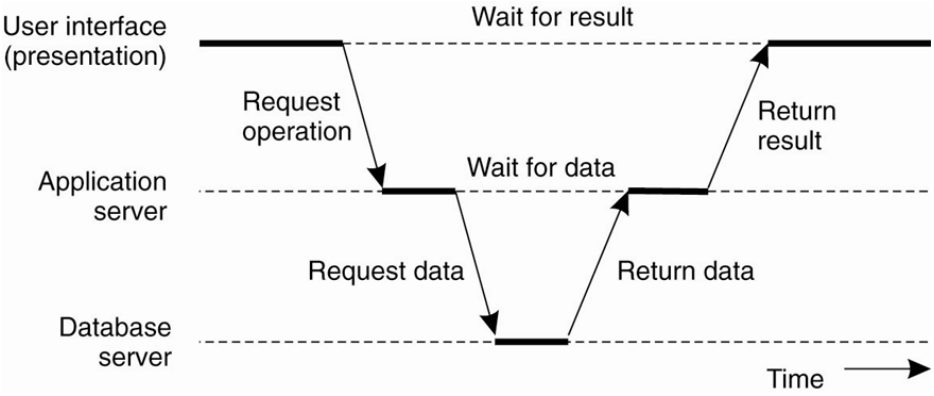

Abbildung 4.2: Ein Beispiel für einen Server, der als Client arbeitet. Quelle: [Tanenbaum und van Steen (2002)]

Die unterschiedlichen Aufteilungen von Clients und Servern sind in der Abbildung [4.3] gezeigt [vgl. Tanenbaum und van Steen (2002)]. Da diese Aufteilungen wesentliche Auswirkungen auf die Leistung, die Zuverlässigkeit und die Sicherheit des resultierenden Systems haben [vgl. Coulouris u. a. (2002)], werden sie an dieser Stelle diskutiert.

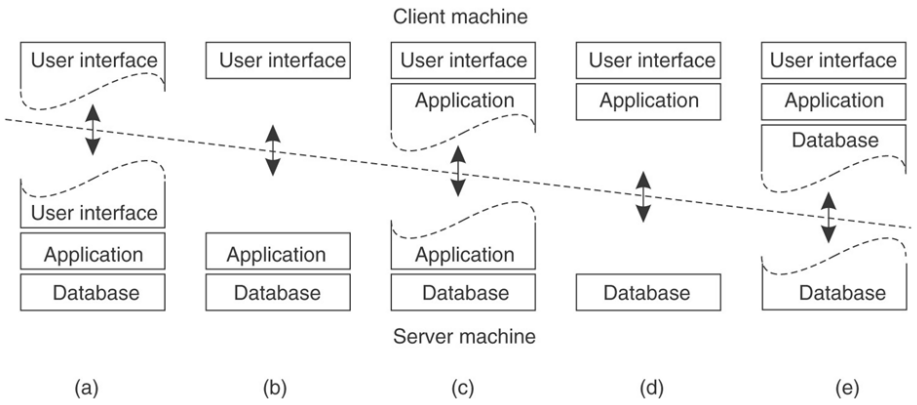

Abbildung 4.3: Alternative Client-Server-Anordnung. Quelle: [Tanenbaum und van Steen (2002)]

Null Client

Bei einem Null Client, wie die Abbildung [4.3](a) zeigt, liegt die Trennung zwischen der graphischen Benutzeroberfläche und dem Benutzerinterface. Den Applikationen wird in diesem

Fall nur eine entfernte Kontrolle über die Darstellung ihrer Daten gegeben. Der Client übergibt nur Tastaturanschläge und Mausbewegungen an den Server. Die bekannteste Technologie, die Null-Client Anordnung verwendet, ist X-Windows [vgl. Scheifler und Gettys (1986)].

Das hohe Transportvolumen von Ein-und Ausgaben über das Netz muss als ein Nachteil einer Null Client Anordnung genannt werden. Dies ist nur bei den Auskunfts-und Anzeigesystemen sinnvoll.

Thin Client

Eine Platzierung der gesamten Software auf der Benutzeroberfläche wird als Thin Client bezeichnet und ist in der Abbildung [4.3] (b) dargestellt. Bei dem Thin Client Konzept liegt die Trennung zwischen dem Benutzerinterface und der Anwendung. Der Client ist nur für das Management der Präsentation verantwortlich [vgl. Feinstein (2000)]. Der Server verwaltet dagegen die Anwendungslogik und die Daten [vgl. Lewandowski (1998)].

Das Thin Client Konzept bezeichnet eine Architektur, bei der die gesamte Datenverarbeitung und Datenspeicherung auf einem Server stattfinden. Als preisgünstige Endgeräte, die an eine Server-basierte Netzwerk-Architektur angeschlossen sind, benötigen die Clients weder leistungsstarke Prozessoren noch große Speicherkapazitäten. Alle Anwendungen laufen auf einem zentralen Server und nicht auf dem lokalen Rechner. Dadurch verringert sich der Aufwand bei den Neuinstallationen und Upgrades merklich - vor allem dann, wenn sehr viele User dieselbe Applikation verwenden. In diesem Fall ist der Installationsvorgang nur auf dem Server notwendig.

Die Thin Client Anwendungen haben Nachteile, die oft bei der Auswahl der Architektur eine große Rolle spielen. Da die Präsentationsdarstellung und -behandlung von der Präsentationslogik getrennt ist, ist ein Thin Client gezwungen, eine bidirektionale Kommunikation zu unterstützen. Da das Kommunikationsmodell der Thin Client Anwendungen auf dem Request-Response-Modell basiert, ist ein Thin Client dazu gezwungen, auf diesem unidirektionalen Kanal (vom Client zum Server) einen bidirektionalen Kanal zu simulieren.

Die Sicherheit ist in der Thin Client Anwendungen noch ein unterschätzter Faktor [vgl. Schäffer (2003)]. Problematisch wird es, wenn von einer Thin Client Anwendung auf die sicherheitsrelevanten Ressourcen eines Clients zugegriffen werden soll. Hierfür sind folgende Beispiele zu nennen: Zugriff auf Dateien oder Starten von Anwendungen, etc. Dies ist aber die Funktionalität, die unter Umständen entscheidend für die Akzeptanz einer Anwendung ist [vgl.Jern (1998)].

Die Wiederverwendbarkeit der Anwendungslogik ist ein großer Vorteil des Thin Client Konzeptes. Dadurch wird die Wartbarkeit des Systems verringert und die Entwicklungseffizienz

steigt. Der Einsatz des Model-View-Controller[3] Architekturmusters ist eine mögliche Grundlage des Thin Clients.

Applet Client

Bei einer Applet Client Anwendung liegt die Trennung nach dem Benutzerinterface. Der Unterschied zu einem Thin Client, wie die Abbildung [4.3] (c) zeigt, besteht darin, dass die Teile der Anwendungslogik mit eingeschlossen werden [vgl. Bengel (2000)]. Auf dem Client läuft die Anwendungslogik in Form eines Applets[4].

Ein Applet Client kann die Korrektheit und die Konsistenz der Benutzereingaben überprüfen und gegebenenfalls mit dem Benutzer kommunizieren. Ein Beispiel, bei dem die Anordnung aus der Abbildung [4.3] (c) verwendet wird, ist eine Textverarbeitung, wo die grundlegenden Bearbeitungsfunktionen auf der Client-Seite ausgeführt werden, wo sie für die lokal im Cache oder im Speicher befindlichen Daten arbeiten. Die komplexeren Werkzeuge, wie beispielsweise die Rechtschreib- und die Grammatikprüfung befinden sich aber auf dem Server [vgl. Tanenbaum und van Steen (2002)].

Fat Client

Da die Fat Client Systeme normalerweise einfach zu entwickeln sind, sind sie von vielen Softwareentwicklern besonders beliebt und bevorzugt. Merkmal dieser Client-Server-Architektur ist, dass ein Großteil der gesamten Anwendungen, wie in der Abbildung [4.3] (d) dargestellt ist, auf der Client Seite und nur ein relativ kleiner Anteil auf dem Server liegt.

Die Abbildung [4.3] (e) stellt die Anordnung dar, bei der ein Client teilweise die Daten verwaltet oder sie auf der Festplatte zwischenspeichert. Sucht man etwas beispielsweise im Web, kann ein Client einen Cache der zuletzt besuchten Webseiten auf der lokalen Festplatte anlegen [vgl. Tanenbaum und van Steen (2002)].

Da die Fat Clients für die Verarbeitung der Daten und teilweise für die Datenhaltung verantwortlich sind, ist eine höhere Anforderung an die Hardware gegeben. Aus diesem Grund sind die Fat Clients im Gegensatz zu den Thin Clients vollwertig ausgestattete und leistungsfähige Desktop-Computer. Die Fat Clients liefern einen Großteil der Prozessorleistung in den Client-Server Anwendungen.

Die Fat Clients bieten den Entwicklern die Möglichkeit komplexe graphische und interaktive Oberflächen zu entwerfen. Der Verzicht auf die reduzierte Datenkapselung ist dabei ein

[3] Für weitere Informationen über den Model-View-Controller Architekturmuster siehe u.a [Shan (1989)]
[4] Für weitere Informationen über Applets siehe u.a. [Roberts (2007)]

Nachteil. Je mehr Verantwortung ein Client über die Datenverarbeitung und die Datenhaltung übernimmt, desto mehr fordert er genauere Informationen über die Datenstruktur auf dem Server an [vgl. Lewandowski (1998)], [vgl. Jern (1998)]. Bei der Entwicklung muss man deswegen mit einem höheren Implemetierungsaufwand rechnen.

Die auf dem Fat Client Konzept basierten Anwendungen müssen lokal oder von einer zentralen Stelle installiert und konfiguriert werden. Dabei besteht das „Change and Configuration Management"Problem, das im Kapitel [1.1] angesprochen wurde. Das Vermischen von Präsentationen und Anwendungslogik führt zu der schlechteren Wartbarkeit des Systems.

4.2 Web Deployment

Eine der mögliche Lösungen des „Change and Configuration Management“ Problems (siehe Kapitel [1.1]) ist Web Deployment. Unter Web Deployment versteht man die Verteilung einer Desktop Anwendung über das Netzwerk. In diesem Abschnitt wird diese Art der Software-Verteilung anhand einer Technologie kritisch dargestellt.

4.2.1 Java Web Start

Java Web Start [Sun Microsystems (c)] ist eine Technologie von Sun Microsystems[5], die es ermöglicht, Java-Anwendungen[6] über das Netzwerk mit einem Klick zu starten.

Mit Java Web Start können die Anwendungen über das Netzwerk automatisch heruntergeladen, installiert und aktualisiert werden. Nur bei dem ersten Start wird eine Anwendung heruntergeladen. Die Installation der Software wird automatisch auf dem Client-Rechner durchgeführt. Alle Ressourcen werden auf dem Client-Rechner abgespeichert. Java Web Start hält sie lokal in einem speziellen Cache. Bei jedem weiteren Start der Software überprüft Java Web Start, ob die Anwendung aktualisiert werden soll. Das heißt, ob auf dem Server bereitgestellte Ressourcen geändert oder erweitert sind. Unter Ressourcen sind hier alle Bibliotheken, Konfigurations- und Property-Dateien, etc. zu verstehen. Die auf dem Server bereitgestellten Ressourcen müssen digital signiert[7] sein. Nur die geänderten oder neuen Ressourcen werden heruntergeladen und abgespeichert. Dadurch können komplizierte Installations- oder Aktualisierungsprozeduren vermieden werden.

[5] Sun Microsystems, Inc. http://www.sun.com ; Zugriffsdatum: 30.01.2008
[6] Desktop Anwendungen, die in Programmiersprache Java entwickelt sind
[7] Für wietere Informationen über digitale Signatur siehe u.a. [Eckert (2004)]

Die HTML-Links, über die die Anwendungen gestartet werden, sind ganz normale HTML-Links, die statt auf eine andere Webseite zu zeigen, jedoch die Verbindung mit einer speziellen Konfigurationsdatei herstellen. Diese Konfigurationsdatei ist eine JNLP-Datei[8] (MIME-Type „application/x-java-jnlp-file jnlp“), die für eine Anwendung in XML-Notation geschrieben und auf dem Webserver abgelegt ist. In der Konfigurationsdatei werden alle benötigten Ressourcen für die Client-Anwendungen spezifiziert. Sie enthält die Informationen, wie den Ablageort der Jar-, der anderen Konfigurations- und Property-Dateien, den Namen der Hauptklasse einer Anwendung und alle zusätzliche Parameter für das auszuführende Programm. Der Aufbau einer JNLP-Datei ist in der Abbildung [4.4] dargestellt.

