

## Wie HTML5 Rich Internet Applications verändert



# Reicher werden

## Nikolaos Kaintantzis

Der Begriff Rich Internet Application – kurz RIA – ist zwar gängig, aber nicht konkret definiert. Wer eine RIA entwickeln will, sollte genau wissen, was sie von Thin und Rich Clients unterscheidet. Und vor allem: welche neuen Funktionen HTML5 für diese Aufgabe mitbringt.

**R**ich Internet Applications (RIAs) sollen vor allem eins gewährleisten: eine den Desktop-Clients vergleichbare komfortable Bedienbarkeit im Browser. Google hat mit gut aussehenden und leicht zu bedienenden Anwendungen vorgemacht, wie RIAs sich präsentieren können. Die neue „Richness“ im Internet ist allgegenwärtig. Moderne Web-Clients für E-Mail, Wegbeschreibungen oder Suchfunktionen für Adressen haben aber bei den Anwendern auch neue Erwartungen bezüglich Interaktivität und Leistungsfähigkeit geweckt, die sie von Desktop-Applikationen her kennen – eine Herausforderung für die Entwickler neuer Applikationen. Daher lohnt es sich, die Konzepte und die Positionierung von RIAs, Thin und Rich Clients zu beleuchten und im Kontext von HTML5 eine aktuelle Auslegung vorzunehmen. Wer will, kann an dieser Stelle in einem kleinen Quiz er-

mitteln, was er selbst unter einer RIA versteht (siehe Quiz-Kasten).

Im Fokus erfolgreicher Applikationen stehen der Benutzer und seine Aufgaben. Nur wenn man sich auf dessen Interaktionsverhalten konzentriert, kann man seine Erwartungen erfüllen oder sogar übertreffen. Dabei können Anforderungen an die Benutzerinteraktion so unterschiedlich formuliert sein wie „Ständige Datenerfassung und Interaktion“ und „Daten lesen“.

Im ersten Fall ist das Eingeben von Daten eine der Hauptaufgaben des Anwenders, beispielsweise bei einer Schadenserfassungssoftware von Versicherungsmitarbeitern oder bei Entwicklungsumgebungen. Die Applikation soll schnell sein und kurze Antwortzeiten gewährleisten. Hilfreich dabei sind die Navigation mit der Tastatur und das Verwenden von Tastenkürzeln für häufige Aufgaben. In diesem Fall unter-

stützt eine lokal installierte Applikation den Benutzer am besten.

Im zweiten Fall will der Benutzer schnell informiert werden. Er selbst gibt Informationen – wenn überhaupt – nur in Form längeren, unformatierten Textes in ein einziges Feld respektive wenige Felder ein. Das sind typischerweise Webapplikationen wie Nachrichtenportale, Blogs und Foren.

Neben der Konzentration auf die Benutzer gibt es betriebliche Aspekte und Projektrahmenbedingungen, die relevant sind. Dazu gehören Anforderungen wie das Umgehen mit unterschiedlichen Versionen einer Applikation oder dass Applikationen auf jedem Firmen-PC vorhanden respektive von jedem Firmen-PC aus erreichbar sein müssen.

Ein häufig verlangter Aspekt ist Offline-Fähigkeit. Beispielsweise soll ein Außendienstmitarbeiter beim Kunden vor Ort eine Offerte ohne Netzverbindung erstellen. Wenn er zurückkommt und der Laptop am Netz ist, soll die Applikation beginnen, Daten zu synchronisieren. Ein anderes Beispiel: Benutzer sollen E-Mails offline schreiben können, die automatisch abgeschickt werden, wenn das Netz wieder verfügbar ist.

Will man schnell viele Personen erreichen, dient dazu häufig eine Applikation, die man nicht installieren muss. Der Anwender soll nicht warten müssen, bis die Applikation benutzbar ist. So soll er etwa die News einer Zeitung schnell lesen können, ohne zuvor ein Plug-in zu installieren oder ein Programm zu aktualisieren. Keine Installation bedeutet zudem, dass man keine Infrastruktur für ein effizientes und schnelles Rollout benötigt und so die Vertriebsinfrastruktur einsparen kann.

