Usability im Web-2.0-Zeitalter

«Rich Internet Applikationen» stellen nicht weniger als eine kleine Revolution bei der Bedienung von Websites dar. Allerdings ist die Sicherstellung der Usability wichtiger denn je.

Von Angie Born und Peter Hogenkamp

Zehn Jahre lang haben wir uns beim «Surfen» im Web an eine Kombination gewöhnt: klicken, warten... neu orientieren... klicken, warten... und so weiter. Feedback, zum Beispiel nach Ausfüllen eines Suchformulars, bekam man immer erst auf der nächsten Webseite.

Der Browser war damit eigentlich ein schlechtes interaktives System, denn als User erwartet man schnelles Feedback: innerhalb von 0.1 Sekunden sollte das System zeigen, dass es die Eingabe registriert hat, und innerhalb von 1 Sekunde sollte das Ergebnis vorliegen – sonst geht man von einem Fehler aus. Seit Modem-Zeiten, in denen diese Vorgaben grotesk überschritten wurden, sind die Wartezeiten zwar deutlich kürzer geworden, aber auch mit den heutigen Breitband-Verbindungen reichen «klassische» HTML-Webseiten immer noch nicht an eine lokale Applikation heran. Dies ändert sich nun auf vielen Websites, dank «Rich Internet Applications», kurz RIA. Sie zeichnen sich dadurch aus, dass der Client laufend Daten mit dem Server austauschen und zwischenspeichern kann. Der Server lädt teilweise Daten «auf Vorrat», und der User erhält wirklich unmittelbar Feedback auf seine Eingabe, oder es wird schnell nachgeladen, auch ohne dass der User die Seite verlässt. Somit fühlen sich RIAs sehr viel «fliessender» an als klassische, sequenzielle Websites.

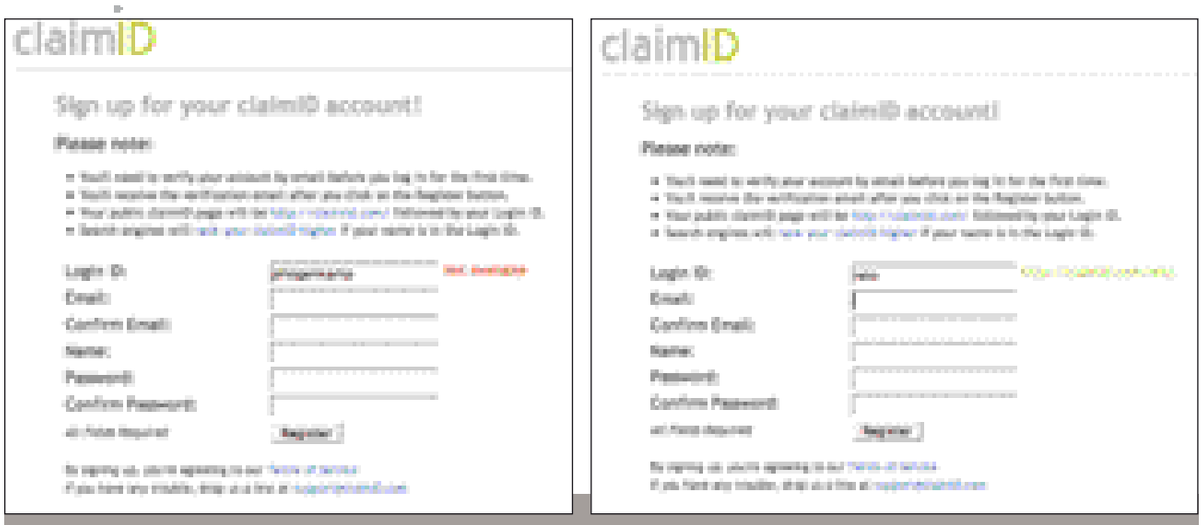
Die am weitesten verbreitete RIA-Technologie ist «Ajax», gefolgt von Adobes «Flex», das vor allem für komplexere Applikationen langsam zum Lieblingskind der Programmierer wird.