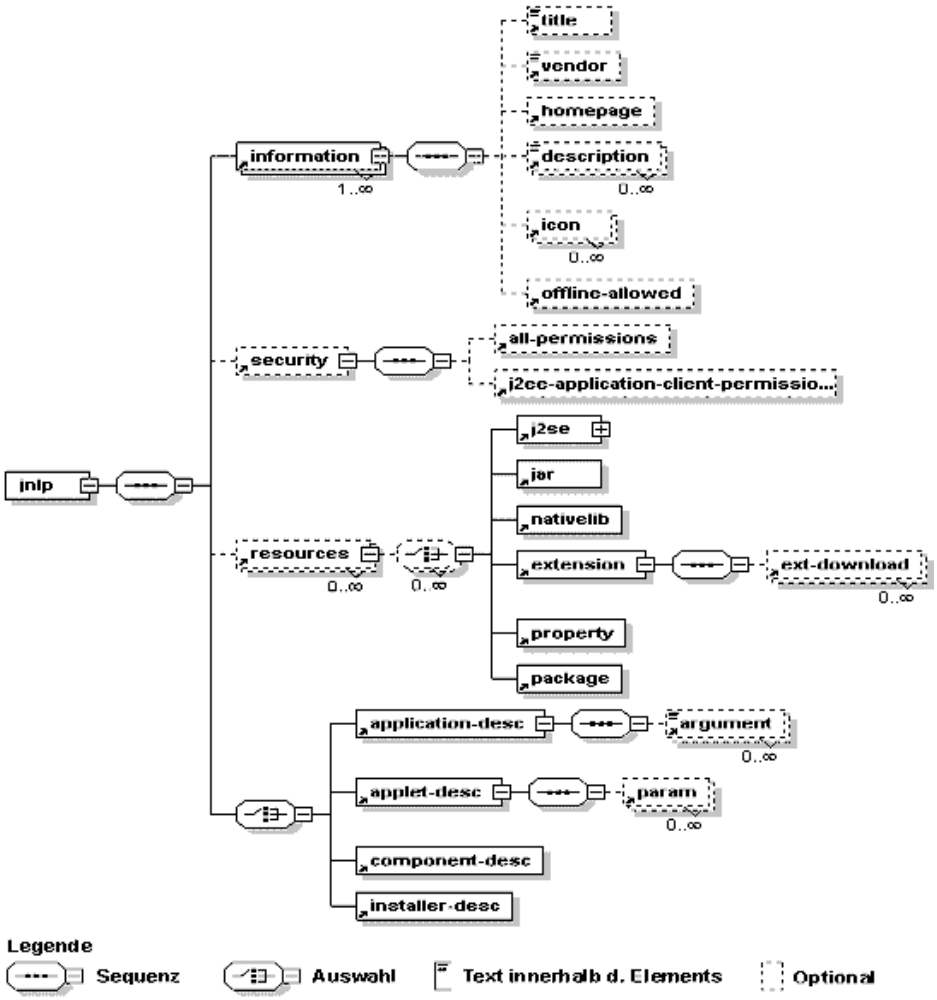

Abbildung 4.4: Der Aufbau der JNLP-Datei. Quelle: [Middendorf u. a. (2002)]

[8] JNLP - Java Network Launching Protocol [Sun Microsystems (a)]

An dieser Stelle können Vorteile bei dem Einsatz von Java Web Start Technologie für die Anwender und Entwickler bestimmt werden:

- **Vorkonfigurierte Anwendungen**
  Da in der JNLP-Datei alle benötigten Ressourcen konfiguriert werden können, ist es möglich, die Konfigurationsdateien benutzerspezifisch zu generieren. Dadurch kann der „Change and Configuration Management“ Aufwand reduziert werden.
- **Caching**
  Da Java Web Start die Anwendungen lokal im Cache hält, stehen die Anwendungen bei jedem Start sofort zur Verfügung. Somit ist auch Offline-Szenario möglich.
- **Automatisches Update**
  Sowohl für die Anwender, als auch für die Entwickler ist das Updaten der Anwendungen kein komplizierter Prozess, weil Java Web Start eine automatische Überprüfung bei jedem Start der Anwendungen durchführt.
- **Einfaches und intuitives Starten**
  Zu einer heruntergeladenen Anwendung können Verknüpfungen auf dem Desktop oder dem Startmenü erstellt werden.
- **Sichere Sandbox**
  Die Anwendungen laufen in einer geschützten Sandbox, die den Zugriff auf die lokalen Ressourcen grundsätzlich verbietet. Zugriffsrechte auf die lokalen Ressourcen können in der JNLP-Datei konfiguriert werden (siehe Abbildung [4.4], *security*).

Das Cashing, das als ein Vorteil schon aufgelistet wurde, bringt auch einen großen Nachteil im Rahmen des im Kapitel [1.1] vorgestellten Szenariums mit. Sollte ein Mitarbeiter auf einem durch den Kunden zur Verfügung gestellten Rechner arbeiten, werden Java Web Start Anwendungen lokal auf dem Rechner des Kunden abgespeichert. Dadurch entsteht ein Sicherheitsproblem, da der Code (Jar-Dateien, etc.) auch nach dem Beenden des Programms auf der Client Seite vorhanden ist und die Anwendungen weiter benutzt werden können. Da die Kunden in dem Szenario keine Lizenzen für die Nutzung der Anwendungen besitzen, ist dies meistens nicht wünschenswert. Das Vorhandensein des Codes auf einem Kundenrechner kann weitere Konsequenzen haben. Diese Konsequenzen werden allerdings im Rahmen dieser Arbeit nicht weiter diskutiert werden.

Man darf auch nicht vergessen, dass die Benutzung von Java Web Start die Java Runtime Environment auf dem Client-Rechner voraussetzt. Die Notwendigkeit der Installation der Java Runtime Environment kann aber wieder zu dem „Change and Configuration Management“ Problem führen.

4.3 Rich Internet Application Architektur

Bei den Rich Internet Applications (RIA) handelt es sich um die serverbasierte Implementierungen von Anwendungen, wobei jedoch durch verschiedene Technologien dem Anwender einen erhöhteren Benutzerkomfort geboten wird. Das Entwerfen von RIAs mit den Web

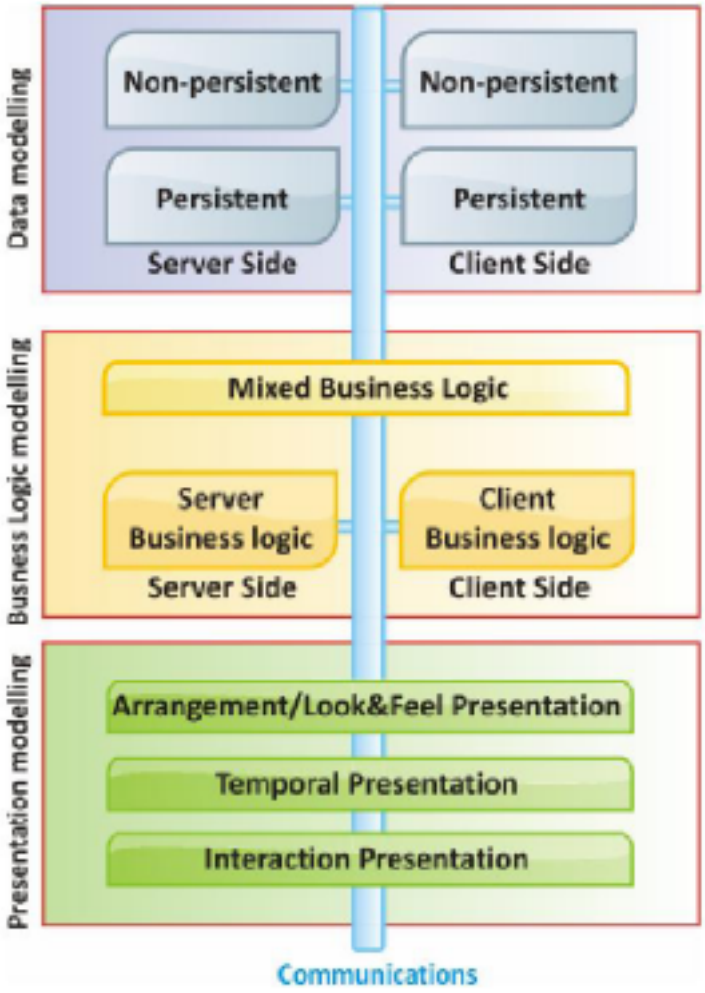

Abbildung 4.5: Konzepte in RIA Design. Quelle: [Preciado (5–6 Oct. 2007)]

Engineering Konzepten verlangt die Anpassung der Entwicklung von traditionellen Web Anwendungen. Die neue clientseitige Kapazität, neue Eigenschaften der Darstellungsschicht und verschiedene Kommunikationsflüsse zwischen dem Client und dem Server müssen mitberücksichtigt werden. In [Preciado (5–6 Oct. 2007)] sind vier Phasen analysiert, die Rich Ineternet Applications charakterisieren:

- Data modeling
- Business logic modeling
- Presentation modeling
- Communication modeling

Die vier Phasen sind in der Abbildung [4.5] gezeigt. Solche Phasen verlangen die Erweiterung des Modells[9] von traditionellen Web-Anwendungen.

4.3.1 Client-Server Anordnung in Rich Internet Applications

Die Architektur einer Rich Internet Application wird in Schichten aufgeteilt. So wie bei den Desktop Anwendungen (siehe Kapitel [4.1]) sind die Darstellungs-, Logik- und Datenschicht in der RIA Architektur vorhanden. Diese Schichtung ermöglicht die klare Trennung zwischen der Darstellung und Anwendungslogik.

Die Abbildung [4.6] zeigt die unterschiedlichen Aufteilungen von Servern und Clients in der Rich Internet Application Architektur. Diese Aufteilungen sind ähnlich zu den Client-Server-

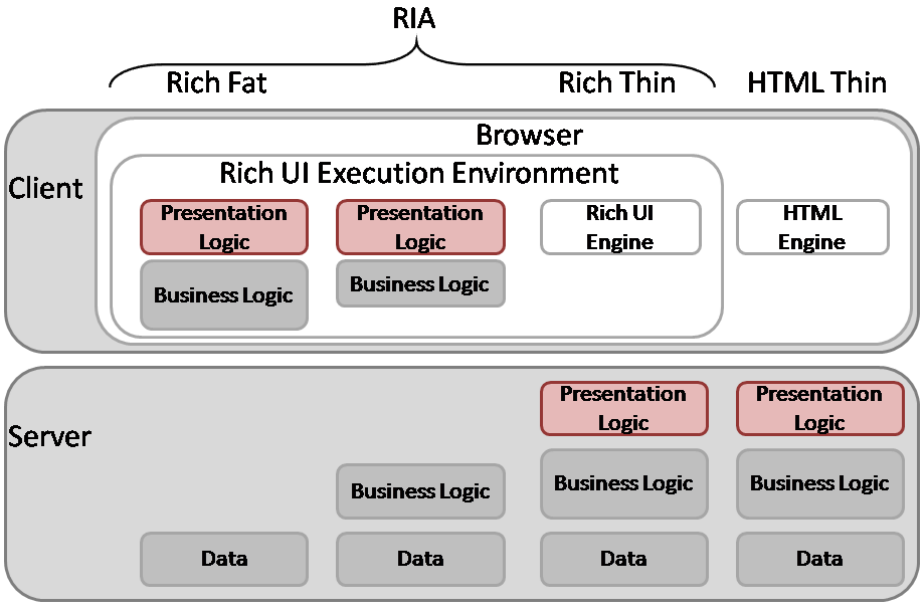

Abbildung 4.6: Client Server Anordnung in Rich Internet Applications

Anordnungen bei der Entwicklung von den Rich Desktop Anwendungen, die in der Abbildung

[9] Für weitere Informationen über Design von RIAs siehe u.a. [Preciado (26 Sept. 2005)]

[4.3] dargestellt sind. Da die unterschiedlichen Client-Server-Anordnungen (z.B. Thin Client, Fat Client) in den Desktop Anwendungen im Kapitel [4.1.2] diskutiert wurden, ist an dieser Stelle wegen der großen Ähnlichkeit nicht notwendig, die Client-Server-Anordnungen bei den Rich Internet Applications zu beschreiben. Der Hauptunterschied zu den Rich Client Anwendungen bei der Client-Server-Anordnung besteht darin, dass die Business Logik einer Rich Internet Application meistens zwischen dem Server und dem Client aufgeteilt wird. Da Client-Rechner heutzutage keine langsamen Rechner sind, können einige Teile der Business Logik clientseitig ausgeführt werden. Dadurch kann ein Server entlastet werden und schnellere Antwortzeiten können erreicht werden.

4.4 Systemarchitektur

In diesem Abschnitt wird die gesamte Systemarchitektur beschrieben. Dabei werden einige Komponenten detailliert beschrieben und andere, die für diese Arbeit keine wichtige Rolle spielen und nur als bestehenden Komponenten verwendet werden, nur definiert.

