SmartClients – Interaktionsparadigmen im Nutzungskontext Webbrowser

Probleme und Lösungen

Jan Groenefeld
User Interface Designer
ERGOSIGN GmbH
Stuhlsatzenhausweg 69
66123 Saarbrücken
groenefeld@ergosign.de
www.ergosign.de

Markus Kühner
User Interface Designer
ERGOSIGN GmbH
Stuhlsatzenhausweg 69
66123 Saarbrücken
kuehner@ergosign.de
www.ergosign.de

Christian Kaspari
User Interface Designer
ERGOSIGN GmbH
Stuhlsatzenhausweg 69
66123 Saarbrücken
kaspari@ergosign.de
www.ergosign.de

Prof. Dr. Dieter Wallach
User Interface Designer
ERGOSIGN GmbH
Stuhlsatzenhausweg 69
66123 Saarbrücken
wallach@ergosign.de
www.ergosign.de

Abstract

Die aktuell in der Webentwicklungsszene vorherrschenden Schlagworte "Rich Internet Applications" (RIA), "SmartClients", "Web2.0" und "AJAX" werden von Webentwicklern in teilweise wahllosem Zusammenhang als verkaufsfördernde "Trendsetter" verwendet. Das Paper grenzt die Begriffe klar gegeneinander ab und bringt sie in einen logischen Gesamtkontext.

Eine kurze technische Einführung in die Verwendung dieser neuen Webtechnologien dient als Grundlage, um aufkommende Usability-Probleme herauszustellen, mit denen sich Webentwickler konfrontiert sehen.

Die Anforderungen an die Benutzerfreundlichkeit einer Webanwendung der Web 2.0 Generation steigen zum einen durch die verwendete Technik als auch durch ein erhöhtes Maß an Interaktionsmöglichkeiten.

Das Paper befasst sich mit Lösungsansätzen sowohl auf konzeptioneller als auch auf technischer Ebene. Neben der rein konzeptionellen Untersuchung von Controls, die für Web 2.0 Anwendungen typisch erscheinen ("Drag&Drop", "Edit In Place") dient die Diskussion eines konkreten Webangebots (www.mtvoverdrive.de) als praxisnahes Beispiel, wie die vorangegangen konzeptionellen Überlegungen zur Steigerung der Benutzerfreundlichkeit im Einzelfall umgesetzt werden könnten.

Keywords

Web 2.0, Rich Internet Applications, AJAX, User Experience, Usability

1.0 Einleitung

Aktuelle Webentwicklungen bieten dank neuer Webtechnologien ungeahnte Funktionsvielfalt und Komplexität. Diese Webanwendungen imitieren nicht selten komplette Desktop-Applikationen und werden in der Regel unter dem Begriff "Web 2.0" zusammengefasst. Als Synonym wird auch der Begriff "Rich Internet Applications" (RIA) verwendet. Die aufkommende nächste Generation von Webanwendungen birgt jedoch neue Herausforderungen für den Benutzer. Diesem sind interaktive Bedienkonzepte sowie ein applikationsartiges Erscheinungsbild einer Webanwendung bisher fremd. Das aktuell vorherrschende Erfahrungsdefizit des Benutzers muss durch verstärkte Bemühungen in der Benutzerfreundlichkeit kompensiert werden.

Darüber hinaus sollte der Benutzer nicht unnötig neuen Bedienkonzepten ausgesetzt werden. Die Entscheidung über die Verwendung eines neuen Interaktionsparadigmas sollte sich weniger nach einem Trend richten, sondern ausschließlich durch die zu erwartende Steigerung der Benutzbarkeit geleitet werden.

Hierbei sind vor allem zwei Problemfelder zu betrachten. Zum einen werfen die neuen Bedienkonzepte aufgrund ihrer steigenden Interaktionskomplexität selbst Probleme auf, zum anderen entstehen Widersprüche zu den Applikations-Controls des Browsers. Insbesondere die History-Buttons eines Browsers werden vom Benutzer als fester Navigationsbestandteil einer Website gesehen. In aktuellen Implementierungen von Web 2.0 Anwendungen weisen diese Controls jedoch kein erwartungskonformes Verhalten auf und führen zu Verwirrung auf Benutzerseite.