Die Umsetzungen von RIA sind unterschiedlich. Es können ganze Applikationen darauf basieren oder nur kleine Features genutzt werden. Neben dem Preloading von Daten aus dem «Umkreis» der aktuell angezeigten kann das System auch schnell Feedback geben, ohne den User aufzuhalten. Beispiel Registrierung: Der User, der den Username «retomueller» registrieren will, bekommt kurz nachdem er das Feld verlassen hat, Feedback, dass dieser Name schon besetzt ist (typischerweise ein grüner Haken neben dem Feld im positiven und ein rotes Warnsymbol im negativen Fall).

Bei ClaimID zum Beispiel sehe ich, während ich noch meine Mailadresse eingebe, bereits, dass mein oben eingegebener gewünschter Username nicht mehr verfügbar ist: «Not available». Im Erfolgsfall dagegen zeigt mir das System in Grün meine zukünftige URL an http://claimid.com/reto.

Gmail als Vorreiter

Die erste populäre «grosse» Ajax-Anwendung war Googles «Gmail» (private Beta ab 2004). Rund zehn Jahre nach dem ersten Gratis-Angebot Hotmail wurde Webmail durch den Einsatz von Ajax völlig neu definiert.



Schnelles Feedback: Username «phogenkamp» ist schon besetzt (links), «reto» ist noch frei (rechts).

Bei Gmail werden die Inhalte einiger E-Mails im Hintergrund heruntergeladen, so dass diese ohne Verzögerung angezeigt werden können. Auch das gesamte Adressbuch des Users wird von der Ajax-Engine zwischengespeichert. Tippt man nun im «To»-Feld die Anfangsbuchstaben «gre» des Empfängers ein, erscheint sofort – das heisst wirklich unmittelbar im Sinne eines guten Interaktionssystems, nämlich nach einem Sekundenbruchteil – die vollständige Mailadresse: «Gregor Urech <gregor.urech@zeix.com>». Nichts Neues für Outlook-Nutzer, aber neu im Web; wer zuvor bei einem klassischen Webmail-Client wie Hotmail oder GMX eine Adresse aus dem Adressbuch abrufen wollte, brauchte dazu etwa 10 Klicks (Adressbuch aufrufen, Suchphrase eingeben, Suche durchführen, Adresse übernehmen) und etwa 15 Sekunden.

Inzwischen ist bei Webmail der Einsatz von RIA bereits mehr oder weniger Pflicht. Microsoft und Yahoo sind mit gebührendem Abstand dem Vorbild von Google gefolgt und haben eigene RIA-Webclients entwickelt, Microsoft mit «Windows Live Hotmail», Yahoo unter Einsatz von Adobe Flex. Sunrise nutzt seit einigen Wochen Gmail als technische Plattform.

Dabei sind die Online-Mailapplikationen inzwischen so weit gekommen, dass sie es – eine schnelle Internet-Verbindung vorausgesetzt – in Sachen Performance problemlos mit den eigentlich «grossen Brüdern» wie Outlook/Exchange oder Lotus Notes/Domino aufnehmen können. Pikanterweise haben sie diese trotz des «Standortnachteils» vielfach bereits überholt: Eine Volltextsuche in Gmail – auf Servern irgendwo auf der Welt in der sagenumwobenen «Cloud» von Google – ist inzwischen schneller als in einer vorindexierten Mail-Datenbank mit Lotus Notes, dessen Server im selben Gebäude steht.

Jetzt schon ein Ajax-Klassiker: Mapping

Das schon jetzt «klassische» Beispiel für den Einsatz von Ajax sind Mapping-Applikationen, Google Maps und map.search.ch wurden etwa gleichzeitig entwickelt. Landkarten und Stadtpläne sind die perfekte Anwendung für das asynchrone Nachladen von weiteren Daten: Zuerst wird gerendered und geliefert, was der User auf dem Bildschirm sieht, dann weitere Kartenausschnitte rundherum. Wer nun mit einer schnellen Internet-Leitung die Karte – stufenlos – verschiebt, dem kommt es vor, als sei sie unendlich nach allen Seiten.