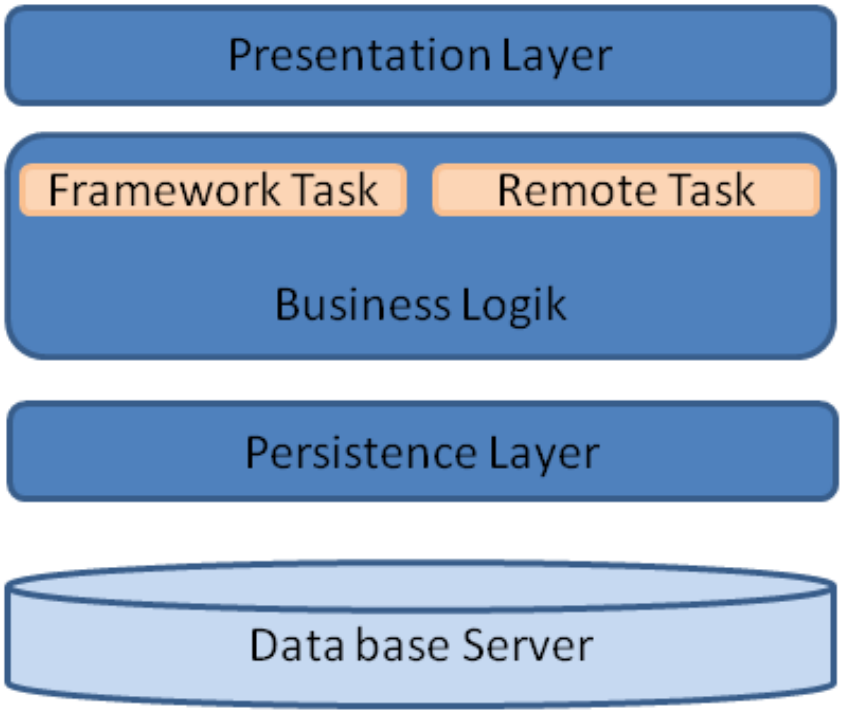

Abbildung 4.7: Architektur existierender Rich Client Anwendung

Da es in dieser Arbeit um eine Migration einer Rich Client Anwendung geht, ist hier als Erstes

die Architektur der existierenden Rich Client Anwendung vorzustellen. In der Abbildung [4.7] ist zu sehen, dass die Architektur der existierenden Rich Client Anwendung eine klassische mehrschichtige Architektur ist. In der existierenden Rich Client Anwendung werden alle Operationen clientseitig ausgeführt, nur die Datenhaltungschicht ist auf einem Data Base Server verteilt.

Die in der Abbildung [4.7] dargestellte Architektur ermöglicht die Wiederverwendbarkeit der bestehenden Dienste. Der Datenspeicher und die Clients sind entkoppelt. Basierend auf dieser Architektur, auf der existierenden Rich Client Anwendung selbst und auf den im Kapitel [3] definierten Anforderungen wird eine Architektur für die Migration der existierenden Rich Client Anwendung auf Code on Demand Technologie entworfen.

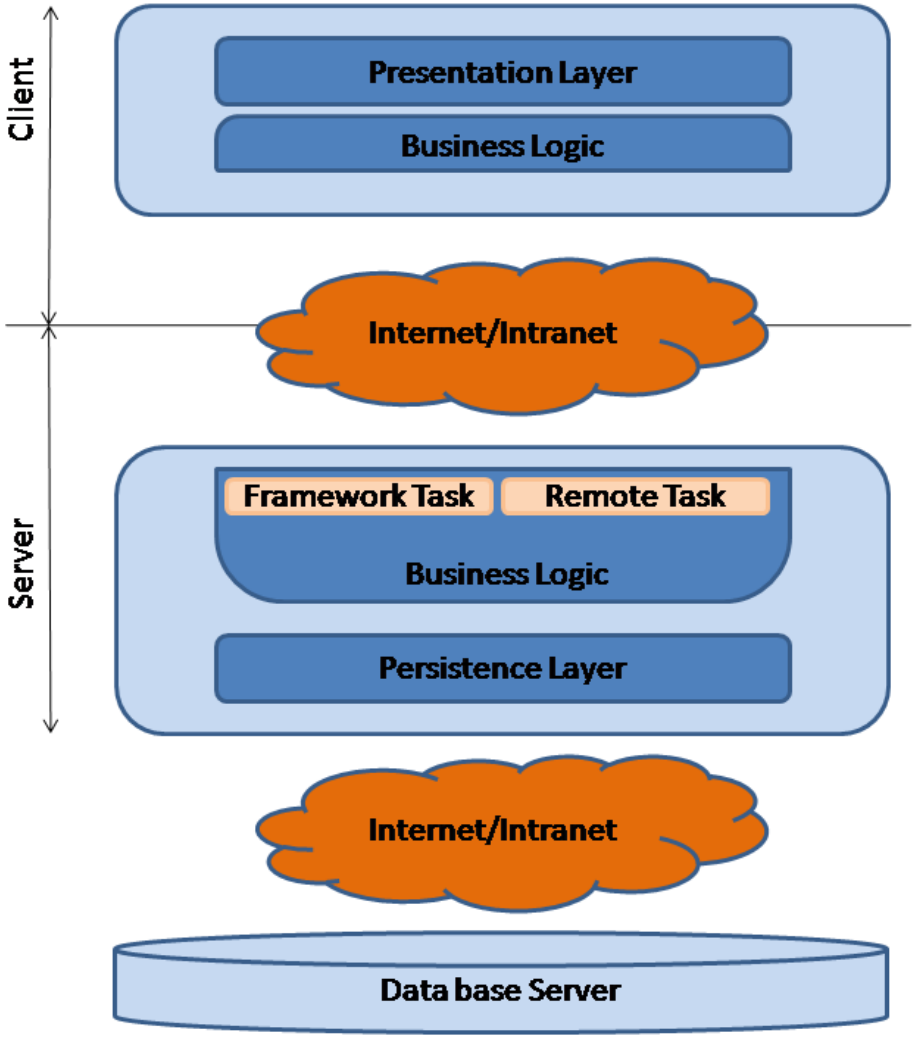

Abbildung 4.8: Systemarchitektur

Die Abbildung [4.8] zeigt eine abstrakte Darstellung der gesamten Systemarchitektur einer Rich Internet Application. Das heißt, sie ist nicht nur für das ausgewählte Beispielsystem anwendbar, sondern auch allgemein für Rich Client Anwendungen, die auf Code on Demand Technologie migriert werden können. An dieser Stelle werden die wichtigsten Komponenten der Systemarchitektur beschrieben. Dabei werden die Unterschiede zwischen der existierenden und der neuen Architektur detailliert diskutiert.

4.4.1 Data Base Server

Als die Datenhaltungsschicht (Data Base Server) wird in dem System der Topgallant®Information Server [vgl. Atlantec Enterprise Solutions (2001a)] eingesetzt. In dem Topgallant®Information Server werden Informationen dauerhaft abgespeichert. Er besteht aus einem potentiell verteilten Directory Server und adapter client APIs[10] für verschiedene Operationen. Das Datenmodell aller signifikanten Typen leitet von dem Information Object ab und ist im Enterprise Reference Model (ERM) definiert [vgl. Atlantec Enterprise Solutions (2003a)].

Da sich der Autor in Rahmen dieser Arbeit nicht mit der Datenhaltungsschicht beschäftigt und der Topgallant®Information Server als die Datenhaltungsschicht von der Firma Atlantec Enterprise Solutions GmbH vorausgesetzt ist, wird der Topgallant®Information Server nur als solcher verwendet und nicht weiter in dieser Arbeit diskutiert. Auch das Application Programming Interface, das die Kommunikation mit dem Topgallant®Information Server und das Verwalten von Daten zur Verfügung stellt, ist für diese Arbeit irrelevant und wird hier nicht weiter untersucht.

4.4.2 Persistence Layer

In der Persistenzschicht ist einen ein abstraktes Topgallant®API definiert, das das Laden, Speichern, Aktualisieren und Löschen von Objekten ermöglicht. Das Topgallant®API ist unabhängig von der konkreten Implementierung, d.h. es ist dem API nicht anzusehen, welche Datenhaltungsschicht verwendet wird. Über die Persistenzschicht kommuniziert die Anwendung mit der Datenhaltungsschicht (Data Base Server). Wie die Datenhaltungsschicht wird die Persistenzschicht im Rahmen dieser Arbeit nicht weiter diskutiert.

[10] Bestandteil von Topgallant®Adapter Developer Kit

4.4.3 Presentation Layer

Die in der Abbildung [4.8] als Presentation Layer genannte Darstellungsschicht ist die direkte Schnittstelle zum Benutzer. Hier wird die Anfrage empfangen. Die gesendeten Informationen werden ausgewertet und es wird die auszuführende Aktionen bestimmt. Da sich das gesamte System auf Code on Demand Technologie basiert, werden einige logische Aufgaben auf der Client Seite erledigt.

Die Darstellungschicht stellt fertige graphische Komponenten für einen Web Client dar. Da die graphischen Komponenten typischerweise ein Bestandteil eines Frameworks sind, wird an dieser Stelle auf deren genaue Beschreibung verzichtet. Einige Frameworks sind in dem Kapitel [2.4] detaillierter beschrieben.

Als wichtige Voraussetzung für die Darstellungsschicht gilt es, dass die neue Rich Internet Application mit ihrem Graphical User Interface kaum Neuerungen für Endnutzer in ihrer Arbeit bringt und die Benutzer ihre Aufgaben ohne große Schulungsaufwand erledigen können.

4.4.4 Business Logic

Die Business Logic Schicht stellt die Anwendungsfunktionalität zur Verfügung. Bei der Entwicklung einer neuen Rich Internet Application werden keine Änderungen an der Funktionalität des gesamten Systems vorgenommen. Die Abbildung [4.8] stellt die Architektur der Rich Internet Application. Es ist zu sehen, dass im Vergleich zu der Architektur der existierenden Rich Client Anwendung, die in der Abbildung [4.7] dargestellt ist, große Änderungen genau auf der Business Logik Schicht vorgenommen werden. Diese Änderungen bedeuten aber keine Änderungen in der Funktionalität des Systems.

Die Architektur der neuen Rich Internet Application (siehe Abbildung [4.8]) stellt genau wie die Architektur der existierenden Rich Client Anwendung (siehe Abbildung [4.7]) die Framework- und RemoteTask Schnittstellen zur Verfügung. Die Tasks werden in dem System für Aktivität, wie das Laden oder das Speichern von Objekten, verwendet und können synchron oder asynchron ablaufen. Ein Task (Aktivität), wie die Abbildung [4.9] zeigt, wird nach Black-Box Prinzip implementiert. Dadurch wird Information Hading unterstützt. Dies bedeutet, das nach Außen nur eine Schnittstelle sichtbar ist, und die konkrete Aktivitäten diese Schnittstelle implementieren.

Da Client Rechner heutzutage schnell sind und leistungsfähige CPU und Arbeitsspeicher besitzen, kann die Business Logik des gesamten Systems zwischen einem Client und dem Server aufgeteilt werden. Dies bedeutet, dass einige Teile der Business Logik clientseitig ausgeführt werden können.

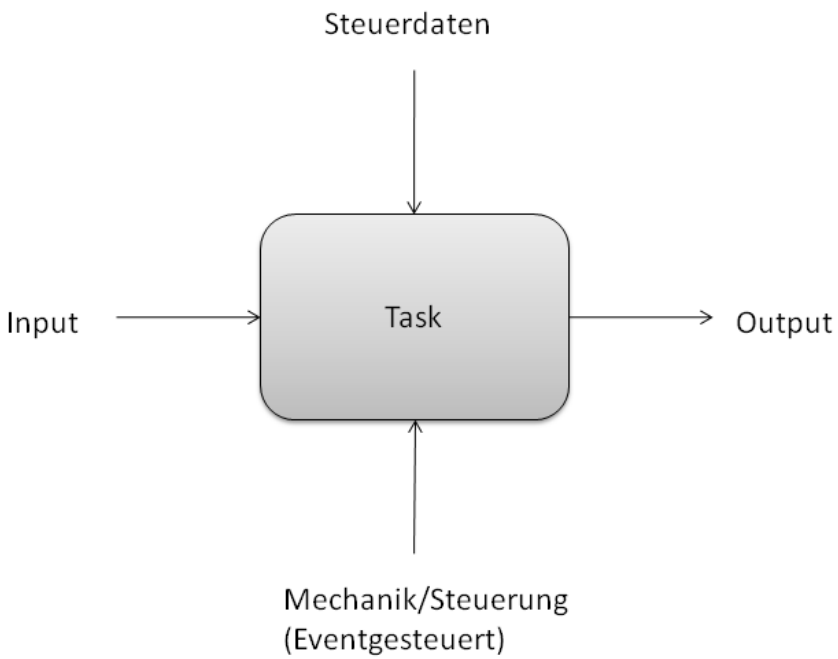

Abbildung 4.9: Business Logik Task. Quelle:[Atlantec Enterprise Solutions (2001b)]

Als Metapher für diese Arbeit wurde ein Szenario ausgewählt, in dem ein Mitarbeiter/Ingenieur rund um die Welt reist und seine Arbeit bei den Kunden vor Ort erledigt. Es ist nicht garantierbar, dass die Bandbreite der Internet Verbindung überall groß ist. Das System muss auch bei einer langsamen Internet Verbindung korrekt und ohne große Verzögerungen arbeiten. Bei der Ausführung der Business Logik auf der Client Seite ist keine Kommunikation zu dem Applikation Server notwendig. Dies bedeutet, dass die Internet Verbindung und deren Geschwindigkeit in diesen Fällen keine große Rolle spielen wird.