In diesem Umfeld von Forderungen, Rahmenbedingungen und Benutzeransprüchen ist es wichtig, die unterschiedlichen Client-Kategorien Thin, RIA und Rich auseinanderzuhalten. Nur so kann man die richtige Kategorie und schließlich die richtige Technik für eine Aufgabe wählen.

## RIA ist nicht gleich RIA

Das RIA-Konzept befindet sich im Spektrum zwischen klassischen seitenbasierten Webseiten (Thin Clients) und Stand-alone-Applikationen (Rich Clients) (siehe Abbildung 1). Zu Ersteren gehören etwa die Internetseiten von Zeitungen, während man zu den Stand-alone-Applikationen Pakete wie MS Office, Photoshop oder Entwicklungs-

## Quiz: Was ist eine RIA?

Der Begriff RIA ist allgegenwärtig, doch selten verstehen zwei Personen dasselbe darunter. Zwei Fragen sollen ermitteln, welches Verständnis ein Leser an dieser Stelle des Artikels von RIA hat (bitte die Punkte in Klammern addieren).

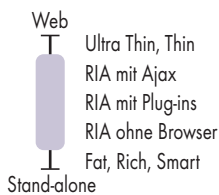
**Frage 1:** RIAs vereinfachen die Verteilung der Applikation, weil der Anwender (außer dem Browser) nichts installieren muss, um mit ihr zu arbeiten.

- (1) Stimme ich voll zu.  
(2) Stimme ich nicht zu.

**Frage 2:** RIAs bieten dieselben Möglichkeiten und denselben Bedienkomfort wie lokal installierte Applikationen.

- (10) Stimme ich voll zu.  
(20) Stimme ich nicht zu.

Die Auswertung des Quiz erfolgt am Ende des Artikels.



**Aus Sicht der Informatik gibt es unterschiedliche RIA-Typen (Abb. 1).**

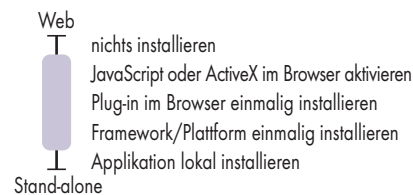
werkzeuge wie Eclipse rechnet. Aus technischer Sicht lassen sich RIAs in drei Kategorien einteilen:

– **RIA mit Ajax**, also auf Basis von JavaScript wie map.search.ch oder Google Mail,

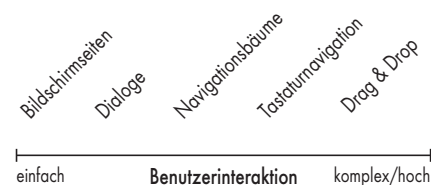
– **RIA mit Plug-in**, zum Beispiel der Routenplaner Map24.com, der auf Java-Applets basiert, oder Photoshop Light/Express (Flash-Client),

– **RIA ohne Browser** kann man technisch als Spezialfall von „RIA mit Plug-in“ verstehen. Der Anwender schaut sich die Applikation im Browser mit einem Plug-in an, und wenn sie ihm gefällt, kann er sie auf den Desktop ziehen oder herunterladen. Beispiele hierfür sind Java Web Start und Adobe AIR.

HTML5 gehört technisch zu „RIA mit Ajax“. Im Wesentlichen erweitert und vereinfacht es die bisherige Version der Hypertext Markup Language. Hinzu kommen viele neue JavaScript-Schnittstellen sowie Neuerungen in CSS3.



**Für den Anwender ist wichtig, was er installieren muss (Abb. 2).**



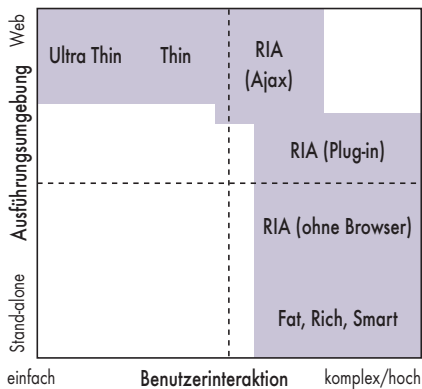
**Der Grad der Benutzerinteraktion in einer Applikation reicht von einfach bis komplex (Abb. 3).**

So weit die technische Sicht. Die Anwendersicht verdeutlicht Abbildung 2. Je näher eine Client-Kategorie in den Bereich „Stand-alone“ vordringt, desto mehr Vorarbeit muss der Anwender leisten, bis er eine konkrete Applikation benutzen kann. Gegebenenfalls ist er nicht in der Lage, diese Vorarbeiten auszuführen, etwa weil er JavaScript aus Gründen der Firmen-Policy gar nicht aktivieren kann oder keine Administratorrechte für die Installation einer Applikation auf seinem Rechner hat.