2.0 Begriffsdefinitionen

2.1 Web 2.0

Der Begriff "Web 2.0" wurde etabliert, um die aktuellen Webentwicklungen in den gesamthistorischen technologischen Fortschritt von Webanwendungen einzuordnen und markiert den Zeitpunkt der Einführung einer neuen Generation von Webapplikationen. "Web 2.0" steht für mehr Interaktivität, mehr Funktionsvielfalt, vom Benutzer beeinflussbare Webinhalte und die Intensivierung der Benutzergemeinschaft (Web-Community).

2.2 Rich Internet Applications (RIA)

Der Begriff "RIA" wird häufig als Synonym für "Web 2.0" verwendet. Während "Web 2.0" eher als Oberbegriff für die Masse an Webanwendungen der neuen Generation steht, trifft der Begriff "RIA" eine direkte Aussage über die inhaltliche Abgrenzung zu älteren Webanwendungen. Die neue Generation ist "reicher" an Interaktion und Funktion.

2.3 SmartClient

Der "Smart Client" vereint die Vorteile eines "Fat Client" mit denen eines "Thin Client" (Keith, J. 2007).

Durch Nutzung der Client-Ressourcen besitzt ein "Smart-Client" die Interaktivität, Produktivität und Performance eines "Fat Client" und kombiniert diese Eigenschaften mit der zentralen Datenhaltung, automatischen Aktualisierung und minimalem Pflegeaufwand eines "Thin Client".

In den letzten Jahren ist eine Entwicklung zu beobachten, die die Verbreitung von Smart Client-Anwendungen im Webkontext stetig vorantreibt.

Desktop-Applikationen werden durch Web-Applikationen ersetzt, um kosteneffizienter zu arbeiten. Um den vollen Funktionsumfang und hohen Produktivitätslevel einer Desktop-Applikation zu erreichen, bedarf es der Verbesserung der (statischen) Web-Applikationen. Diese werden durch die Verwendung der Smart Client-Technologie erreicht.

Als Anwendungsbeispiele sind hier in erster Linie eine Vielzahl von Email-Clients und Video- und Foto-Communities anzuführen, die sich der Smart Client-Technologie bedienen, um neue Mehrwerte für den Benutzer in der Interaktion und Bedienung zu generieren.

2.4 AJAX

Die Abkürzung "AJAX" steht für *Asynchrone Javascript And XML* und dient als zusätzlicher Kommunikations-Layer zwischen Server und Client. Mit Hilfe dieses Layers wird die asynchrone Kommunikation zwischen Client und Server für den Benutzer unbemerkt im Hintergrund durchgeführt, so dass kein Reload der gesamten Seite nötig ist, um Teile des Contents zu ersetzen. "AJAX" ist vollständig in JavaScript implementiert.

Die Hauptcharakteristik einer RIA besteht in der Regel in der simulierten Zustandslosigkeit des Browsers, die dem Benutzer das Gefühl vermittelt, es handele sich um eine Desktop-Applikation.

Die "AJAX-Technologie" bildet den technischen Grundpfeiler zur zustandslosen Client-Server Kommunikation.

3.0 Überforderung des Benutzers durch Funktionsvielfalt?

Verfolgt man die Entwicklung von Webanwendungen über die Zeit, so ist aktuell ein großer technologischer Sprung zu beobachten. Es besteht jedoch die Möglichkeit, dass der Durchschnittsnutzer, der die Kurzweiligkeit und Einfachheit der Hyperlink-Philosophie schätzt, durch Funktionsvielfalt und Komplexität überfordert wird.

Bis vor kurzem beschränkten sich die Navigations- und Interaktionsmechanismen auf Hyperlinks und gelegentlich Javascripts, die besonders ansprechende Navigationsmenüs ermöglichten.

Momentan gilt im Web die Philosophie "Höher, schneller, weiter, bunter und interaktiver". Insbesondere bei Webangeboten, die komplette Applikationen imitieren (Mailprogramme, Tabellenkalkulationen), sind Usability-Probleme unausweichlich. Entgegen der ursprünglichen Charakteristik des Webangebots ist aufgrund der höheren Komplexität eine lange Lernkurve wie bei typischen Desktop Applikationen zu erwarten. Eine Vielzahl von neuen Funktionen wird dem User verwehrt bleiben, wenn ihm diese nicht mittels durchdachter Usability-Konzepte zugänglich gemacht werden (Maurer, D. 2006).