Die Funktionalität, die auf der Client Seite ausgeführt werden kann, wird im folgenden Abschnitt [4.5] beschrieben. Da einige Teile der Business Logik clientseitig ausgeführt werden, werden die Benutzerinteraktion und die Kommunikation zwischen einem Client und dem Server für den Nutzer in schnelleren Antwortzeiten erfolgen.

4.5 Trennung der Funktionalität

Das Thema dieser Arbeit ist die Migration einer Rich Client Applikation auf Code on Demand Technologie. Daraus folgt, dass eine Web Anwendung auf Basis einer existierenden Desktop Anwendung entwickelt werden soll. Basierend auf der Anforderungsanalyse und

der Systemarchitektur werden in diesem Abschnitt einige Einzelheiten beleuchtet, die bei der Realisierung zu beachten sind.

4.5.1 Clientseitige Funktionalität

Bei der Entwicklung einer Web Anwendung ist zu entscheiden, welche Funktionalität auf der Server- und welche auf der Client-Seite ausgeführt wird. Diese Entscheidung ist sehr wichtig, da dies zu unterschiedlichen Antwortzeiten des gesamten Systems führen kann. Werden die Funktionen auf der Server Seite ausgeführt, heißt das, dass der Client eine Anfrage senden soll. Der Server wird die Anfrage bearbeiten und eine Antwort an den Client senden. Diese Client-Server Kommunikation kann die Zeit in Anspruch nehmen. Abhängig von dem Funktionsumfang kann es zu längeren Wartezeiten auf der Client Seite führen. Es ist auch zu berücksichtigen, dass die Bandbreite nicht überall groß ist und aus diesem Grund der Transport der Daten langsam sein kann. Die längeren Antwortzeiten sind bei interaktiven Anwendungen nicht akzeptabel.

Code on Demand Technologie ermöglicht durch das Laden des Codes und dessen Ausführung auf der Client Seite, die Business Logik zwischen dem Server und einem Client aufzuteilen. An dieser Stelle wird einige Funktionalität zusammengefasst, die auf Client-Seite ausgeführt werden kann:

- **Verwalten von benutzerspezifischen Daten**
  Um mit den Daten arbeiten zu können, stellt ein Benutzer als Erstes eine Verbindung zu einem Datenbankserver her. Es ist realistisch, dass ein Benutzer gegen mehrere Datenbankserver arbeiten soll, um seine Aufgabe zu erledigen. Die Verbindungen zu einem oder dem anderen Datenbankserver sind benutzerspezifisch und können auf der Client Seite verwaltet werden.
- **Überprüfung von Benutzereingaben**
  Die Interaktion mit einem Benutzer, sei es die Überprüfung der Eingabe von Benutzernamen und Passwort, oder ob der Benutzer eine korrekte Auswahl von notwendigen Objekten und deren Eigenschaften getroffen hat etc., kann auf der Client-Seite stattfinden.
- **Intelligente Vorschläge**
  Während der Eingabe von Suchkriterien können dem Benutzer intelligente Vorschläge gemacht werden. Soll nach Einträgen gesucht werden, die die bestimmten Werte bei denen Eigenschaften haben, können Eigenschaftennamen als Vorschläge gemacht werden. Dies ist eine clientseitige Funktionalität.
- **Metadaten**
  Bei dem Erzeugen oder das Editieren von Objekten sind durch das Modell bestimmte

Eigenschaften zu setzen. Einige Eigenschaften dürfen nur durch das Modell bestimmte Werte haben. Die notwendigen Eigenschaften und die vordefinierten Werten sind als Metadaten in einem Katalog vorhanden. Da die Metadaten meistens keine großen Objekte sind und sehr schnell transportiert werden können, wird deren einmaliges Laden keine Verzögerung beeinflussen. Danach stehen die Metadaten auf der Client Seite zur Verfügung und die Funktionalität, die sie benötigt, clientseitig ausgeführt werden kann.

Durch clientseitige Funktionalität werden die Antwortzeiten des gesamten Systems reduzieren. Die unterschiedliche Bandbreite spielt bei der Ausführung der Funktionalität auf der Client Seite keine große Rolle.

4.5.2 Client-Server Interaktion

Das Konzept von Code on Demand Technologien besagt, dass der Code teilweise auf der Client Seite ausgeführt wird. Die Abbildung [4.10] veranschaulicht die Interaktion zwischen dem Server und einem Client am Beispiel von einem Anwendungsfall [3.1.6]. Sie zeigt, dass der Client, nur wenn es nötigt ist, mit dem Server interagiert und viele Code-Aufrufe clientseitig ablaufen.

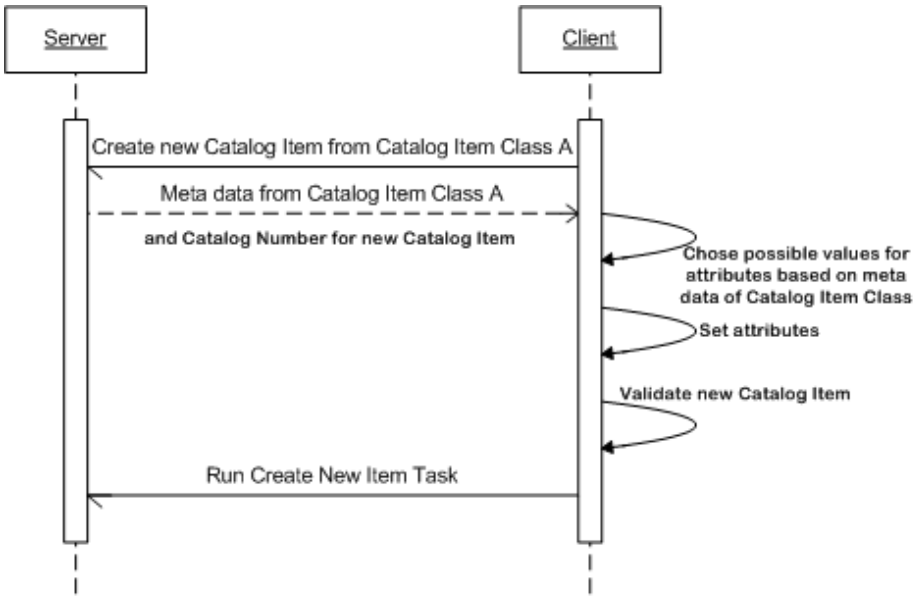

Abbildung 4.10: Interaktion zwischen dem Server und einem Client

4.5.3 Sicherheit

Im Rahmen dieser Arbeit wird sich der Autor nur begrenzt mit dem Thema Sicherheit beschäftigen. Die Sicherheit ist aber ein sehr wichtiger Aspekt der Softwareentwicklung. Da eine Web Anwendung meistens über das Internet geladen wird, spielen die Sicherheitsfragen eine sehr große Rolle.

Als mögliche Lösungen für die Sicherheit einer Web Anwendung sieht der Autor die Verwendung von Virtual Private Network [RFC2764] oder HyperText Transfer Protocol Secure (HTTPS) [Rescorla (2000)].

4.6 Fazit

In diesem Kapitel wurden die Architektur und das Design der auf der Basis von einer existierenden Rich Client Anwendung zu entwickelnden Web Anwendung vorgestellt. Es wurden konzeptionelle Entscheidungen getroffen.

Wie kann das „Change and Configuration Management“ Problem bei dem Einsatz von Software gelöst oder zu mindestens reduziert werden? Diese Frage war eine der wichtigsten Fragen, die in diesem Kapitel beantwortet werden sollte. Sollte Web Deployment (siehe Kapitel [4.2]) eingesetzt werden oder eine Web Anwendung basieren auf der existieren Rich Client Anwendung mittels Code on Demand Technologien entwickelt werden?

Basierend auf den in den „Analyse“ (Kapitel [3]) und „Design und Konzeptionelle Entscheidungen“ (Kapitel [4]) Kapiteln gewonnenen Erkenntnissen, ist die Migration der existierenden Rich Client Anwendung auf Code on Demand Technologie durchzuführen. Dabei sollte eine Web Anwendung mittels Code on Demand Technologie nach den State-of-the-Art-Entwurfsprinzipien entwickelt werden. Dass die Entwicklung solcher Anwendungen nach den State-of-the-Art-Entwurfsprinzipien möglich ist, bestätigen die Untersuchungen von Virginio Carfagno, die von ihm im Rahmen seiner Diplomarbeit [Carfagno (2007)] durchgeführt wurden.

5 Evaluierung

Nachdem das gesamte System analysiert und designt worden war, wurde die Migration der Rich Client Anwendung auf Code on Demand Technologie durchgeführt. Bei der Migration wurde eine Web Anwendung mittels des Java-basierte Echo Frameworks entwickelt. In diesem Kapitel wird das System evaluiert und die gefundenen Ergebnisse werden bewertet. Dabei sollten folgende Fragen beantwortet werden:

1. Was ändert sich für einen Nutzer durch den Einsatz einer Web Anwendung statt der existierenden Rich Client Anwendung?
2. Worauf sollte ein Entwickler bei dem Entwurf und der Entwicklung einer auf Code on Demand Technologie basierten Web Anwendung achten?
3. Welche Vor- und/oder Nachteile bringt die Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie für ein Unternehmen?

5.1 Akzeptanz durch Nutzer

Da ein Endnutzer bei der Software Entwicklung immer im Vordergrund steht, ist es sehr wichtig, seine Akzeptanz für die neue Anwendung zu gewinnen. Die Benutzer müssen auf den bei der Einführung einer neuen Web Anwendung zusätzlichen Aufwand, der sich in Grenzen halten soll, und die notwendigen Schulungsmaßnahmen vorbereitet werden.

Nach der Migration einer Rich Client Anwendung auf Code on Demand Technologie wird eine neue Web Anwendung für die Endnutzer zur Verfügung gestellt. Dabei ist es wichtig, dass die Funktionalität der Web Anwendung der Funktionalität der bestehenden Desktop Rich Client Anwendung, so nahe wie möglich, entspricht. Auch das neue User Interface soll ähnlich zu dem Bestehenden bleiben. Kann ein Endbenutzer die existierende Rich Client Anwendung bedienen, wird es ihm nicht aufwändig die neue Web Anwendung verwenden zu können. Da die große Ähnlichkeit des User Interfaces zu dem alten System vorhanden und die Funktionalität der existierenden Anwendung gleich geblieben ist, wird der Schulungsaufwand minimal bleiben und dadurch von den Endbenutzern akzeptiert.

Der nächste relevante Vorteil für einen Endbenutzer ist die Flexibilität. Ein Benutzer kann die Web Anwendung überall verwenden und seine Arbeit überall verrichten, gleichgültig, wo er sich zu dem Zeitpunkt befindet. Die Web Anwendung setzt keine Installation voraus. Als einzige Voraussetzung gilt ein Web Browser. Da alle Betriebssysteme einen Web Browser besitzen, ist diese Voraussetzung ohne den zusätzlichen Aufwand für einen Endbenutzer erfüllt. Dadurch wird das „Change and Configuration Management“ Problem reduziert. Von einem bestehenden Internetzugang wird in diesem Fall ausgegangen.

Bei den Desktop Anwendungen spielt die Betriebsystemunabhängigkeit eine große Rolle. Die gleiche Rolle spielt die Browserunabhängigkeit bei den Web Anwendungen. Da die Web Anwendung von den unterschiedlichen Endbenutzern verwendet wird, ist zu berücksichtigen, dass die Endbenutzer mit unterschiedlichen Web Browsern arbeiten können. Die mittels Echo Framework entwickelte Web Anwendung wurde mit der Verwendung von fünf Web Browsern getestet. Die fünf folgenden Abbildungen veranschaulichen, dass das User Interface in allen der fünf Web Browser gleich aussieht. Diese fünf Web Browser (Internet Explorer v. 7.0[1], Internet Explorer v. 6.0[2], Firefox[3], Opera[4] und Safari[5]) wurden laut W3CSchools[6] im Dezember 2007 von ca. 94% aller Internetnutzer verwendet.