Nach zwei bis drei Jahren Ajax-Maps sieht man auch das Fortschreiten der eigenen Erwartungshaltung: Bei der ersten Nutzung kam es einem gar nicht so besonders vor – aber wenn man heute mal wieder eine Kartenapplikation nutzen muss, die noch nach der alten Methode «springt», sobald man auf einen Pfeil am Kartenrand klickt, kommt man sich vor wie in der Web-Steinzeit. Diese Entwicklung – ohne RIA wird man zur Anwendung zweiter Klasse – dürfte sich in Zukunft verstärkt auch auf andere Typen von Applikationen ausweiten.

Ajax-basierte Anwendungen erfreuen sich in der Schweiz ohnehin grosser Popularität. Mit map.search.ch 2005, der «aufgebohrten» Version immo.search.ch 2006 und local.ch 2007 gewannen drei Jahre nacheinander Ajax-Sites den «Best of Swiss Web»-Master.

Einmal mehr fällt jedoch auch der Unterschied zwischen den USA und Europa auf. In den USA sind bestimmte Funktionsweisen (wie die oben genannte Live-Abfrage des Usernames bei der Registrierung) bereits Standard bei allen neu entwickelten Websites, in Europa dagegen ist es noch eher die Ausnahme.

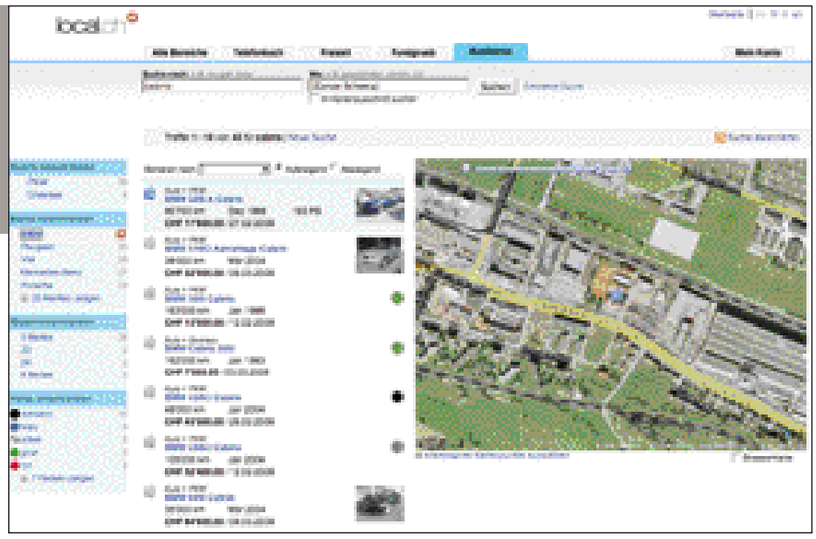
Einstieg schaffen

Wichtig bei Ajax-Applikationen ist es, die User zu Beginn der Nutzung nicht zu überfordern. Viele nehmen den Unterschied zu «klassischen» Websites nämlich gar nicht bewusst wahr, wenn die Applikation gut gemacht ist und vom Desktop bekannte Funktionalität übernimmt.

So ist der Unterschied zwischen der lokalen Browser-Funktion «AutoAusfüllen» und der (bisher nur in den «Google Labs» verfügbaren) Funktion «Google Suggest» kaum merklich – obwohl fundamental, denn beim einen wird nur die lokale Browser-History durchsucht, beim anderen nach der Eingabe jedes Zeichens die Google-Datenbank.