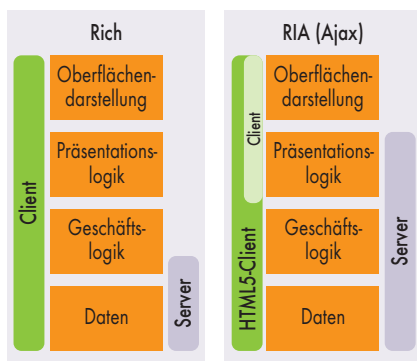


- Webapplikationen gibt es in diversen Ausprägungen – von der einfachen Webseite bis zur Rich Internet Application –, die sich für unterschiedliche Aufgaben eignen.
- Wichtige Kriterien bei der Wahl der richtigen Client-Technik sind unter anderem der Grad der Benutzerinteraktion und die durch die Projektrahmenbedingungen bestimmte Ausführungsumgebung.
- Während sich bisher einige Funktionen nur durch zusätzliche Bibliotheken oder Plug-ins realisieren ließen, können Entwickler sie jetzt nativ in Browsern nutzen, die HTML5 unterstützen.

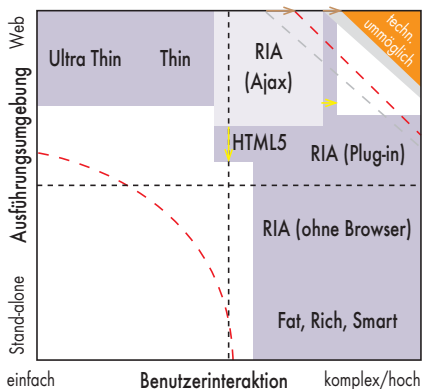
Bei der Entscheidungsfindung, welche Client-Kategorie ein Entwickler nutzen will, spielt auch der geforderte Grad an Interaktion mit dem User-Interface eine wichtige Rolle. Hinsichtlich der Dimension der Benutzerinteraktion unterscheidet man zwischen einer einfachen Navigation über Bildschirmseiten sowie interaktiven und komplexen Vorgängen. Um die Art der Benutzerinteraktion im Spektrum „ein-



**Clients lassen sich in unterschiedliche technische Kategorien einteilen (Abb. 4).**



**Ein RIA-Client mit Ajax konnte vor HTML5 nicht alle Schichten einer Webanwendung abdecken (Abb. 5).**



**Sowohl der Technik als auch dem Aufwand sind bei RIAs Grenzen gesetzt (Abb. 6).**

fach“ bis „komplex/hoch“ (siehe Abbildung 3) einordnen zu können, sollte man verschiedene Fragen stellen: Sind Dialoge erforderlich, die beispielsweise einen Kunden einem Produkt zuordnen? Gehört das schnelle Arbeiten über Tastatur-Navigation zu den Anforderungen? Benötigt der Anwender Drag & Drop, um zum Beispiel einen Datensatz aus der Datenbank von einer Applikation zur anderen zu schieben? Jede mit „Ja“ beantwortete Frage erhöht die Reichhaltigkeit der Interaktion und verlangt den Einsatz einer unterstützenden Technik. Nicht alle Client-Kategorien unterstützen die Interaktion gleichermaßen. Abbildung 4 verdeutlicht dies im Kontext der Benutzerinteraktion und der Ausführungsumgebung.

## Der Anwender hat seine eigene Sicht

Gemäß Abbildung 4 unterstützen sowohl Rich Clients als auch RIAs ein hohes Maß an Interaktion mit der Benutzerschnittstelle. Die Unterschiede werden deutlich, wenn man die Schichten einer Applikation betrachtet (siehe Abbildung 5).