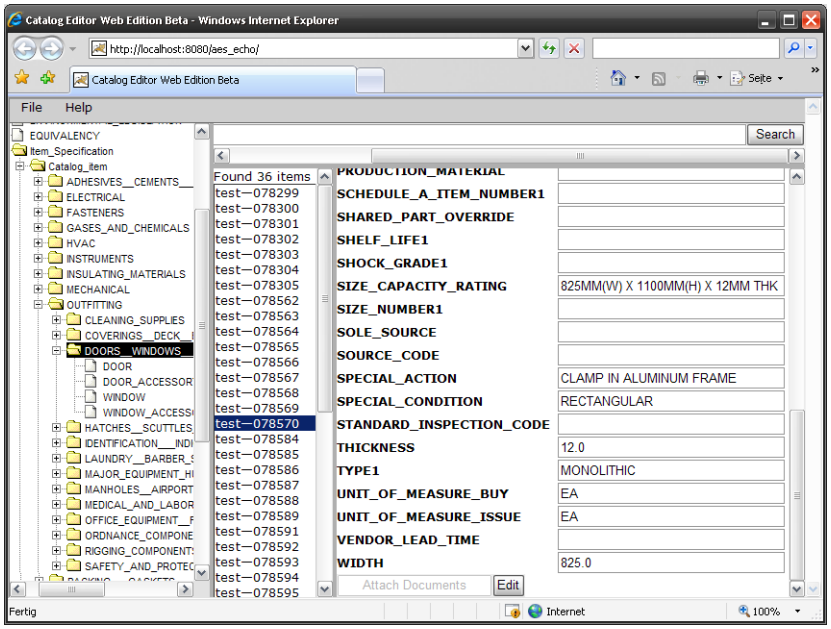

Abbildung 5.1: Web Anwendung im Browser Internet Explorer v. 7.0

[1] http://www.microsoft.com/windows/products/winfamily/ie/default.mspx ; Zugriffsdatum: 30.01.2008
[2] http://www.microsoft.com/windows/ie/ie6/default.mspx ; Zugriffsdatum: 30.01.2008
[3] http://www.mozilla.com/firefox/ ; Zugriffsdatum: 30.01.2008
[4] http://www.opera.com/ ; Zugriffsdatum: 30.01.2008
[5] http://www.apple.com/safari/ ; Zugriffsdatum: 30.01.2008
[6] http://www.w3schools.com/browsers/browsers_stats.asp ; Zugriffsdatum: 30.01.2008

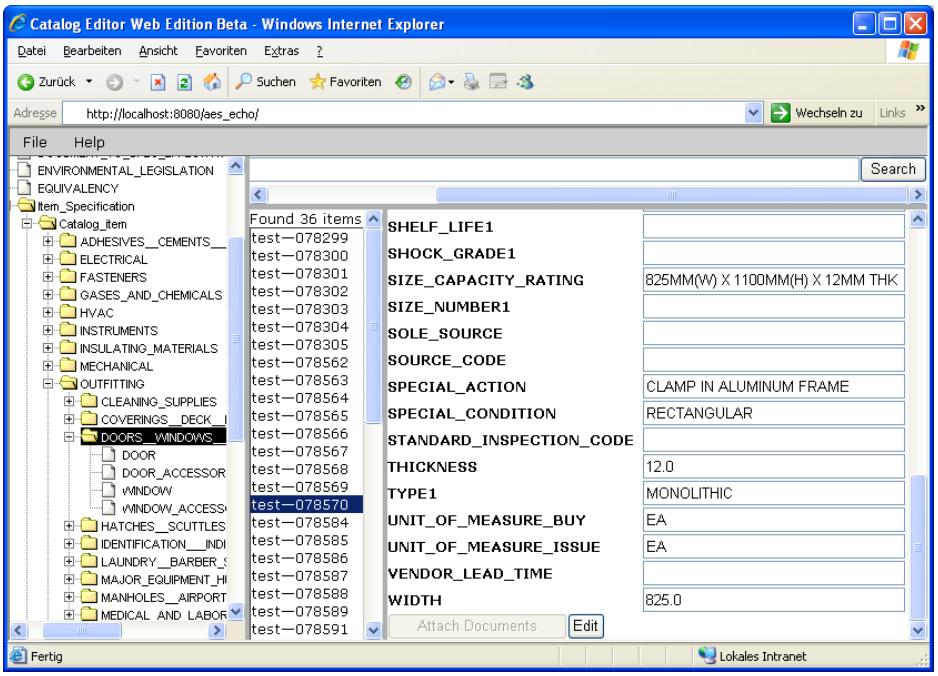

Abbildung 5.2: Web Anwendung im Browser Internet Explorer v. 6.0

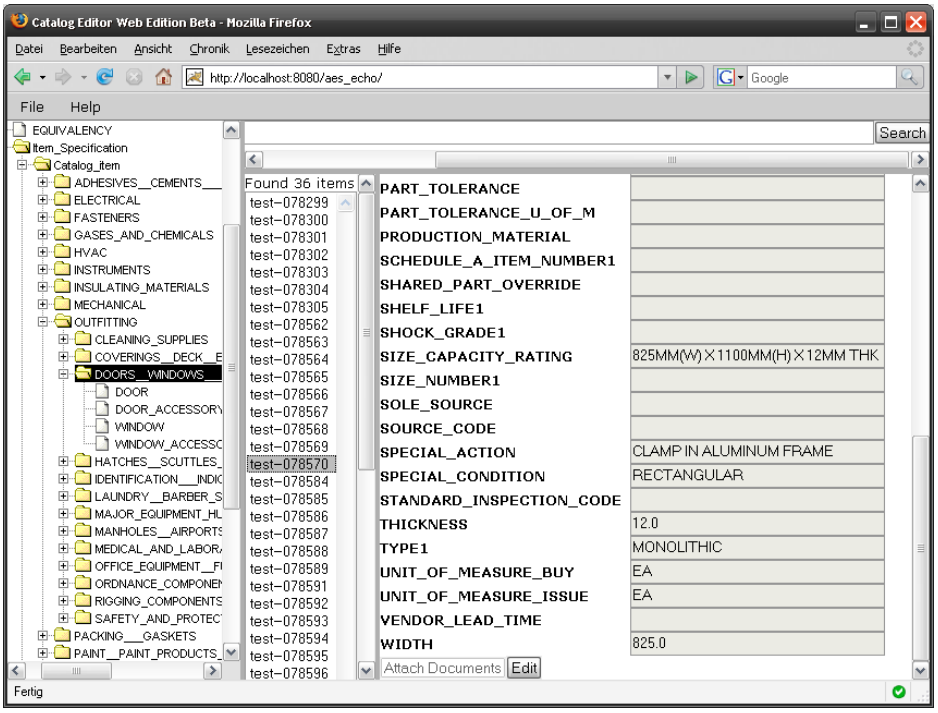

Abbildung 5.3: Web Anwendung im Browser Firefox v. 2.0.0.11

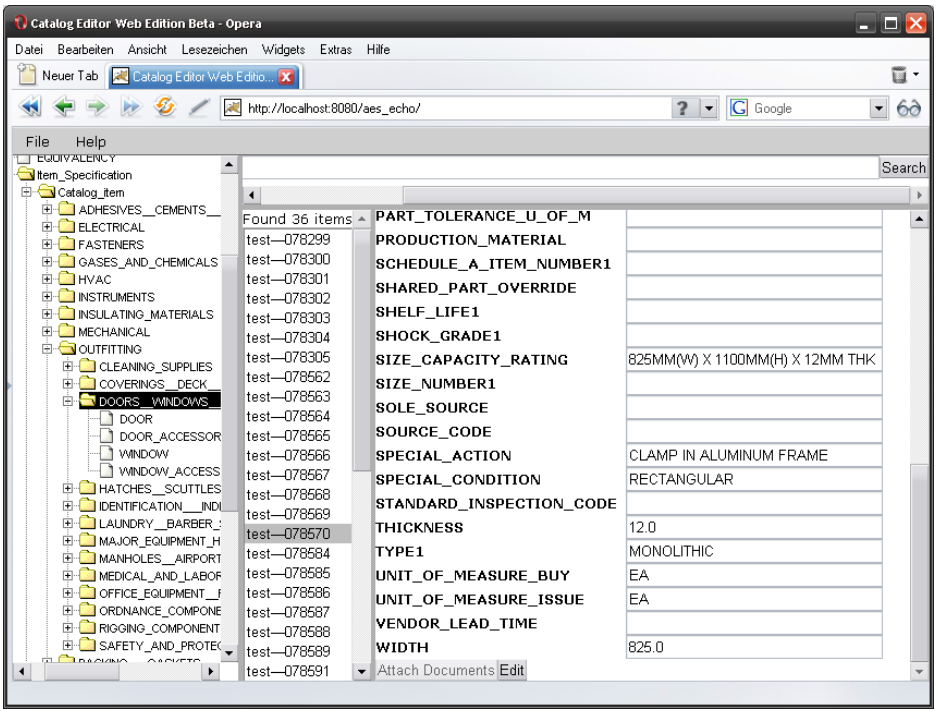

Abbildung 5.4: Web Anwendung im Browser Opera v. 9.25

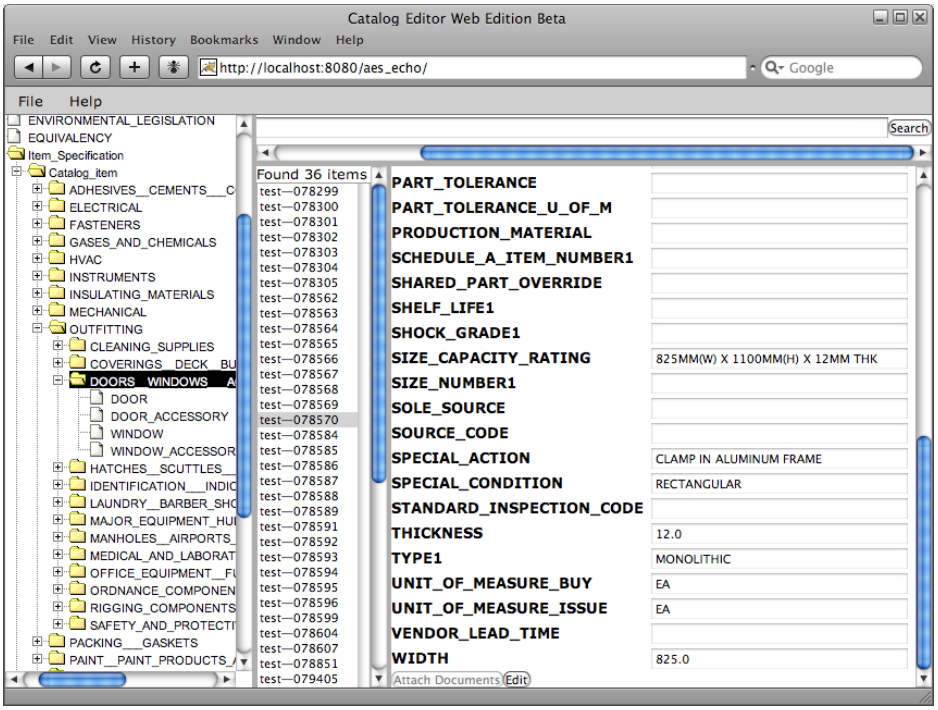

Abbildung 5.5: Web Anwendung im Browser Safari v. 3.0.4

Bei der Nutzung von Web Anwendungen gilt ein bestehender Internetzugang als Voraussetzung. Da die Kapazität der Bandbreite nicht immer und überall gewährleistet werden kann, kann es zu verlängerten Antwortzeiten führen. Diese Zeiten müssen in Grenzen gehalten werden, damit ein Nutzer die Web Anwendung an der Stelle von einer Desktop Anwendung akzeptiert.

Basierend auf den Gesprächen mit den unterschiedlichen Nutzer der verschiedenen Desktop Anwendungen konnte festgestellt werden, dass es für einen Nutzer gewöhnungsbedürftig ist, eine Anwendung innerhalb eines Web Browsers zu bedienen. Der Back-Button eines Web Browsers kann viele Benutzer verwirren. Durch den Einsatz von Native Window[7] (siehe Abbildung [5.6]) kann diese Verwirrung genommen werden. Das Native Window kann den Browser-Geschmack bei den Benutzern wegnehmen. Wenn ein Benutzer die Web Anwendung verwendet, dann besteht für ihn kein inhaltlicher Unterschied zur Verwendung der ähnlichen Desktop Anwendung.