Die Bemühungen um Bedienbarkeit im Web müssen entsprechend der neuen Bedienkonzepte ausgeweitet werden. Im folgenden Abschnitt werden exemplarisch Interaktionsparadigmen vorgestellt und Lösungsansätze präsentiert, die dem Benutzer den Zugang vereinfachen.

4.0 Interaktionsparadigmen

Viele der unten aufgeführten Interaktionsparadigmen sind dem geneigten Internetbenutzer unbekannt. Anderen Benutzern sind diese Controls aus "Stand Alone"-Applikationen bekannt. Doch auch diese Benutzergruppe ist nicht auf ein derartiges Interaktionsangebot im Web vorbereitet. Mangels Erwartung werden entsprechende Interaktionselemente unter Umständen schlicht übersehen.

Die Aufgabe des Designers besteht in diesem Fall in der Bekanntmachung der betroffenen Controls. Einem Leitsatz der

Usability folgend sollte ein Bedienelement seine Funktion schon durch seine Gestaltung unmissverständlich preisgeben. Dies wird auch als "percieved affordance" bezeichnet (Cooper & Reimann 2003, 256f).

Als Beispiel dienen dreidimensionale Push Buttons. Ihr optisches Herausragen aus dem flachen Screen fordern den Benutzer unmissverständlich zum Drücken auf.

Die Einführung neuer Controls sollte in jedem Fall durch die Verwendung von Demonstrationen und kleinen Tutorials begleitet werden, auf die der Benutzer beim ersten Besuch der Seite aufmerksam gemacht wird (Maurer, D. 2006).

Die folgenden Interaktionselemente bieten in ihrem spezifischen Fall weiteres Optimierungspotential.

4.1 Drag & Drop

Beschreibung:

Ein Element wird durch einen Mausklick aktiviert und gehalten. In diesem Zustand kann es mit der Maus bei weiterhin gedrückter Taste in einen beliebigen Bildschirmbereich verschoben werden. Das Loslassen der Taste deaktiviert das Element und legt es an der aktuellen Position ab. "Drag&Drop" wird häufig verwendet, um einen Sortiervorgang zu realisieren. Durch die direkte Objektmanipulation wird dem Benutzer ein natives Sortiererlebnis suggeriert.

Das Problem besteht in erster Linie darin, dass ein Großteil der Benutzer, zumindest im Webkontext, nicht mit dieser Art der Interaktion vertraut ist. Aus diesem Grund wird die Möglichkeit häufig schlicht übersehen.

Lösungsansatz:

Bei verschiebbaren Elementen hat sich mittlerweile die Verwendung von kleinen "Grip-Flächen" durchgesetzt. Diese "Anfasser" geben in Kombination mit der Veränderung der Cursor-Erscheinung (Pfeil wird zur Hand) einen klaren visuellen Hinweis auf die versteckte Funktionalität. Darüber hinaus sollte das Objekt als ganzes bei Aktivierung farblich hinterlegt werden. So wird dem Benutzer visuell bestätigt, welche Screenteile zu einem verschiebbaren Container gehören.

Die Nachahmung physikalischer Gegebenheiten unserer Umwelt (vgl. Push-Buttons) könnte auch in diesem Fall Anwendung finden. Ein leichtes "Herausheben" aus dem Bildschirmlayout vermittelt dem Benutzer den Eindruck, er könne das Element nun über den restlichen Bildschirminhalt hinweg verschieben.

Ebenfalls sollten Bereiche farblich markiert werden, in denen ein Element sortiert bzw. verschoben werden kann. Dies sollte immer dann visuell geschehen, wenn sich der Benutzer mit dem Maus-Zeiger und dem aktuell verschobenen Element über diesem Bereich befindet.

4.2 Edit in Place

Beschreibung:

Ähnlich eines gewöhnlichen "Inputfeldes" können Inhalte durch den Benutzer direkt editiert werden.