Ein Rich Client deckt alle Schichten von der Datenhaltung bis zur Oberflächendarstellung ab. Der Server, falls er gebraucht wird, ist in der Regel nur für die gemeinsame Datenhaltung zuständig – eventuell auch für einen Teil der Geschäftslogik. Microsofts Word ist ein Beispiel für einen Rich Client, der keinen Server benötigt. Outlook hingegen ist ein Rich Client mit Server, der für die Aufbewahrung der Mails zuständig ist.

Deckt ein Client mehrere Schichten ab, kann die Kommunikation zwischen ihm und dem Server in tieferen Schichten und weniger häufig erfolgen. Solch ein Client hat ein schnelleres Antwortverhalten, bietet somit eine bessere Benutzerunterstützung und erlaubt so ein flüssigeres Arbeiten.

Im RIA-(Ajax)-Fall ohne HTML5 kann der Client maximal die Schichten bis zur Geschäftslogik abdecken. Gleichzeitig muss immer ein Server vorhanden sein. Er kann Aufgaben bis zur Präsentationslogik übernehmen. Die daraus resultierende längere Antwortzeit beeinträchtigt unter Umständen die Interaktivität. Komplexere User-Interfaces reagieren meistens einen Tick zu langsam, sodass ein flüssiges Arbeiten nicht immer möglich ist. Einen ganzen Tag intensiv mit ei-

nem Ajax-Client zu arbeiten, ist oft anstrengend.

Mit HTML5 lassen sich ebenfalls Ajax-Applikationen realisieren. Abbildung 5 zeigt, dass ein damit realisierter Client standardisiert in tiefere Schichten vordringen kann, ermöglicht unter anderem durch Local Storage, Indexed Database und File-API. Ein RIA-Ajax-Client kann mit HTML5 seine Daten selbst verwalten und benötigt hierfür keinen Server. Das führt wie beim Rich Client zu schnellen Reaktionszeiten. Muss der HTML5-Client aber mit dem Server interagieren, weil er von dort Daten oder Logik bezieht, kann sich das wie bei Ajax ohne HTML5 negativ auf die Interaktivität auswirken.

Abbildung 4 zeigt oben rechts und unten links freie Flächen, die nicht von Client-Kategorien abgedeckt sind. Für RIAs mit Ajax gibt es technische Grenzen, die noch nicht überwindbar sind. Dazu gehören unter anderem:

- **Drag & Drop zwischen Applikationen:** Diese Funktion steht für Text und Bilder zwar zur Verfügung, allerdings haben nicht die Applikationen die Kontrolle darüber. Technisch gar nicht machbar ist zurzeit das Verschieben von Business- oder Domain-Objekten zwischen Programmen (zum Beispiel das Kunden-Objekt von der Applikation A in die Stammdatenhaltung).
- **Reaktionszeiten:** Komplexere, interaktive Applikationen mit vielen Objekten können zu schlechten Reaktionszeiten der Benutzerschnittstelle führen.
- **Automatische Hardware-Interaktionen:** Nicht möglich ist beispielsweise das automatische Inspizieren des Inhalts eines Memory-Sticks oder einer -Karte beim Anschluss.

## Grenzen von Technik und Aufwand

Somit bieten Ajax-Clients einen geringeren Grad an Interaktionsmöglichkeiten als Rich Clients. Dies symbolisiert das Dreieck rechts oben in Abbildung 6. Die gestrichelte Linie darunter symbolisiert die Kosten-Nutzen-Grenze. Das Entwickeln nah an dieser Grenze wird schnell teuer, weil Standards verlassen werden und Unterschiede in Browsern oder Plattformen nicht mehr durchs Framework gekapselt sind. Dies gilt es frühzeitig zu erkennen und durch die Wahl einer passenderen Client-Kategorie zu vermeiden, in die „Pareto-Falle“ zu tappen. Diese 80/20-Regel besagt, dass 80 Prozent der Arbeit in nur

20 Prozent der Gesamtzeit erledigt werden, die verbleibenden 20 Prozent also die meiste Arbeit verursachen.

HTML5 hat dazu beigetragen, die Ajax-Box zu vergrößern und somit die Grenzen zu verschieben, unter anderem mit einer JavaScript-API für natives Drag & Drop. So kann man etwa Dateien aus dem Dateisystem in eine Webapplikation hineinziehen. Auch das Drag & Drop selbst definierter Typen ist möglich. Nichtsdestotrotz bleibt sowohl die technische als auch die Kosten-Nutzen-Grenze bestehen.

