Prototyping von Rich Internet Applikationen

Master of Advanced Studies in Human Computer Interaction Design

Masterarbeit

Autoren:

Andreas Binggeli
Marc Blume
Yuan-Yuan Sun

Betreuer:

Thomas Bircher

30. Januar 2009

UNI BASEL

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

interactions that work stimmt.

Erklärung zur Urheberschaft:

Wir erklären hiermit an Eides statt, dass wir die vorliegende Arbeit ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt haben; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum Andreas Binggeli

Ort, Datum Marc Blume

Ort, Datum Yuan-Yuan Sun

Abstract

Rich Internet Applikationen (RIA) beschreiben eine neue Generation von Web-Anwendungen, welche die Vorteile von traditionellen Web- und Desktop-Applikationen miteinander vereinen. RIA stellen die Designer vor grosse Herausforderungen, da die neuen Konzepte und Interaktionsmöglichkeiten bei den Anwendern teilweise noch unbekannt sind, und die Dynamik und Reichhaltigkeit von RIA grössere Anforderungen an das Erstellen und Testen von Prototypen der Anwendungen stellt.

Das Ziel dieser Studie war es, verschiedene Prototyping Methoden auf Ihre Eignung zur Simulation und Evaluation von RIA Patterns zu untersuchen. Für die Untersuchung wurden zehn typische RIA Interaktionsmuster ausgewählt und mit vier verschiedenen Prototyp Versionen umgesetzt: handgezeichnete Papier-Prototypen, mit PowerPoint erstellten Papier-Prototypen, mit der Prototyping Software „Axure RP Pro“ erstellte digitalen Prototypen und ausprogrammierten Ajax Prototypen. Alle wurden in zwei Iterationen mit insgesamt 15 Benutzern getestet.

Die quantitative Auswertung der Tests erfolgte auf der Basis eines „subjektiven“ Fragebogens welcher die Testteilnehmer nach jeder Testaufgabe ausfüllten sowie einem „objektiven“ Fragebogen mit unseren eigenen Einschätzungen über die Schwierigkeiten bei der Lösung der Aufgaben. Die Ergebnisse der quantitativen Auswertung wurden anhand von neun definierten Hypothesen überprüft.

Alleine die langsamere Reaktionszeit der Papier-Prototypen gegenüber den beiden digitalen Prototypen war statistisch bedeutsam. Ansonsten wurde von den Testteilnehmern gemäss der Daten der Fragebogen sowohl das Verhalten wie auch erstaunlicherweise das Aussehen der vier Prototyp-Versionen als gleichwertig empfunden.

Die qualitative Beobachtung und Befragung in der Testsituation zeichnete hingegen ein anderes Bild. Einige der simulierten RIA Patterns konnten aufgrund Beschränkungen bestimmter Prototyping Methoden nur ungenügend simuliert werden und führen bei den Benutzertests mitunter zu erheblichen Schwierigkeiten.

Die Diskussion der Untersuchungsergebnisse führte zu dem Schluss, dass alle Methoden ihren Platz in einem benutzerorientierten Gestaltungsprozess haben, je nach Ausgangslage und Zielsetzung aber nur mit Vorbehalten für das Testen von RIA Pattern Prototypen geeignet sind.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Management Summary

Ausgangslage und Zielsetzung

Rich Internet Applikationen (RIA) beschreiben eine neue Generation von Web-Anwendungen, welche gegenüber traditionellen Web-Anwendungen erhöhte Dynamik und Benutzerkomfort versprechen. Beim Design und der Entwicklung von RIA muss dem Umstand Rechnung getragen werden, dass die neuen Konzepte und Interaktionsmuster teilweise noch unbekannt oder nicht standardisiert sind und deshalb dem Prototyping spezielle Beachtung geschenkt werden muss.

Die folgende Studie beschäftigt sich mit der Fragestellung, wie gut sich verschiedene Prototyping Methoden für die Simulation von Rich Internet Applikationen eignen. Die theoretischen Grundlagen werden aufgezeigt, die Potenziale von vier ausgewählten Methoden in einer empirischen Untersuchung miteinander verglichen und Empfehlungen für den Einsatz und die Optimierung der Techniken gegeben.