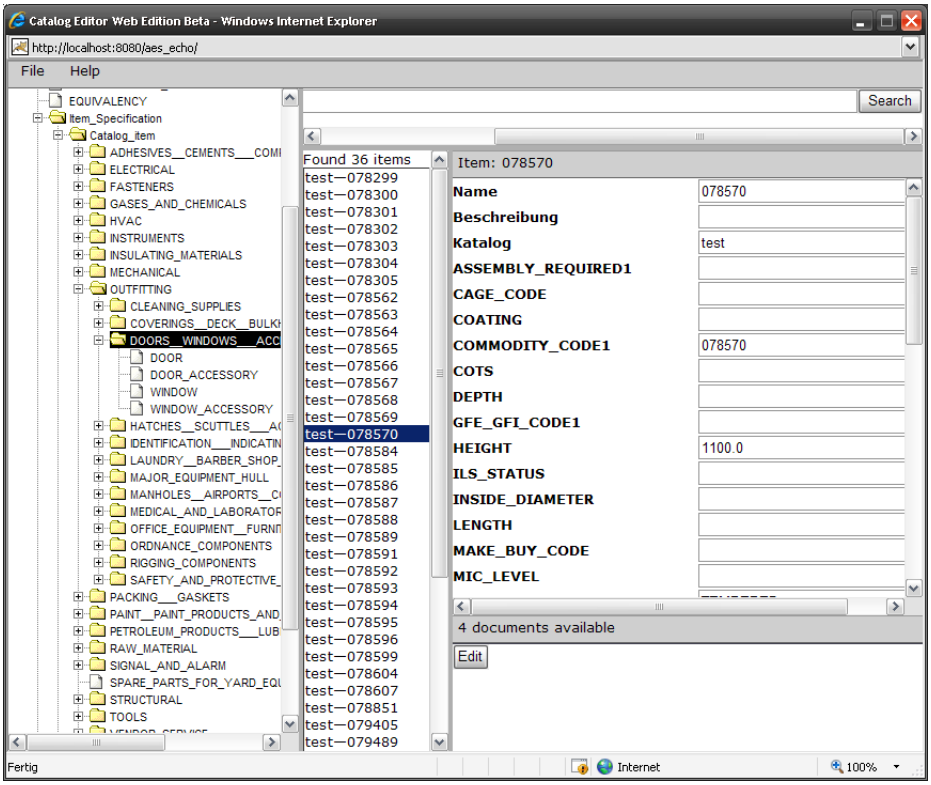

Abbildung 5.6: Web Anwendung im Native Window

Auf der Basis der Untersuchungen mit der Verwendung von einem Native Window konnte

[7] Ein Native Window beinhaltet keinen Back-Button

festgestellt werden, dass die Benutzer ganz ohne oder mit den geringen Schulungsmaßnahmen die neue Web Anwendung bedienen und ihre Arbeit erledigen können.

5.2 Akzeptanz durch Entwickler

Bei der Migration einer existierenden Desktop Anwendung auf Code on Demand Technologie ist die Akzeptanz der Entwickler sehr wichtig, die diese Migration durchführen sollen. Aus den Gesprächen mit mehreren Entwicklern sammelte der Autor dieser Arbeit einige Punkte, die für ihn und andere Entwickler bei der Migration wichtig sind. An dieser Stelle werden diese Punkte vorgestellt.

Da die existierende Rich Client Anwendung in der Programmiersprache Java entwickelt wurde, wurde im Kapitel [3.2.6] definiert, dass bei der Analyse ein Framework für die Entwicklung einer Rich Internet Application ausgewählt und bei der Implementierung angewendet werden soll, das ein Java-basiertes API zur Verfügung stellt.

Viele Entwickler fragen sich, warum die Programmiersprache Java anstatt vom HTML mit JavaScript für die Entwicklung einer Rich Internet Application verwendet werden muss. Die Programmiersprache Java ist eine Objekt-orientierte Programmiersprache, die eine statische Typüberprüfung zur Verfügung stellt. Bei der Verwendung von der Programmiersprache Java werden allgemeine Fehler, die man bei Entwicklung mit HTML und JavaScript hat, sei es Tippfehler oder falsche Typzuordnung, zur Übersetzungszeit und nicht wie bei JavaScript und HTML zur Laufzeit gefangen. Dadurch werden die treueren Laufzeitfehler reduziert und die Produktivität der Entwicklung erhöht. Nach der Meinung vieler Entwickler ist der Java Code viel strukturierter und leicht verständlicher als der HTML und JavaScript Code. Es ist flexibler eine in der Programmiersprache Java geschriebene Anwendung zu testen und zu debuggen.

Bei der Verwendung eines Java-basierten AJAX Frameworks ist es wichtig, dass ein Entwickler die State-of-the-Art-Entwurfsprinzipien und die bekannten Entwurfsmuster einsetzen kann. Das Framework muss gut dokumentiert sein und weiterentwickelt werden. Bei dem Einsatz von solchen Frameworks ist eine aktive Comnunity sehr wichtig. Code Generatoren, Preparsing in IDE und GUI Builder sind gewünscht.

Bei dem Einsatz von einer Web Anwendung muss man sehen, wie einfache Fehler der Benutzer (z.B. JavaScript ist im Web Browser ausgeschaltet, etc.) vermieden werden können. Obwohl eine Web Anwendung auf der Client Seite nicht installiert werden muss, gelten einige Einstellungen des Web Browsers als Voraussetzung. Eine der Einstellungen ist z.B. die Erlaubnis der JavaScript-Ausführung. Während der Schulungsmaßnahmen müssen solche Einstellungen auch den Benutzern verständlich erläutert werden.

Da eine Web Anwendung keine Installation voraussetzt, ist deren Deployment einfach. Bei allen Änderungen der Anwendung kann ein Entwickler eine neue Version ohne zusätzlichen Aufwand auf den Web Server ausliefern. In diesem Fall steht die Anwendung sofort den Nutzern zum Testen zur Verfügung. Mit anderen Worten kann gesagt werden, dass sich die auf Code on Demand Technologie basierten Web Anwendungen immer in der Testphase befinden. Diese Phase ist als Perptual Beta bekannt und wird an dieser Stelle vorgestellt.

5.2.1 Perpetual Beta

In [Musser u. a. (2006)] beschreiben John Musser und Tim O'Reilly das Perpetual Beta Modell als Development 2.0. Es ist keine Version, keine Installation und kein Upgrade der Anwendung notwendig. Der traditionelle Design-Entwicklung-Testen-Auslieferung-Installation Zyklus der Standard-Software wird beenden [vgl. Musser u. a. (2006)]. Eine Software ist ein Dienst geworden, der immer da ist, und ständig verfeinert und verbessert wird.

In dem Perpetual Beta Modell sind die Software so früh und so oft wie möglich auszuliefern. Die neue Funktionalität oder die Fehlerbeseitigung werden in die Web Anwendungen eingebaut und stehen dadurch den Nutzern sofort zur Verfügung. Viele Projekte wollen den Zusatz „Beta“ am liebsten in alle Ewigkeit behalten, denn moderne Web Anwendungen sind einem ständigen Wandel ausgesetzt. Die Benutzer sind gleichzeitig Beta Tester, deren Meinung als wichtige Kritiken zur Verbesserung und Erweiterung der Anwendungen ist. In diesem Fall spielt ein Benutzer eine Co-Entwickler Rolle. Sein Feedback steht immer im Vordergrund. Dadurch bekommen die Entwickler schnell Informationen, welche Funktionalität einige Fehler beinhaltet, welche Funktionen fehlen oder im Gegensatz überflüssig sind. Mit anderen Worten, die Benutzer entwickeln die Software mit.

In dem Perpetual Beta Modell ist es sinnvoll, einige Werkzeuge einzusetzen, die erfassen können, wie die Benutzer/Kunden die Anwendungen nutzen. Die erfassten Daten darüber, wie die Benutzer die Anwendungen bedienen, sind für einen Entwickler wesentlich informativer, als die Aussagen der Benutzer. Dadurch kann z.B. die Priorität der zu entwickelnden Funktionen festgelegt werden.

5.3 Akzeptanz durch Unternehmen

Da ein Unternehmen in der Software Entwicklung eine große Rolle spielt, ist es wichtig, an dieser Stelle vorzustellen, welche Vor- und/oder Nachteile die Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie für ein Unternehmen bringt. Dadurch kann die Akzeptanz eines Unternehmens gewonnen werden. Die Unternehmen lassen sich in zwei Gruppen unterscheiden. Zu der ersten Gruppe gehören die Unternehmen, die

die Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie durchführen. Die Unternehmen, die dagegen die migrierte Anwendung verwenden sollen, gehören zu der zweiten Gruppe.

5.3.1 Unternehmen, die eine Migration durchführen

Als eine der wichtigsten Voraussetzungen für die Migration einer Rich Client Anwendung auf Code on Demand Technologie gilt die saubere Architektur der existierenden Anwendung. Eine saubere Architektur bedeutet in diesem Fall, dass es eine klare Trennung zwischen der Business Logik Schicht und den anderen Schichten existiert. Nicht alle Legacy-Anwendungen wurden mit einer guten Architektur entworfen. Manche Anwendungen sind riesig und kaum noch beherrschbar. Nicht nur die Migration, sondern auch die Änderungen an Geschäftsprozessen sind sehr schwer umzusetzen. In solchen Fällen kann die Service-orientierte Architektur (SOA) ins Spiel kommen. Die existierende Anwendung kann durch die SOA-Transformation in eine SOA-fähige Anwendung transformiert werden. Einige mögliche Lösungsmuster für die SOA-Transformation von Legacy-Anwendungen und ihre Anwendung auf Grundlage einer Typisierung der Alltsysteme stellt Oliver F. Nandico in seinem Artikel [vgl .Nandico (2007)] vor. Nach so einer Transformation beinhaltet die Architektur der Anwendung u.a. eine Service-Schicht (die Schicht „Services“ [3] in der Abbildung [5.7]), die die unterschiedliche Funktionalität der Anwendung als Services zur Verfügung stellt. Auf diese Services kann dann bei der Migration auf Code on Demand Technologie zugegriffen werden.

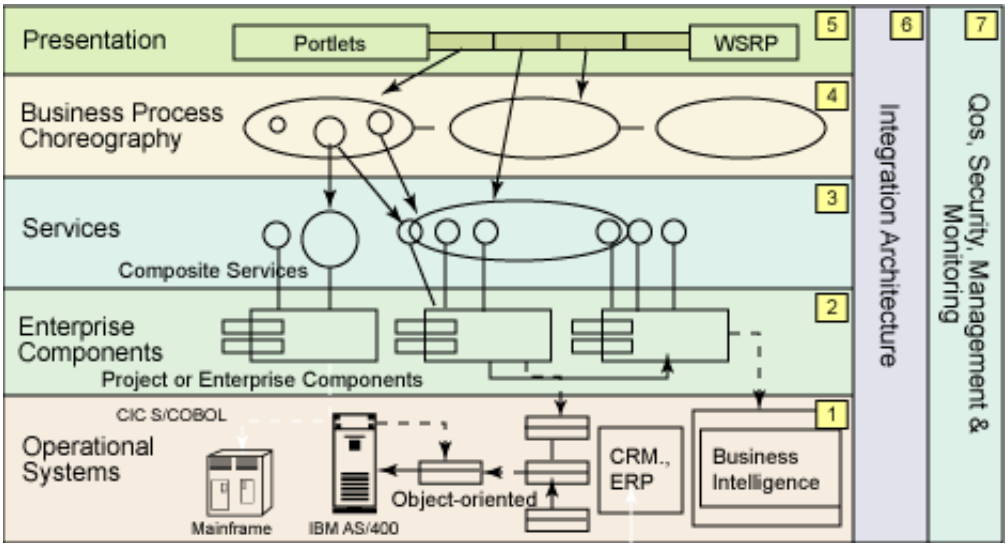

Abbildung 5.7: SOA-Schichten. Quelle: [Arsanjani (2004)]

Um eine existierende Rich Client Anwendung auf Code on Demand Technologie migrieren zu können, braucht ein Unternehmen das notwendige Know-how über Code on Demand Technologie. Sind bei dem Unternehmen Entwickler angestellt, die diese Kenntnisse besitzen, kann das Unternehmen solche Mitarbeiter in die Migrationsprojekte einsetzen. Sollte das Wissen über die notwendigen Technologien im Unternehmen nicht vorhanden sein, sind die Schulungsmaßnahmen für die Entwickler erforderlich. Die Schulungsmaßnahmen bedeuten für ein Unternehmen die zusätzlichen Investitionen. Ein Unternehmen kann seine Entwickler aber selbst weiterbilden lassen. Trotzdem bedeutet es für das Unternehmen zusätzliche Kosten, weil die Entwickler neue Technologien während ihrer Arbeitszeit erlernen werden und während dieser Zeit nicht in andere Projekte eingesetzt werden können.

Da die existierende Rich Client Anwendung in der Programmiersprache Java entwickelt wurde, ist es in dieser Arbeit vorausgesetzt, dass die Migration mittels eines Java-basiertes Frameworks durchgeführt werden sollte. Das heißt auch, dass das Unternehmen das Know-how über die Java-Entwicklung besitzt und es nur die Einarbeitung der Entwickler in das ein oder anderes Framework notwendig ist. Die Einarbeitung kostet auch Zeit und somit für ein Unternehmen auch das Geld. Dieser Zeitaufwand kann man in Grenzen halten und die Zeit muss nur einmal investiert werden. Sollte ein Unternehmen weitere Projekte planen, in denen eine ähnliche Migration durchgeführt werden sollte, dann besitzen die Entwickler das notwendige Know-how und können ohne zusätzliche Investitionen in ein weiteres Projekt eingesetzt werden.

Der Einsatz von einer Web Anwendung bring einen großen Vorteil für die Vertrieb- und Marketingabteilungen eines Unternehmens. Eine Web Anwendung kann von den Abteilungen ohne einen zusätzlichen Aufwand (z.B. den Installationsaufwand) für die Demozwecke verwendet werden. Die potenzielle Kunden bekommen in diesem Fall Zeitlich befristete Zugriffsrechte. Dadurch werden Marketingkosten reduziert.