Auch bei wenig interaktiven Stand-alone-Clients stellt sich die Kosten-Nutzen-Frage. Im Bereich unten links (Abbildung 6) ist es oft effektiver, sich für einen Web-Client zu entscheiden.

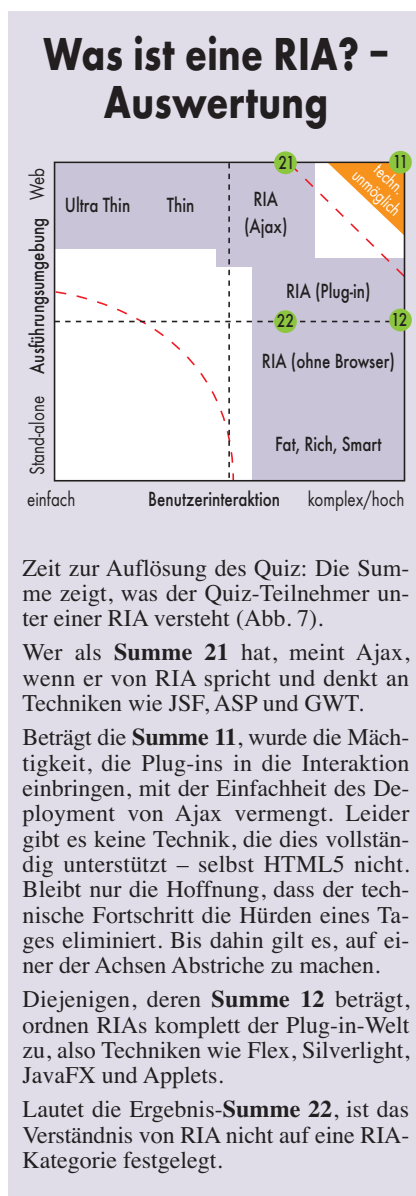
Die Entwicklung mobiler Applikationen (Apps) ist ein Geschäftsfeld, in dem Rich Clients für wenig interaktive Arbeitsweisen erstellt werden. Die Optimierung von Webseiten für jedes Handy kann sich als ebenso umfangreich erweisen wie eine Applikation „nativ“ für die Plattform – beispielsweise iPhone oder Android – zu schreiben. Für die native Stand-alone-Variante spricht zudem, dass die im Trend liegenden App-Stores respektive Market-Places es erleichtern, Applikationen zu finden. Der Benutzer weiß, dass eine Applikation aus dem Store zum Telefon passt. Bei einer Webseite kann er nicht davon ausgehen, dass sie für sein Telefon optimiert wurde.

Wer jetzt sein eigenes Verständnis, was eine RIA ausmacht, mit dem vergleichen möchte, das er am Anfang dieses Artikels hatte, findet die Antwort in der Auswertung.

## Konkrete Veränderungen durch HTML5

Die Abbildungen 5 und 6 haben gezeigt, dass HTML5 neue Möglichkeiten eröffnet. Obwohl die Spezifikation noch nicht fertig ist, unterstützen moderne Browser Teile davon bereits jetzt. Jede neue Browser-Generation bietet mehr Support für das aktuelle HTML5.

Einige Funktionen, die HTML5 ins Rampenlicht rückt, ließen sich bereits früher mit JavaScript-Bibliotheken realisieren. Jedoch musste der Entwickler für jedes Feature eine gesonderte Library verwenden. Diese Bibliotheken funktionierten jedoch oft nur mit einzelnen Browsern in bestimmten Versionen. Der Vorteil von HTML5 als Standard ist, dass die neuen Browser

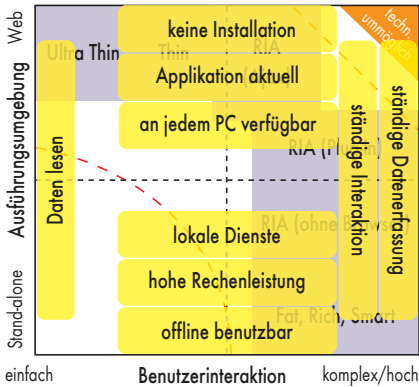


wettbewerbsbedingt früh und möglichst viel seiner Funktionen unterstützen.