Im Falle einer Web 2.0 Anwendung sind diese editierbaren Bereiche jedoch nicht als erwartete bekannte Formularelemente zu erkennen. Der Gestaltung eines editierbaren Elements sind kaum Grenzen gesetzt. Über verschiedene Webangebote hinweg betrachtet leiden hierunter die Konsistenz und der Widererkennungswert dieses Controls nachhaltig.

Der Editiervorgang beginnt in der Regel mit der Aktivierung eines editierbaren Inhaltes. Dies geschieht beispielsweise durch Doppelklicken des Objektes. Die Editierbarkeit wird dem Benutzer häufig durch einen blinkenden Cursor und einen markierten Inhalt bestätigt.

Wurde der Inhalt editiert, wird das Element wieder deaktiviert. Häufig handelt es sich beispielsweise um eine Tabellenzelle, in die ein neuer Wert eingetragen wurde.

Grundsätzlich beschränkt sich das Problem auf die Bekanntmachung des Controls (vgl. Drag&Drop). Es wird jedoch wesentlich durch den Umstand verstärkt, dass der Benutzer bisher eher mit der passiven Nutzung von Seiteninhalten vertraut ist. Die Möglichkeit der inhaltlichen Gestaltung hält er demnach noch für ziemlich unwahrscheinlich. Dies schränkt die mögliche zufällige Exploration zusätzlich ein.

Lösungsansatz:

Ein verbreiteter Ansatz besteht darin, den Benutzer mit kontextsensitiv eingeblendeten Controls auf zur Verfügung stehende Interaktionsmöglichkeiten hinzuweisen. Bewegt der Benutzer die Maus beispielsweise über ein editierbares Objekt, wird ein Control zur Aktivierung dieser Interaktion direkt (in Place) am Objekt angeboten.

In Kombination mit einem Hilfssymbol könnte der Benutzer zusätzlich auf die Möglichkeit hingewiesen werden, ein editierbares Element nicht durch das entsprechende Control sondern durch einen Doppelklick zu aktivieren. Diese Information könnte ebenfalls in einem für den Benutzer unumgänglichen Popup-Fenster erscheinen, das bei der erstmaligen Nutzung des Webangebots eingeblendet wird.

5.0 Browserverhalten

Internetnutzer haben über die Jahre ihr eigenes mentales "Page Model" verinnerlicht.

Demnach kann nahezu jede inhaltliche Veränderung des Browsers durch den Back-Button rückgängig gemacht werden. Dies gilt jedoch nur für Webinhalte älterer Generationen, in denen jede Inhaltsänderung durch einen Server-Request eingeleitet wurde.

Die durch die "AJAX-Technologie" erzeugte Zustandslosigkeit des Browsers verhindert die automatische Generierung einer History. Somit können alte Zustände nicht ohne weiteres durch die Verwendung der History-Buttons wiederhergestellt werden.

Obwohl die History-Buttons nicht als Undo-Funktion zu verwenden sind, ist der Benutzer geneigt, das ihm vertraute Page-Model auch auf Web 2.0 Anwendungen zu übertragen. Dies stellt sowohl den Benutzer als auch die Entwickler in der Realisierung auf eine harte Probe.

Es existiert mittlerweile eine Hand voll Lösungsansätze für die "Applikation in Applikation"-Problematik. Diese unterscheiden sich jedoch deutlich in Anwendbarkeit und Aufwand und werden im folgenden Kapitel diskutiert.

5.1 Browsercontrols

5.1.1 History Buttons

Im Normalfall wird eine History automatisch aufgebaut, indem jeder Request als neuer Eintrag und Bewegung des Benutzers im Netz registriert wird. Der vorhergehenden technischen Erläuterung folgend wird aufgrund der "Zustandslosigkeit" des Browsers keine gültige History mitgeschrieben. Dies hat ein unerwartetes Verhalten der History-Buttons für den Benutzer zur Folge. Der letzte gesendete Request führt in der Regel auf die zuvor besuchte Internetpräsenz. Damit tritt der schlechteste Fall für jeden Betreiber einer Internetseite ein: Das ungewollte Verlassen seines Webangebots.

Es bestehen verschiedene Minimallösungen, die jedoch nicht als endgültige Maßnahme gelten können. Eine unkomplizierte Methode, die zumindest den oben beschriebenen "Worst Case" ausschließt, besteht im Löschen der History durch ein einfaches JavaScript. Auf diese Weise sind die History-Buttons disabled.