«Refinement» mit Ajax

Eine weitere nützliche Funktion ist die Vorschau der Trefferanzahl. Immer mehr Sites zeigen bei einer strukturierten (d.h. aus mehreren Feldern bestehenden) Suche schon



Die Autosuche bei auto.local.ch mit der Live-Treffervorschau in der linken Spalte http://auto.local.ch/de/q/cabrio.html#makeref=BMW

während des Ausfüllens der Maske die «Konsequenz» der aktuellen Suchkombination an. Schränkt man bei der Autosuche auf auto.local.ch die Treffer auf eine Marke ein, aktualisieren sich die Trefferzahlen der anderen Kategorien sofort. Ich sehe also auf einen Blick, dass meine Suche nach «Cabrio» 64 Autos mit der Farbe «schwarz» findet; schränke ich weiter auf die Marke «BMW», gibt es nur noch 10 Treffer.

Interessant ist, dass in Usability-Tests das Verständnis dieser schrittweisen Einschränkungen («Refinement» genannt) ohne die Treffervorschau sehr schlecht war: Viele Testpersonen begriffen nicht, wofür die linke Spalte überhaupt gut sein sollte, und nach welchen Kriterien die Einschränkungen erschienen. Erst durch die kleine graue Zahl mit der Anzahl begriffen die Leute das Erscheinen und Verschwinden von zusätzlichen Einschränkungskriterien – und waren zudem sehr glücklich über die «Vorschau» der Suchtreffer. Die Nutzung des gesamten Features wurde plötzlich statt «verwirrend» und «störend» als «super nützlich» und «sehr einfach» bezeichnet – ein Indiz, wie wichtig Testing bei der Entwicklung von Ajax-Applikationen ist, denn nicht alles, was technisch möglich ist, wird auch von den Usern als Erleichterung der Arbeit wahrgenommen.

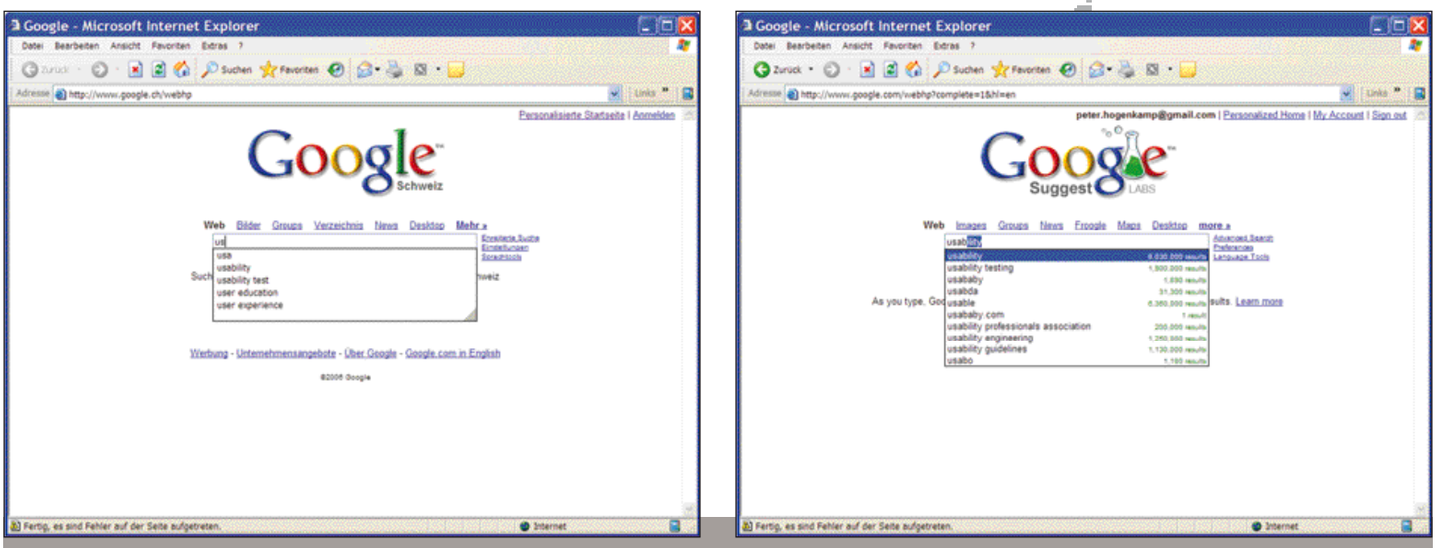
Drag & Drop jetzt auch im Browser

Gerade für Applikationen ist auch die Drag-&-Drop-Funktion sehr nützlich. So lassen sich zum Beispiel Umfragen erheblich einfacher gestalten: In einer Umfrage zu Skihandschuhen musste man zehn vorgegebene Hersteller-Marken zuordnen nach «bereits gekauft», «in Erwägung gezogen», «bekannt» und «unbekannt».