Konkret haben Neuerungen rund um HTML5 eine Annäherung von Ajax an „Stand-alone“-Clients gebracht, also die RIA(Ajax)-Box aus Abbildung 6 nach unten vergrößert. Dazu zählen

– **Offline-Möglichkeiten:** Durch den Application Cache in HTML5 kann der Autor einer Webseite respektive Applikation bestimmen, welche Teile auch offline verfügbar sein sollen. Diese werden auf dem Gerät gespeichert und dann verwendet, wenn keine Onlineverbindung vorhanden ist.

– **Local Storage und Indexed Database:** Wie erwähnt, wurde mit HTML5 der Local Storage eingeführt. Somit kann nun ein RIA-Ajax-Client mit der Hypertext Markup Language bis zu einer gewissen Menge seine Daten selbst verwalten und benötigt hierfür keinen



### Alle Kriterien für die Auswahl der richtigen Client-Technik auf einen Blick (Abb. 8)

Server. Für größere Datenmenge gibt es eine lokale Datenbank.

– **File-API:** HTML5 hat eine mächtige API zum Lesen und Schreiben von Dateien eingeführt. Unter anderem lassen sich Daten auf einfache Weise asynchron hochladen oder in ein temporäres Verzeichnis kopieren.

– **Geolocation:** Mit Geolocation kann ein Anwender die Position seines Mobiltelefons (sofern verfügbar und erlaubt) abfragen. Bei PCs handelt es sich dabei meist um die Position des Zugangspunktes des Internet-Providers.

– **Device Elements:** HTML5 erlaubt den Zugriff auf Hardware-Elemente des Geräts wie Kamera oder GPS-Sensor. Allerdings sind nicht alle Sensoren standardisiert ansprechbar. Beispielsweise hat HTML5 keine Kompass-API. Die in der Entwurfsphase befindliche Spezifikation „DeviceOrientation Event Specification“ (siehe „Onlinequellen“ [a] und iX-Link) wird es erlauben, herauszufinden, wie das Mobiltelefon oder der PC orientiert ist und welche Beschleunigungskräfte darauf einwirken.

– **Web Messaging:** Channel Messaging erlaubt die Kommunikation zwischen Browsern.

Weitere Neuerungen verbessern zudem die Interaktion und vergrößern damit die RIA(-Ajax)-Box aus Abbildung 6 nach rechts.

Das Einführen neuer Typen für das Element *input* etwa vereinfacht die Eingabe von Daten. Smartphones können

darauf reagieren und die eingeblendete Tastatur optimieren (bei Zahlen nur Zahlen, bei E-Mail das @-Zeichen an einer schnell auffindbaren Stelle und so weiter). Darüber hinaus gibt es neue Typen für Datum, Zeit und Farben. Somit ist es für den Benutzer einfacher, Daten einzugeben beziehungsweise auszuwählen. Prinzipiell ging das zwar schon vor HTML5, aber der Webentwickler musste entweder eine komfortable Eingabemöglichkeit selbst programmieren oder auf eine Bibliothek zurückgreifen.

Mit HTML5 erhält das Element *input* das Attribut *Pattern*. Diese Regular Expression dient zur Validierung der Benutzereingabe.

Drag & Drop ist in HTML5 eingebaut und nun ohne Bibliotheken verwendbar. Der Entwickler kann auch eigene Typen definieren.

Canvas und WebGL erlauben über Scripts ein einfaches Rendering von Szenen, sodass man Teile der Benutzerschnittstelle selbst professionell gestalten kann. Die nötige Performance wird über das Ansprechen der lokalen Grafikkarte erreicht.

Web-Sockets ermöglichen unter anderem die Kommunikation zwischen Server und Client und zwar ohne aufwendige Workarounds wie „Long Polling“ oder den Einsatz von Bibliotheken. Gibt es auf dem Server ein Ereignis, kann er das dem Client sofort mitteilen. Reverse-Ajax respektive Server-Push ist somit standardisiert und ohne Kniffe gelöst. Über Web-Sockets lässt sich ähnlich kommunizieren wie über Sockets zwischen Rich Clients und Server.