Eine derartige Einschränkung der Kontrolle des Benutzers über die Applikation kann jedoch nicht das Ziel sein. Darüber hinaus wird bei diesem Vorgang der Verlauf besuchter Seiten unwiderruflich gelöscht, so dass der Benutzer keinen Zugriff mehr hierauf hat.

Eine weitere Möglichkeit, die zumindest oberflächlich Abhilfe schafft, besteht in der Konfiguration des Browser-Fensters. Jedes Fenster kann im Aufruf durch ein Javascript mit den gewünschten Controls konfiguriert werden. Auf diese Weise besteht die Möglichkeit ein Fenster zu generieren, das einem Applikationsfenster sehr nahe kommt. Sämtliche Controls des Browsers werden schlicht ausgeblendet. Auf diese Weise wird zugleich das "Applikation in Applikation"-Problem entschärft und der Benutzer wird schon durch die Erscheinung darauf vorbereitet, dass es sich nicht um einen gewöhnlichen Webinhalt, sondern um eine komplexe Applikation handelt.

5.1.2 Bookmarks

Die Problematik ungültiger Bookmarks resultiert ebenfalls aus dem technischen Problem, dem auch die History unterliegt. Allerdings können Bookmarks weitestgehend vernachlässigt werden, da diese zumindest verlässlich auf die Startseite des Webinhalts verweisen.

5.2 Technische Lösung

Einen ernsthaften technischen Lösungsansatz bietet das "Really Simple History Framework" (RSH) (Neuberg, B. 2007).

Das RSH zeichnet die Browser-History von "AJAX-Anwendungen" mit Hilfe eines Javascripts auf und verknüpft diese mit den History-Buttons des Browsers.

Eine detaillierte Dokumentation sowie der Code sind unter http://www.onjava.com/pub/a/onjava/2005/10/26/ajax-handling-bookmarks-and-back-button.html zu finden.

6.0 Beispiel mit konkreten Verbesserungsansätzen (MTV Overdrive)

Repräsentativ für eine typische Web 2.0 Anwendung dient das Video Portal "MTV Overdrive" als konkretes Beispiel, um Schwachstellen zu identifizieren und konzeptionelle Verbesserungen einzuführen.

MTV Overdrive ist ein Video Portal, dessen Archiv auf ausgestrahlten Inhalten des Fernsehsenders MTV basiert. Für den Benutzer besteht die Möglichkeit vergangene Beiträge und Musikclips anzuschauen und zu seiner persönlichen Playlist hinzuzufügen. Die Playlist bietet jederzeit Zugriff auf die gewünschten Inhalte.

6.1 Integration der Browsercontrols

Das "Applikation in Applikation"-Problem wurde auch in diesem Fall nur unzureichend gelöst. Beim ersten Besuch der Seite funktioniert der History-Button erwartungskonform, indem er auf die zuvor besuchte Seite verlinkt. Mit jeder Bewegung innerhalb des MTV Contents, wie beispielsweise der Abruf eines neuen Videos, wird ein neuer History-Eintrag angelegt. Entgegen den Benutzererwartungen, jeder Eintrag beziehe sich auf ein zuvor abgespieltes Video, besitzt die History keinerlei Navi-

gationsfunktionalität. Es handelt sich um "blinde" Einträge, die durch ihre Masse lediglich daran hindern, ungewollt die Seite zu verlassen.

Eine Verbesserung der Benutzbarkeit könnte durch die Integration einer globalen Video-History erzielt werden, die dem Benutzer unabhängig von seiner Playlist einen Überblick über die zuletzt betrachteten Videos gibt. Diese History sollte mit Hilfe des RSH (vgl. Kapitel 5.2) auf den Browsercontrols abgebildet werden.

6.2 Support neuer Interaktionselemente

MTV Overdrive verzichtet fast gänzlich darauf, den Benutzer gezielt mit den neuen Controls vertraut zu machen. Ein textueller Hinweis an einer in der Navigationsstruktur versteckten Stelle muss in diesem Fall ausreichen, um die "Drag&Drop"-Funktionalität bekannt zu machen.