Da es in dieser Arbeit um die Migration einer Business Anwendung geht, ist das Lizenz Modell der neuen Anwendung ein Punkt, der beachtet werden muss. Bei dem Einsatz von Web Anwendungen ist die serverseitige Lizensierung möglich und es können parallele Sessions lizensiert werden.

5.3.2 Unternehmen, die eine migrierte Anwendung nutzen

Die Unternehmen (Kunden), die eine Rich Client Anwendung verwenden oder verwendeten und jetzt die neue auf Code on Demand Technologie migrierte Anwendung einsetzen und nutzen werden, gehören zu der zweiten Gruppe. Für die ßUnternehmen ergeben sich bei dem Einsatz der migrierten Anwendung auf jeden Fall Kostenvorteile.

Die Code on Demand-basierte Web Anwendung wird zentral administriert. Durch die zentrale Administrierbarkeit werden die Kosten für die Administration und die Wartung der Anwendung für das Unternehmen reduziert.

Da das Unternehmen eine Web Anwendung verwendet werden kann, ohne sie installiert zu haben, werden die Support-Zeiten verkürzt. Dadurch werden die Kosten des Unternehmens reduziert, die für den Support fällig sein können.

Bei dem Einsatz von der migrierten Anwendung muss einen folgenden Punkt berücksichtigt werden. Die neue Web Anwendung muss genau wie die existierende und von dem Unternehmen verwendeten Rich Client Anwendung alle notwendige Verbindungen zu der Peripherie des Unternehmens herstellen können. Die Verbindungen zu den Kunden Datenbanken und den weiteren Kunden-Systemen müssen sicher gestellt werden.

Abbildung 5.8: Jack PC. Quelle: [Chip PC]

Heutzutage geht die Tendenz in den Unternehmen in die Richtung von Thin Client Rechnern. Ein Beispiel der Thin Client Rechner ist Jack PC[8] von der Firma Chip PC Technologies. Jack PC ist ein kostengünstiger Rechner und lässt sich in die Wand, in den Boden oder ins Möbel

[8] http://www.chippc.com/thin-clients/jack-pc ; Zugriffsdatum: 30.01.2008

einbauen (siehe Abbildung [5.8]). Jack PC ist ein Thin Client und greift auf die Ressourcen von Rechenzentrum. Ein Benutzer kann im Jack PC keine zusätzlichen Konfigurationen vornehmen. Er hat aber einen Web Browser, mit dem die Web Anwendung genutzt werden kann, und kann seine Arbeit erledigen.

5.4 Fazit

In diesem Kapitel wurde das System evaluiert und die gefundenen Ergebnisse wurden bewertet. Es wurde vorgestellt, ob und wie die neue auf Code on Demand Technologie basierte Web Anwendung von drei unterschiedlichen Menschengruppen akzeptiert wird. Diese drei Gruppen sind die Endbenutzer, die Entwickler und die Unternehmen.

Die Evaluierung basiert auf den Gesprächen, die der Autor dieser Arbeit mit den unterschiedlichen Endnutzern, den erfahrenen und weniger erfahrenen Entwicklern und den Vertrieb-, Marketing- und Supportmitarbeitern der Unternehmen führte. In diesem Kapitel wurde gezeigt, was sich für einen Endnutzer durch den Einsatz einer Web Anwendung statt der existierenden Rich Client Anwendung ändert, worauf ein Entwickler bei dem Entwurf und der Entwicklung einer auf Code on Demand Technologie basierten Web Anwendung achten soll und welche Vor- bzw. Nachteile die Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie für ein Unternehmen bringt.

6 Schlussfolgerung

Zum Abschluss dieser Arbeit werden in diesem Kapitel eine Zusammenfassung und ein Ausblick der gesamten Arbeit gegeben.

6.1 Zusammenfassung

Im Rahmen dieser Arbeit beschäftigte sich der Autor mit der Migration einer Rich Client Applikation auf Code on Demand Technologie. Bei der Migration einer existierenden Desktop Anwendung wurde eine Web Anwendung mit der Verwendung von dem Echo Framework entworfen. Dieser Entwurf breitete sich nur auf die Entwicklung der obersten Schicht, und zwar der Presentation Schicht, der Architektur des gesamten Systems (s. Kapitel [4.4]) aus. Wie in den Anforderungen definiert wurde, wurde die Business Logik Schicht nicht geändert. Es wurden nur die Maßnahmen vorgenommen, die für die Trennung zwischen der serverseitigen und cleintseitigen Funktionalität notwendig waren.

Eine Migration auf Code on Demand Technologie bedeutet in dieser Arbeit nicht nur die Entwicklung einer Web Anwendung, sondern auch die Untersuchung der folgenden Fragen:

- Mit welcher Technologie und inwiefern kann die Migration einer bestehenden Rich Client Geschäftsanwendungen durchgeführt werden? Welche der existierenden Frameworks sind für die Migration geeignet?
- Was hat ein Entwickler mit dem aktuellen Kenntnisstand über Softwareentwicklung und Design zu erledigen, um Geschäftsanwendungen auf Code on Demand Technologie mit geringeren Aufwand migrieren zu können?
- Was ändert sich für einen Nutzer durch den Einsatz einer Web Anwendung statt der existierenden Rich Client Anwendung?
- Welche Vor- bzw. Nachteile bringt eine Migration einer existierenden Rich Client Anwendung auf Code on Demand Technologie für ein Unternehmen?
- Wie kann das „Change and Configuration Management“ Problem reduziert werden?

Zum Verständnis der Arbeit wurden im Kapitel [2] notwendige Grundlagen vorgestellt. Es wurden die Konzepte von Rich Client und Rich Internet Applikationen beschrieben. Der Leser wurde mit dem Begriff Web 2.0 und dem Asynchronous JavaScript and XML (AJAX) Konzept als Basis für die Code on Demand Technologie vertraut gemacht. Zum Schluss des Grundlagenkapitels wurden Java-basierte AJAX Frameworks (Google Web Toolkit und Echo Framework) präsentiert und anhand der Vor- und Nachteile verglichen.

Das Analysekapitel [3] beschreibt die genauen Anforderungen an das System. Da die Funktionalität der Geschäftsanwendung bei der Migration und der Entwicklung einer neuen Web Anwendung bestehen bleiben soll, basieren die definierten Anwendungsfälle auf den Vorgaben des Unternehmens (Atlantec Enterprise Solutions GmbH) und den Anforderungen an die existierende Rich Client Anwendung.

In dem Design und Konzeptionelle Entscheidungen Kapitel [4] wurden unterschiedliche Architekturstyle der Desktop Anwendungen und Rich Internet Applikationen diskutiert. Als eine der möglichen Lösungen des „Change and Configuration Management“ Problem wurde das Web Deployment aufgezeigt. Es wurde die Architektur des gesamten Systems präsentiert. Darauf aufbauend folgt die Erörterung, welche Funktionalität der Anwendung auf der Server- und welche auf der Client-Seite ausgeführt wird. Unter der Entwicklung einer neuen Web Anwendung wird bei der Migration die Entwicklung der obersten Schicht in dem Systemarchitektur und zwar der Graphische User Interface Schicht verstanden. Die bestehende Businesslogik muss bei der Migration verwendet werden und darf nicht geändert werden.

Die am Anfang dieses Abschnittes aufgelisteten Fragen wurden im Evaluierungskapitel [5] mit der Untersuchungen nach der Akzeptanz jeweils durch Nutzer, Entwickler und Unternehmen beantwortet. Bei der Beantwortung der Fragen basiert der Autor nicht nur auf seinen Erfahrungen und Erkenntnissen während dieser Arbeit, sondern auch auf den Gesprächen, die mit den unterschiedlichen Endnutzern, den erfahrenen und weniger erfahrenen Entwicklern und den Vertrieb-, Marketing- und Supportmitarbeitern der Unternehmen geführt wurden.

6.2 Ausblick

Durch die Virtualisierungslösung auf der Server Seite wird es den Kunden ermöglicht, deren Kosten zu reduzieren, die Serverauslastung zu steigern und neue Server rasch bereitzustellen. Die Virtualisierung optimiert die Infrastruktur und erhöht die Serververfügbarkeit. Dadurch kann der Administrationsaufwand gesenkt werden.

Da die existierende Rich Client Anwendung auf jedem Client Rechner installiert wurde, bedeutet es, dass sie von mehreren Benutzern gleichzeitig verwendet werden kann. Die durch die Migration entstandene Web Anwendung muss hohe Verfügbarkeit gewährleisten. Es ist wichtig, zu evaluieren, wie viele Benutzer gleichzeitig die Web Anwendung verwenden und

damit ihre Arbeit erledigen können. Es muss sichergestellt werden, dass die Verfügbarkeit der Anwendung durch die Endbenutzer akzeptiert wird. Weitere zu evaluierende Aspekte sind die Sicherheit, die Benutzerauthentifizierung, die Integrationsfähigkeit mit den Kunden-Datenbanken und den weiteren Kunden-Systemen.

Eine Anwendung innerhalb eines Web Browsers zu bedienen, ist für viele Nutzer der verschiedenen Desktop Anwendungen gewöhnungsbedürftig. In wie weit möchte man den Back-Button im Web Browser beibehalten, ist eine zu untersuchende Frage. Man kann das Native Window ohne den Back-Button einsetzen, das den Browser-Geschmack bei den Benutzern wegnimmt. Es muss untersucht werden, ob dieser Einsatz von Benutzern akzeptiert wird.

Heutzutage existieren schon einige Frameworks (z.B. AjaxSwing[1]), die eine automatische Konvertierung der Java Swing und AWT Anwendungen in voll funktionsfähige HTML und AJAX Anwendungen. Es wäre sinnvoll, zu evaluieren, ob solche Frameworks für die Konvertierung der Business Anwendungen geeignet sind. Die saubere Architektur der existierenden Anwendung gilt auch bei einer automatischen Konvertierung als eine der wichtigsten Voraussetzungen. Ob sich solche Frameworks in der Praxis durchsetzen und einen Erfolg haben, ist schwer vorauszusagen.

Nichtsdestotrotz wird die Migration der weiteren Rich Client Anwendungen auf Code on Demand Technologie von dem Unternehmen Atlantec Enterprise Solutions GmbH basierend auf den Erkenntnissen und den Untersuchungen dieser Arbeit in seine Überlegungen einbezogen.

[1] http://www.creamtec.com/products/ajaxswing/ ; Zugriffsdatum: 30.01.2008

Literaturverzeichnis

[RFC2764] RFC 2764 (rfc2764) - A Framework for IP Based Virtual Private Networks. . – URL http://www.faqs.org/rfcs/rfc2764.html. – Zugriffsdatum: 30.01.2008

[IEEEStd 2000] IEEE Recommended practice for architectural description of software-intensive systems. In: *IEEE Std 1471-2000* (2000), S. i–23

[Alpar u. a. 2007] ALPAR, Paul ; BLASCHKE, Steffen ; KESSLER, Steffen: *Web 2.0: Neue erfolgreiche Kommunikationsstrategien für kleine und mittlere Unternehmen*. Hessen-Media, Hessische Ministerium für Wirtschaft, Verkehr und Landesentwicklung. 2007

[Arsanjani 2004] ARSANJANI, Ali: *Service-oriented modeling and architecture*. SOA and Web Services Center of Excellence. IBM Global Services. November 2004. – URL http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/. – Zugriffsdatum: 30.01.2008

[Atlantec Enterprise Solutions 2001a] ATLANTEC ENTERPRISE SOLUTIONS: *Information Server. Specification.* November 2001

[Atlantec Enterprise Solutions 2001b] ATLANTEC ENTERPRISE SOLUTIONS: *Process Engine. Specification*. November 2001

[Atlantec Enterprise Solutions 2003a] ATLANTEC ENTERPRISE SOLUTIONS: *Enterprise Reference Model*. 2003

[Atlantec Enterprise Solutions 2003b] ATLANTEC ENTERPRISE SOLUTIONS: *Part Catalog. Specification*. September 2003

[Atlantec Enterprise Solutions 2006] ATLANTEC ENTERPRISE SOLUTIONS: *Topgallant Infrastruktur*. 2006

[Banditwattanawong Oct. 18 2006–Sept. 20 2006] BANDITWATTANAWONG, Hironori; Hidaka Soichiro; Maruyama K.: Partial and On-Demand Incremental Deployment of Java Application Program over the Internet. In: *Communications and Information Technologies, 2006. ISCIT '06. International Symposium on* (Oct. 18 2006-Sept. 20 2006), S. 428–433

[Bengel 2000] BENGEL, Günther: *Verteilte Systeme: Client-Server-Computing für Studenten und Praktiker.* Vieweg, 2000. – ISBN 3-528-05738-6

[Booch 2001] BOOCH, Grady: The architecture of Web applications. (2001), Juni. – URL http://www-128.ibm.com/developerworks/java/library/it-booch_web/. – Zugriffsdatum: 30.01.2008