## HTML5 ist mehr als ein Hype

HTML5 verschiebt also die Grenzen von RIAs mit Ajax. Somit lassen sich mit den hinter der Spezifikation stehenden Konzepten Aufgaben relativ einfach lösen, die vorher nicht oder nur mit Aufwand und unter Einbeziehung proprietärer Software realisierbar waren. Die Grenzen von RIAs mit Ajax kann HTML5 aber (noch) nicht beseitigen. Beispielsweise lassen sich nicht

alle Sensoren eines Mobiltelefons standardisiert ansprechen.

Laut Gartner Hype Cycle [b] wecken neue Techniken zuerst eine übertriebene Erwartungshaltung, gefolgt von Desillusionierung (Tal der Tränen). Die Technik gewinnt wieder an Boden, wenn Realismus die Oberhand gewinnt, oder sie verschwindet, weil sie doch keine wirkliche Innovation darstellt oder ihre Nachteile überwiegen. HTML5 wird nicht im Tal der Tränen verbleiben, da das Verschieben der Grenzen neue Möglichkeiten eröffnet hat.

Weder Schichtenarchitektur noch aktuelle Trends sollten die Auswahl der Client-Technik bestimmen. Organisatorische und projektbedingte Rahmenbedingungen bestimmen die vertikale und das Interaktionsverhalten die horizontale Achse der Matrix aus Abbildung 6. Abbildung 8 zeigt entscheidende Rahmenbedingungen und Anwendungsfälle ohne Anspruch auf Vollständigkeit.

Treffen obige Rahmenbedingungen zu, sollte die Entscheidung Richtung Web-Client gehen. Gelten die unteren Anforderungen, ist eine Stand-alone-Applikation die richtige Wahl. Sollte der linke Anwendungsfall auf die Applikation zutreffen, bietet sich ein einfacher Client an, und aufgrund der Kostengrenze wird es ein Thin Client sein. Die Anwendungsfälle auf der rechten Seite verlangen eine Applikation mit hohem Anteil von Benutzerinteraktion. Aufgrund der Machbarkeits- und Kostengrenze wird man sich in dem Fall eher für einen Fat, Smart oder Rich Client entscheiden oder allenfalls für eine RIA ohne Browser.

Allerdings gehören konkurrierende und widersprüchliche Anforderungen und Rahmenbedingungen zum Projektalltag. So auch in diesem Fall. Zum Beispiel kann der Anwendungsfall „ständige Datenerfassung“ zu den Anforderungen gehören und gleichzeitig „keine Installation“ von höheren Instanzen im Unternehmen als Rahmenbedingung vorgegeben sein. „Keine Installation“ und „Offline-Fähigkeit“ werden ebenfalls häufig gleichzeitig gefordert.

Drei Maßnahmen können solche Konflikte lösen:

- priorisieren (respektive bei einzelnen Anforderungen Abstriche machen),
- Mehrkanalfähigkeit anstreben, also mehrere Clients entwickeln und
- hohen Aufwand investieren, um alle Anforderungen in einem Client zu erfüllen.

Wie sich solche Konflikte lösen lassen, zeigen im Folgenden drei Beispiele.

### Onlinequellen

- |   |  |
|---|--|
| [a] DeviceOrientation Event Specification | <a href="http://dev.w3.org/geo/api/spec-source-orientation">dev.w3.org/geo/api/spec-source-orientation</a>   |
| [b] Gartner Hype Cycle                    | <a href="http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp">www.gartner.com/technology/research/methodologies/hype-cycle.jsp</a> |
| [c] Stopping the Gears                    | <a href="http://gearsblog.blogspot.com/2011/03/stopping-gears.html">gearsblog.blogspot.com/2011/03/stopping-gears.html</a>                             |

Eine sehr spezielle Anforderungskombination ist „Daten lesen“ und „ständige Interaktion“. Wahrscheinlich stehen hier zwei unterschiedliche Arten von Benutzertypen hinter den Anforderungen (siehe Flückiger und Richter zum Personas-Modell [1]). Beispielsweise Sachbearbeiter (Eingabe) und Manager (Reports). An dieser Stelle ist die Überlegung angebracht, zwei Applikationen zu entwickeln. Eine für die Reports und eine für die Erfassung. Die Konfliktlösungsstrategie „mehrere Clients schreiben“ ist hier aus Usability-Gründen am naheliegendsten.