Statt in endlosen Radio-Button-Wäldern herumklicken zu müssen (immerhin 50 für die erste Frage), konnte man die Marke einfach in das entsprechende Feld ziehen. (Ob allerdings jeder Nutzer die Anweisung «Bitte ziehen Sie die Marken in die entsprechenden Felder per Drag & Drop» versteht, sei dahingestellt.)

Auch das Personalisieren von Websites (oder Intranets) durch das reine «Herumschieben» von Inhaltscontainern wird viel einfacher, intuitiver und somit schneller – und somit überhaupt erst einigermassen realistisch, denn in den ersten zehn Web-Jahren war den meisten Leuten Personalisierung generell zu aufwendig, so dass sie ein frommer Wunsch der Anbieter blieb.

Die Aufgabe «Elemente auf dem Bildschirm neu anordnen» wird um ein Mehrfaches einfacher: Bei der klassischen Variante musste man zunächst vom normalen Modus in den Bearbeitungsmodus wechseln (Link «Reihenfolge festlegen»). Dann kann man ein Element markieren und mit den Links «nach oben» und «nach unten» befördern, um schliesslich die



Browser-History versus Live-Abfragen via Internet: Die lokale Suche nach «usa» holt Inhalte aus dem Browser (links), «Google Suggest» fragt live die Datenbank nach Suchen mit «usab» ab (rechts).

Zehn Marken sind schnell in die fünf Felder gezogen.

neue Reihenfolge zu «speichern». Mit einer RIA-Applikation und Drag & Drop dauert das gleiche eine halbe Sekunde – und ist deutlich intuitiver.

So begeistert war die US-Szene von den neuen Möglichkeiten, dass im Jahr 2005 ein regelrechter Boom einsetzte, Desktop-Applikationen auf dem Web nachzubauen, der bis heute nicht abgeebbt ist.

So schossen etwa rund ein Dutzend Browser-Startseiten-Startups wie Pageflakes und Netvibes aus dem Boden (sowie Google das gleiche mit «iGoogle» realisierte), mit denen sich eine persönliche Browserstartseite mit den beliebtesten Funktionen und News zusammenstellen lässt. Eine marktbeherrschende Stellung scheint allerdings niemand erreicht zu haben. Auch vieles andere wird ins Web portiert, bis zu hoch komplexen Applikationen wie Photoshop. Denn RIA und das andere Trend-Thema «Software as a Service» hängen zusammen: Nur mit RIA-Methoden ist es möglich, Online-Software so attraktiv zu gestalten, dass SaaS überhaupt denkbar wird – und die ganzen übrigen Vorteile wie Ortslosigkeit etc. greifen.

Einfacher wird die Benutzung durch Ajax vor allem dort, wo man einen Prozess auch näher an das «mentale Modell» des Users heranführen kann. Das Aufgeben einer Kleinanzeige war bisher zum Beispiel ein Task, der nicht viel mit dem fertigen Layout des Inserats zu tun hatte: Man füllte eine lange Maske mit vielen Optionen aus, und heraus kam ein völlig anders aussehendes Inserat. Heute dagegen kann man quasi direkt in das Layout reinschreiben, was ungleich intuitiver ist.

Auch beim Gratis-Online-Fotodienst Flickr kann man den Titel und die Beschreibung eines Bildes direkt durch einen einfachen Mausklick auf den bestehenden Text anfügen. Dieses «In-Page-Editing» an Ort und Stelle schneidet in Usability Tests sehr gut ab, da es den Erwartungen der User entspricht.