[Bozzon u. a. 2006a] BOZZON, Alessandro ; COMAI, Sara ; FRATERNALI, Piero ; CARUGHI, Giovanni T.: Capturing RIA concepts in a web modeling language. In: *WWW '06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA : ACM, 2006, S. 907–908. – ISBN 1-59593-323-9

[Bozzon u. a. 2006b] BOZZON, Alessandro ; COMAI, Sara ; FRATERNALI, Piero ; CARUGHI, Giovanni T.: Conceptual modeling and code generation for rich internet applications. In: *ICWE '06: Proceedings of the 6th international conference on Web engineering*. New York, NY, USA : ACM, 2006, S. 353–360. – ISBN 1-59593-352-2

[Carfagno 2007] CARFAGNO, Virginio: *Evaluation des Google Web Toolkits durch Entwicklung einer ajaxbasierten Mind-Mapping-Anwendung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, März 2007

[Carzaniga 17–23 May 1997] CARZANIGA, G.P.; Vigna G.: Designing Distributed Applications with Mobile Code Paradigms. In: *Software Engineering, 1997., Proceedings of the 1997 (19th) International Conference on* (17-23 May 1997), S. 22–32. – ISSN 0270-5257

[Chip PC] CHIP PC, Tecnology: *Jack PC*. – URL http://www.chippc.com/thin-clients/jack-pc/

[Coulouris u. a. 2002] COULOURIS, George ; DOLLIMORE, Jean ; KINDBERG, Tem: *Verteilte Systeme: Konzepte und Design*. 3, überarbeitete Auflage. Addison-Wesley, 2002. – ISBN 3-8273-7022-1

[Crane u. a. 2005] CRANE, Dave ; PASCARELLO, Eric ; JAMES, Darren: *Ajax in Action*. Manning Publications, October 2005. – ISBN 1932394613

[Duhl 2003] DUHL, Joshua: *Rich Internet Applications. White Paper*. November 2003. – URL http://www.adobe.com/resources/business/rich_internet_apps/whitepapers.html. – Zugriffsdatum: 30.01.2008

[Eckert 2004] ECKERT, Claudia: *IT-Sicherheit*. München [u.a.] : Oldenbourg, 2004. – ISBN 3-486-20000-3

[Erenkrantz u. a. 2007] ERENKRANTZ, Justin R. ; GORLICK, Michael ; SURYANARAYANA, Girish ; TAYLOR, Richard N.: From representations to computations: the evolution of web architectures. In: *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations*

of software engineering. New York, NY, USA : ACM, 2007, S. 255–264. – ISBN 978-1-59593-811-4

[Eytorff u. a. 2007] EYTORFF, Thomas ; JANKOVICS, Elin ; SCHWALB, Markus: RIA - Hype oder geniale Lösung? In: *JavaSPEKTRUM* (2007), Februar/März, Nr. 1

[Farrell 25–28 June 2007] FARRELL, G.S.: Rich Internet Applications The Next Stage of Application Development. In: *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on* (25-28 June 2007), S. 413–418. – ISSN 1330-1012

[Feinstein 2000] FEINSTEIN, Wei P.: A study of technologies for Client/Server applications. In: *ACM-SE 38: Proceedings of the 38th annual on Southeast regional conference*. New York, NY, USA : ACM, 2000, S. 184–193. – ISBN 1-58113-250-6

[Fielding und Taylor 2002] FIELDING, Roy T. ; TAYLOR, Richard N.: Principled design of the modern Web architecture. In: *ACM Trans. Inter. Tech.* 2 (2002), Nr. 2, S. 115–150. – ISSN 1533-5399

[Fielding 2000] FIELDING, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*. Irvine, CA, USA, University of California, Dissertation, 2000

[Garcia 2007] GARCIA, Jürgen S.: *Enterprise 2.0: Web 2.0 im Unternehmen*. VDM Verlag Dr. Müller, 2007. – ISBN 978-3-8364-0399-3

[Garrett 2005] GARRETT, Jesse J.: Ajax: A New Approach to Web Applications. (2005), Februar. – URL http://www.adaptivepath.com/ideas/essays/archives/000385.php. – Zugriffsdatum: 30.01.2008

[Geihs 2001] GEIHS, Kurt: Middleware challenges ahead. In: *IEEE Computer* 34 (2001), Juni, Nr. 6, S. 24–31

[Gibson 2007] GIBSON, Becky: Enabling an accessible web 2.0. In: *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*. New York, NY, USA : ACM, 2007, S. 1–6. – ISBN 1-59593-590-X

[Google 2006] GOOGLE, Inc: *Google Web Toolkit*. Mai 2006. – URL http://code.google.com/webtoolkit/. – Zugriffsdatum: 30.01.2008

[Haft und Olleck 2007] HAFT, Martin ; OLLECK, Bernd: Komponentenbasierte Client-Architektur. In: *Informatik Spektrum* (2007), Juni, Nr. Band 30 Heft 3, S. 143–158

[Hasselbring 2006] HASSELBRING, Wilhelm: Software-Architektur. In: *Informatik Spektrum* 29 (2006), Februar, Nr. 1, S. 48–52

[Jern 1998] JERN, Mikael: "Thin"vs. "fat"visualization clients. In: *AVI '98: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA : ACM, 1998, S. 270–273

[Joeris 1997] JOERIS, Gregor: Change management needs integrated process and configuration management. In: *ESEC '97/FSE-5: Proceedings of the 6th European conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering*. New York, NY, USA : Springer-Verlag New York, Inc., 1997, S. 125–141. – ISBN 3-540-63531-9

[Khare und Taylor 2004] KHARE, Rohit ; TAYLOR, Richard N.: Extending the Representational State Transfer (REST) Architectural Style for Decentralized Systems. In: *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2004, S. 428–437. – ISBN 0-7695-2163-0

[Lewandowski 1998] LEWANDOWSKI, Scott M.: Frameworks for component-based client/server computing. In: *ACM Comput. Surv.* 30 (1998), Nr. 1, S. 3–27. – ISSN 0360-0300

[Ludewig und Lichter 2006] LUDEWIG, Jocjen ; LICHTER, Horst: *Software Engineering: Grundlage, Menschen, Prozesse, Techniken*. 1. Auflage. dpunkt.verlag, September 2006. – ISBN 3-89864-268-2

[Mesbah Jan. 2007] MESBAH, A.: An Architectural Style for Ajax. In: *Software Architecture, 2007. WICSA '07. The Working IEEE/IFIP Conference on* (Jan. 2007), S. 9–9

[Middendorf u. a. 2002] MIDDENDORF, Stefan ; SINGER, Reiner ; HEID, Jörn: *Java. Programmierhandbuch und Referenz für die Java-2-Plattform, Standard Edition*. 3., überarbeitete und erweiterte Auflage. dpunkt.verlag, Oktober 2002. – URL http://www.dpunkt.de/java. – Zugriffsdatum: 30.01.2008. – ISBN 3-89864-157-9

[Musser u. a. 2006] MUSSER, John ; O'REILLY, Tim ; O'REILLY RADAR TEAM the: *Web 2.0 Principles and Best Practices*. O'Reilly Media, Inc., November 2006. – 101 S. – ISBN 0-596-52769-1

[Nandico 2007] NANDICO, Oliver F.: SOA-Transformation von Legacy-Anwendungen. In: *OBJEKTspektrum* (2007), September/Oktober, Nr. 5, S. 18–24

[NextApp 2007] NEXTAPP: *Echo2 Technical Overview*. 2007. – URL http://echo.nextapp.com/site/echo2/doc/tov. – Zugriffsdatum: 30.01.2008

[O'Reilly 2005] O'REILLY, Tim: What is Web 2.0? Design patterns and business models for the next generation of software. (2005), September. – URL http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. – Zugriffsdatum: 30.01.2008

[Preciado 5–6 Oct. 2007] PRECIADO, M.; Comai S.; Sanchez-Figueroa F.: Designing Rich Internet Applications with Web Engineering Methodologies. In: *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on* (5-6 Oct. 2007), S. 23–30

[Preciado 26 Sept. 2005] PRECIADO, M.; Sanchez F.; Comai S.: Necessity of methodologies to model rich Internet applications. In: *Web Site Evolution, 2005. (WSE 2005). Seventh IEEE International Symposium on* (26 Sept. 2005), S. 7–13. – ISSN 1550-4441

[Rescorla 2000] RESCORLA, E.: *HTTP Over TLS*. RFC 2818 (Informational). Mai 2000 (Request for Comments). – URL http://www.ietf.org/rfc/rfc2818.txt. – Zugriffsdatum: 30.01.2008

[Roberts 2007] ROBERTS, Eric: Resurrecting the applet paradigm. In: *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*. New York, NY, USA : ACM, 2007, S. 521–525. – ISBN 1-59593-361-1

[Scheifler und Gettys 1986] SCHEIFLER, Robert W. ; GETTYS, Jim: The X window system. In: *ACM Trans. Graph.* 5 (1986), Nr. 2, S. 79–109. – ISSN 0730-0301

[Schäffer 2003] SCHÄFFER, Bruno: Probleme bei Thin-Clients. In: *JavaSPEKTRUM* (2003), Januar, Nr. 1

[Shan 1989] SHAN, Y.-P.: An event-driven model-view-controller framework for Smalltalk. In: *OOPSLA '89: Conference proceedings on Object-oriented programming systems, languages and applications*. New York, NY, USA : ACM, 1989, S. 347–352. – ISBN 0-89791-333-7

[Shaw und Garlan 1996] SHAW, Mary ; GARLAN, David: *Software architecture: perspectives on an emerging discipline*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1996. – ISBN 0-13-182957-2

[Stamey u. a. 2007] STAMEY, John ; LASSEZ, Jean-Louis ; BOORN, Daniel ; ROSSI, Ryan: Client-side dynamic metadata in web 2.0. In: *SIGDOC '07: Proceedings of the 25th annual ACM international conference on Design of communication*. New York, NY, USA : ACM, 2007, S. 155–161. – ISBN 978-1-59593-588-5

[Steyer 2006] STEYER, Ralph: *AJAX mit Java-Servlets und JSP. So bringen Sie Speed in Ihre Webpräsenz*. 2006

[Sun Microsystems a] SUN MICROSYSTEMS, Inc.: *Java Network Launching Protocol & API Specification (JSR-56)*. – URL http://jcp.org/aboutJava/communityprocess/mrel/jsr056. – Zugriffsdatum: 30.01.2008

[Sun Microsystems b] SUN MICROSYSTEMS, Inc.: *Java Platform, Standard Edition*. – URL http://java.sun.com/javase. – Zugriffsdatum: 30.01.2008

[Sun Microsystems c] SUN MICROSYSTEMS, Inc.: *Java Web Start*. – URL http://java.sun.com/products/javawebstart/. – Zugriffsdatum: 30.01.2008

[Tanenbaum und van Steen 2002] TANENBAUM, Andrew S. ; STEEN, Maarten van: *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002. – ISBN 0-13-088893-1

[Tatemura u. a. 2007] TATEMURA, Junichi ; SAWIRES, Arsany ; PO, Oliver ; CHEN, Songting ; CANDAN, K. S. ; AGRAWAL, Diviyakant ; GOVEAS, Maria: Mashup Feeds:: continuous queries over web services. In: *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. New York, NY, USA : ACM, 2007, S. 1128–1130. – ISBN 978-1-59593-686-8

[Veen 2003] VEEN, Jeffrey: The Business Value of Web Standards. (2003), September. – URL http://www.adaptivepath.com/ideas/essays/archives/000266.php. – Zugriffsdatum: 30.01.2008

[Vlissides 1996] VLISSIDES, John: *Pattern Languages of Program Design*. Addison-Wesley Professional, 1996. – ISBN 0201895277

[Wiegers 2003] WIEGERS, Karl E.: *Software Requirements*. Redmond, WA, USA : Microsoft Press, 2003. – ISBN 0735618798

[Zajicek 2007] ZAJICEK, Mary: Web 2.0: hype or happiness? In: *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*. New York, NY, USA : ACM, 2007, S. 35–39. – ISBN 1-59593-590-X

[Zakas u. a. 2006] ZAKAS, Nicholas C. ; MCPEAK, Jeremy ; FAWCETT, Joe: *Professional Ajax*. Wrox, January 2006. – ISBN 0471777781

Glossar

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CSS Cascading Style Sheets

DOM Document Object Model

ERM Enterprise Reference Model

GUI Graphical User Interface

GWT Google Web Toolkit

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

JNLP Java Network Launching Protocol

JRE Java Runtime Environment

JSNI JavaScript Native Interface

JVM Java Virtual Machine

PHP Hypertext Preprocessor

REST Representational State Transfer

RIA Rich Internet Application

RUE Rich User Experiences

SOA Serviceorientierte Architektur, engl. Service Oriented Architecture

SOAP Simple Object Access Protocol

URL Uniform Resource Locator

VPN Virtual Private Network

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

XSL Extensible Stylesheet Language

XSLT XSL Transformation

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 15. Februar 2008	
Ort, Datum	Unterschrift