Die "Drag&Drop"-Funktionalität sollte bei erstmaligem Seitenaufruf durch eine Funktionsdemonstration eingeführt werden. Die Demonstration zeigt schematisch, an welcher Stelle bestimmte Elemente verschoben werden können.

6.3 Integration des "Drag&Drop"-Mechanismus

Der "Drag&Drop" Mechanismus wurde nicht konsequent genug umgesetzt. Es besteht lediglich die Möglichkeit, die Einträge der Playlist durch "Drag&Drop" zu sortieren. Wünschenswert wäre die Ausweitung von "Drag&Drop" auf das Hinzufügen und Abspielen von Playlist-Einträgen. Dies könnte folgendermaßen geschehen:

Die Referenz eines laufenden Videos wird per "Drag&Drop" auf den Reiter "Playlist" gezogen. Dies wird durch ein kleines "Overlay" quittiert, das direkt an dem Reiter platziert ist (Beispieltext: "You've added 1 clip"). Darüber hinaus wird die Playlist geöffnet, um dem Benutzer den aktuellen Inhalt seiner Auswahl anzuzeigen.

Auf die gleiche Art und Weise könnten Videos aus der "Channel-View" hinzugefügt werden. Die "Channel-View" präsentiert eine kategorisierte Auswahl von Videos, die teilweise zu Beitragsgruppen zusammengefasst sind, welche wiederum mehrere Clips umfassen. Dies wird durch eine einfache Baumstruktur präsentiert. Hierbei sollte die Möglichkeit bestehen, sowohl einzelne Clips als auch ganze Gruppen per "Drag&Drop" hinzuzufügen.

Die konsequente Umsetzung lässt ein intuitiveres Nutzungsverhalten und eine verbesserte "User Experience" erwarten.

6.4 Fazit MTV Overdrive

Das Beispiel MTV Overdrive zeigt, dass die gezielte Verwendung typischer Web 2.0 Controls durchaus Vorteile in der Bedienbarkeit mit sich bringt und eine gesteigerte "User Experience" erwarten lässt.

Vor ihrem Einsatz sollte jedoch sowohl der thematische Kontext als auch die erwartete Zielgruppe auf ihre Eignung überprüft werden.

Im Fall von MTV Overdrive eignet sich das Paket aus moderner Erscheinung, junger Zielgruppe und technologischem Gesamtzusammenhang sehr gut zur Einführung typischer Web 2.0 Controls.

7.0 Fazit

Kapitel 6 zeigt deutlich, dass bei gezieltem und dosiertem Einsatz neuer Interaktionselemente eine deutliche Steigerung sowohl der intuitiven Benutzung als auch des Nutzungserlebnisses ("User Experience") zu erwarten ist.

Entwickler von RIAs sollten jedoch nicht der Versuchung unterliegen, Controls aus falschen Motivationen zu implementieren. Controls sollten keinen allgemeinen Trends folgen, sondern in jeder Nutzungssituation auf ihre Tauglichkeit und eine mögliche Steigerung der Benutzerfreundlichkeit geprüft werden.

Darüber hinaus muss der Benutzer langsam an das neue Bedienumfeld herangeführt werden.

Ein durchdachtes Interaktionskonzept in Kombination mit einer ausgereiften Informationsstruktur ist auch weiterhin die beste Garantie für eine wachsende Benutzergemeinschaft.

8.0 Literaturverzeichnis

Cooper, A.; Reimann, R. (2003): About Face 2.0 – The Essentials of Interaction Design. Indianapolis, Indiana: Wiley Publishing, Inc.

Keith, J. (2007): Ajax & Accessibility. http://north.webdirections.org/presentations/AjaxAndAccessibility.pdf

Maurer, D. (2006): Usability for Rich Internet Applications. http://www.digital-web.com/articles/usability_for_rich_internet_applications

Neuberg B. (2005): AJAX – How to Handle Bookmarks and Backbuttons. http://www.onjava.com/pub/a/onjava/2005/10/26/ajax-handling-bookmarks-and-back-button.html

9.0 Beispiele für mit AJAX realisierte Web 2.0 Anwendungen

http://www.mtv.de/overdrive/index.php

http://www.flickr.com/

http://maps.google.com