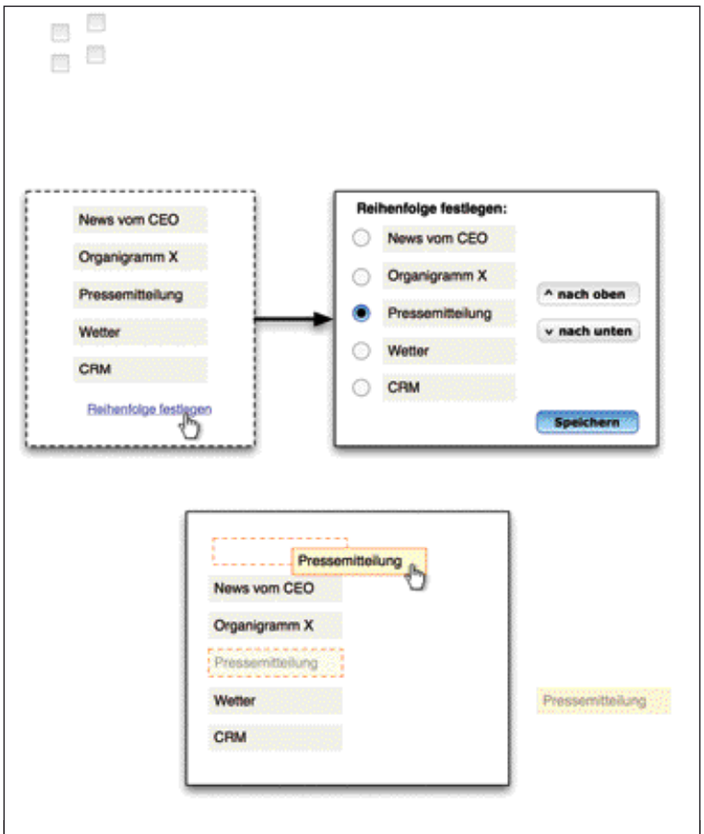
Nicht nur Vorteile

Doch bedeutet der Einsatz von Ajax, dass auch die Usability automatisch besser wird? Nicht unbedingt. Es heisst zuerst einmal, dass man umfangreicher testen muss, denn die User sehen sich einer neuen Logik der Benutzerschnittstelle gegenüber. Das Surfen im Web 1.0 war zwar langsam und langweilig, aber es war vertraut.

Ein typisches Beispiel sind Checkboxen: Eigentlich ist die immer noch geltende Usability-Regel, dass eine Checkbox nicht direkt zu einer Aktion führen darf, sondern die Maske erst mittels «Submit»-Button abgesandt werden muss. Dieses Prinzip wird jedoch von den meisten RIA-Applikationen unterlaufen, indem heute viele Aktionen direkt mit dem Klick auf die Checkbox ausgeführt werden, ein «Absenden» ist nicht mehr nötig. In einem Test von immo.search.ch brachte ein rotes PLZ-Feld eine Nutzerin zur Verzweiflung - sie hatte nicht verstanden, dass das Hinterlegen des Felds mit Rot direkt beim Verlassen bedeutete, dass sich mit dieser PLZ überhaupt keine Treffer finden lassen.

Oft ist, während der weniger versierte Nutzer nach dem Button sucht, mit dem er seine Aktion ausführen kann, das Gewünschte schon längst geschehen – und eventuell unbemerkt geblieben, während der Benutzer immer noch wartet und sich fragt, was das System denn gerade macht. Nur sehr gutes Feedback-Design kann dieses Problem lösen (manchmal hilft auch ein «Dummy»-Button, das gewohnte Gefühl wiederherzustellen).

Ähnlich ist es auch mit der bereits erwähnten Drag-&-Drop-Funktion im Browser: Wenn die Möglichkeit nicht visuell angedeutet wird, wird sie von vielen Nutzern gar nicht erwartet. Ist also keine Alternative geboten, sehen wir oft ratlose Nutzer, die nicht weiterkommen.