Will man den Widerspruch zwischen „keine Installation“ und „hohe Datenerfassung“ auflösen, sollte man priorisieren, also eine der Anforderungen abschwächen oder fallen lassen. Gewinnt die hohe Datenerfassung, kann vielleicht eine RIA-Technik, die näher an Web ist, die Installationsproblematik lösen. Ist „keine Installation“ ein Killerkriterium, muss man besonderes Augenmerk auf die Benutzerschnittstelle legen. Unzufriedene Anwender nutzen eine Applikation nicht – selbst dann nicht, wenn sie nichts installieren müssen. Zudem sinkt die Motivation, und die Fehleranfälligkeit steigt, wenn Mitarbeiter schneller arbeiten könnten, als es die Applikation zulässt.

Mehrere Clients sind – sofern es die Finanzen erlauben – ebenfalls eine gute Wahl. Was sich auf den ersten Blick verschwenderisch anhört, kann sich als Wettbewerbsvorteil herausstellen. Als Anwendungsbeispiel sei die Fotobestellung über das Internet genannt. Oft gibt es einen rudimentären Web-Client, über den Anwender Fotos hochladen und bestellen können. Daneben steht häufig ein RIA-Client zur Verfügung, der Massen-Uploads und Bildmanipulationen erlaubt. Einige Anbieter haben darüber hinaus einen Rich Client für die lokale Bildbearbeitung, mit dem der Benutzer ein Foto beispielsweise zuschneiden und die roten Augen entfernen kann, ohne dass er den Server des Labors bemühen muss. Erst am Ende dieser Vorarbeiten werden die Bilder hochgeladen. Dies kann für einige Kunden angenehmer sein, als die Bilder zuerst hochzuladen und dann zu bearbeiten. Die Foto-Labors wenden sich mit ihren Clients an unterschiedliche Benutzergruppen: von Personen, die nur schnell die Fotos hochladen und bestellen wollen, bis zu Stammkunden, die eine lokale Applikation installiert haben, alles vorbereiten und erst danach alles hochladen.

E-Mail ist hierfür ein weiteres Beispiel. Nur einen Rich Client zu haben genügt nicht. Web-Clients sind Pflicht.

Ein weiterer Widerspruch ist der zwischen den häufig geäußerten Anforderungen „keine Installation“ und „Offline-Fähigkeit“. Das Bedürfnis nach Offline-Fähigkeit wächst seit Jahren immer mehr. So gab es früh Lösungsansätze wie Googles Gears. Das Projekt wird aber nicht mehr weiterentwickelt und soll Ende 2012 eingestellt werden [c]. Die Anstrengungen des Gears-Teams sind in HTML5 eingeflossen.

## Zusammenfassung

Komplexe, interaktive Web-Clients haben technische Grenzen. HTML5 verschiebt diese Grenzen und hat somit das Potenzial, sich über einen Hype hinaus zu etablieren. Als Voraussetzung für seinen breiten Einsatz müssen sich die HTML5-fähigen Browser etablieren.

HTML wird jedoch auch dann Techniken aus anderen Client-Kategorien nicht verdrängen. Um die richtige Kategorie zu finden, muss man weiterhin die Anforderungen analysieren. Entscheidend sind Rahmenbedingungen wie „keine Installation“ und „Offline-Fähigkeit“ sowie das Interaktionsverhalten des Anwenders, der „Daten lesen“ und „ständige Datenerfassung“ betreiben soll. Ignoriert man die Anforderungen aus Rahmenbedingungen und Interaktionsverhalten der Benutzer zugunsten einer bevorzugten Technik, muss man mit höheren Entwicklungskosten oder Akzeptanzproblemen rechnen.

Mit HTML5 sind reichere RIAs möglich geworden. Ist aber eine hohe Interaktion oder automatische Hardware-Interaktion gefordert, sind Rich Clients „reicher“.

(ka)

### NIKOLAOS KAINANTZIS

arbeitet als Softwarearchitekt bei Zühlke Engineering und leitet dort im Competence Center Client Technology die Java-Gruppe.

### Literatur

- [1] Markus D. Flückiger, Michael Richter; Usability Engineering kompakt: Benutzbare Software gezielt entwickeln; Spektrum Akademischer Verlag, 2007

Alle Links: [www.ix.de/ix1112086](http://www.ix.de/ix1112086)