Web-Elemente umsortieren: «klassisch» eine mühsame Sache (links), mit Ajax einfach und intuitiv (rechts).

Hinzu kommt, dass auf vielen heutigen Websites noch dieselben Usability-Fehler gemacht werden wie vor zehn Jahren. Wenn diese in das neue Zeitalter eintreten, muss es nicht unbedingt besser werden. Bei Eingabeformularen zum Beispiel bestehen manche Online-Shops immer noch darauf, dass man die Kreditkartennummer statt mit Leerzeichen (wie auf der Karte angegeben) konsequent ohne Leerzeichen (wie intern übermittelt und abgespeichert) eingibt. Wenn nun diese mediokren Formularentwickler denken, mit Ajax sei alles per se einfacher, dann kommen neue Probleme auf uns zu.

Ebenfalls einen schlechten Einfluss auf die User Experience hat die Ladegeschwindigkeit. Zwar fühlen sich viele moderne RIA «nahtloser» an, dies jedoch oft nur mit einer guten Internet-Verbindung und meist erst, wenn ein erster Teil der Applikation geladen ist. Gerade Flex-Anwendungen haben beim Start der Session diesen grossen Nachteil. Will man zum Beispiel im australischen Online-Shop für Camping- und Fischereizubehör «BCF», der ein wegweisendes RIA-Benutzungskonzept aufweist (store.bcf.com.au) schnell ein paar Wanderschuhe suchen, wartet man selbst mit einem schnellen Breitband Anschluss (4 Mbit/s bei uns im Büro) fast zwei Minuten, bis die Applikation geladen ist. Bei mobilem Zugang ist somit an eine Nutzung fast nicht zu denken.

Neben der Geschwindigkeit schliessen auch die Kosten eine mobile Nutzung oft aus. Wer pro Kilobyte bezahlt, wird sicher nicht wollen, dass megabytegrosse Kartenausschnitte, die er sich vielleicht nie anschauen wird, «auf Verdacht» heruntergeladen werden. Daher wird jeweils auch eine Version ohne Ajax-Unterstützung bereitgestellt werden müssen.

Einmal mehr: Die Usability macht's

RIA-Applikationen können sehr gut werden, wenn sie von Teams entwickelt werden, die sehr sorgfältig umsetzen und sehr viel Wert auf Usability legen, was zum Beispiel bei Google der Fall ist. Ist das nicht der Fall, ist man schnell mal schlechter gestellt. Der Einsatz von Ajax und Verwandten bedeutet deutlich mehr Testing, und zwar sowohl Funktionstests – viel davon auf Browserkompatibilität – wie auch Usability-Tests. Für «faule» Webdesigner ist er also sowieso nicht geeignet.

Aber wie gesagt, das heisst nicht, dass man sicherheitshalber besser auf den Einsatz von RIA-Technologien verzichten sollte. Wenn ein neuer Mitbewerber kommt und diese einsetzt, dürften die erfahrenen User sehr schnell abwandern – und die anderen langsam hinterher.

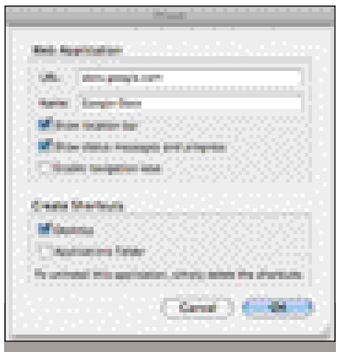
Angie Born und Peter Hogenkamp sind Partner der Usability-Agentur Zeix AG, die z.B. das User Interface von local.ch aus Usability-Sicht mitentwickelt hat.

RIA-Technologien im Vergleich

Von Urs Binder

Online oder offline, im Browser oder über eine eigenständige Runtime-Engine: Rich Internet Appliacations können vielfältig daherkommen.

Rich Internet Applications, kurz RIAs, sind laut Wikipedia «Webanwendungen, die mit den Features und der Funktionalität herkömmlicher Desktop-Anwendungen aufwarten». Damit sich dieses Development- und Deployment-Modell auch durch Entwickler realisieren lässt, denen die Feinheiten von HTTP und Konsorten nicht bis ins Detail geläufig sind, haben zahlreiche Anbieter Tools und Frameworks für die RIA-Entwicklung auf den Markt gebracht – oder sie stecken, wie in manchen Fällen, gerade mitten in der Entwicklung.



Mit Mozilla Prism lässt sich jede Web-Anwendung in eine RIA umwandeln.

RIA – was bedeutet das?

Im Vergleich zu einer herkömmlichen Webanwendung, meist als Abfolge von HTML-Formularen mit Programmlogik auf dem Webserver und einer Datenbank als Grundlage realisiert, bietet eine RIA ein stark grafikorientiertes Interface mit allen Bedienungselementen, die man von Desktop-Anwendungen kennt.

Je nach Technologie kommt als Client ein gewöhnlicher Browser zum Einsatz, der mit DHTML und Javascript «RIA-fähig» wird, oder ein Browser-Plug-in wie Flash oder Silverlight mit eigener Präsentationsschicht.

Flash- und Silverlight-Anwendungen ermöglichen mit ihrer Unterstützung für Animation und Multimedia-Elemente eine «reichere Benutzererfahrung» als rein auf Webstandards gebaute RIAs. Das kann sich aber auch nachteilig auswirken: Beide Technologien bieten bei der Gestaltung der Oberfläche praktisch unbegrenzte Freiheit, so dass zum Beispiel ein Button nicht als solcher erkennbar ist oder ein Scrollbalken untypisches Verhalten zeigt.

Webanwendungen ohne Browser

RIAs sind aber auch ganz ohne Browser möglich, wenn für Präsentation eine eigenständige Engine eingesetzt wird, ein sogenanntes RIA-Runtime. Damit fällt ein Problem weg, an dem

RIA-Technologien im Vergleich

Technologie	AIR 1.0	Flash 9/CS3	Flex 3	Curl Platform 5	Gears
Anbieter	Adobe	Adobe	Adobe	Curl	Google
Charakteristik	Runtime für Deployment ausserhalb des Browsers	Browser-Plug-in für interaktive Oberflächen und Medienpräsentation	Open-Source-Framework für HTML/Flash-basierte Anwendungen	Plattform für Business-orientierte RIA mit Runtime-Engine	Browser-Plug-in für Local Storage und Ajax
Status März 2008	released	released	released	released	Beta
URL	www.adobe.com/products/air/	www.adobe.com/products/flash/	www.adobe.com/products/flex/	www.curl.com	http://gears.google.com/
Entwicklung					
Framework	Flex	-	Flex	Curl	–
Local Storage	■	beschränkt («Shared Objects»)	beschränkt («Shared Objects»)	■	■
Entwicklungsumgebung	Web-Editoren, Flex Builder, Flash CS3, AIR SDK	Flash CS3	Flex SDK, Flex Builder, Flash CS3, BlazeDS	Curl IDE	Web-Editoren
Sprachen/Methodologien	HTML, diverse gängige Web-Entwicklungssprachen	ActionScript	HTML, diverse gängige Web-Entwicklungssprachen	Curl Language	alle Web-Entwicklungssprachen
Deployment					
Deployment ohne Browser	■	■	□	□	□
Offline-fähig	■	□	□	■	■
Runtime-Engine	AIR	Browser/Flash Player 9	Browser/Flash Player 9	Curl RTE	Browser (IE oder Firefox 1.5+)
Grösse Runtime-Engine/ Plug-in	10 MB	– (Flash Player meist schon vorinstalliert)	– (Flash Player meist schon vorinstalliert)	12 – 38 MB	< 1 MB
Systemplattformen	Windows, Mac OS X	Windows, Mac OS X, Linux, Solaris	Windows, Mac OS X, Linux, Solaris	Windows, Linux, Mac OS X (Beta)	Windows, Mac OS X, Linux

■ = ja, □ = nein; k.A. = keine Angaben; 1)