Rich Internet Applikationen

Web-basierte Applikationen gewinnen seit der Einführung des World Wide Webs immer mehr an Bedeutung und Popularität. Mittlerweile ist das Web die beherrschende Technologie für interne, Business-to-Business (B2B) und Business-to-Consumer (B2C) Applikationen. Auch wenn die Hypertext Markup Language (HTML) und das Hypertext Transfer Protocol (HTTP) durch ihre Standardisierung eine enorme Verfügbarkeit und Reichweite erlangen, sind die Möglichkeiten zur Gestaltung der Benutzeroberflächen im Vergleich zu Desktop Applikationen sehr primitiv. Desktop Applikationen weisen andererseits Probleme bei der Software-Verteilung oder der Interoperabilität auf. Rich Internet Applikationen sind ein ausgezeichneter Kompromiss, indem sie die Reichweite von HTML und HTTP mit der Benutzerkomfort von Desktop Anwendungen vereinen. Rich Internet Applikationen zeichnen sich dadurch aus, dass sie Teile der Darstellung dynamisch aktualisieren können, sofort auf Benutzereingaben reagieren und Daten in Echtzeit zwischen dem Client und dem Server austauschen können.

RIA Interaktions-Design Patterns

Interaktions-Design Patterns beschreiben generell anwendbare Lösungen auf ein bestimmtes Problem im Bereich der Gestaltung von Benutzeroberflächen. Für den Kontext der Fragestellung, wie gut sich verschiedene Prototyping Methoden für die Simulation von RIA eignen, war es wichtig Kriterien und Beispiele von RIA Interaktions-Design Patterns zu analysieren. Direkte Manipulation von Objekten, sofortige Reaktion auf Benutzereingaben und unmittelbares Feedback, Aktualisierung von Bereichen des Bildschirms und die Ausführung von Aktionen im Kontext eines Objektes sind Beispiele von Attributen die RIA-Patterns auszeichnen. Mittlerweile beschreiben Bücher und Web Sites Interaktionsmuster, die diesen Kriterien entsprechen. Im Rahmen dieser Arbeit ist eine eigene Sammlung von Patterns entstanden, von denen zehn ausgewählt und mit Prototypen umgesetzt wurden.

Prototyping

Prototyping ist ein fester Bestandteil in verschiedenen Phasen des benutzerorientierten Gestaltungsprozess (User-centered Design). Prototypen werden erstellt, um interaktive Anwendungen oder Teile davon auf vereinfachte Weise zu repräsentieren, bevor das fertige System gebaut wird. In einem iterativen Verfahren können die Prototypen laufend verfeinert und mit Benutzern getestet werden. Prototyping Methoden mit unterschiedlicher Wiedergabetreue, von handgezeichneten Skizzen bis zu ausprogrammierten Artefakten, können in den verschiedenen Phasen des Prozesses angewendet werden. Werkzeuge wie Papier und Bleistift, Diagramm- und Graphik-programme, spezialisierte Prototyping

Software oder HTML Editoren unterstützen die Umsetzung der Prototypen. Evaluationsmethoden wie kognitive Walkthroughs und Usability Tests dienen dazu, die Zwischenergebnisse mit Benutzern zu validieren, damit die daraus gewonnenen Erkenntnisse in die nächste Phase der Entwicklung einfliessen können.

Konzeption und Durchführung der Untersuchung

Aus der Menge der gesammelten RIA Interaktions-Design Patterns haben wir exemplarisch zehn Patterns aus verschiedenen Kategorien ausgewählt:

- Schieberegler (Slider)
- Editierbare Tabellen
- Rich Text Editor
- Eingabevorschläge (Type & Suggest)
- Pulldown Menüs
- Kontext Menüs
- Kartenausschnitte verschieben
- Drag und Drop
- Sofortige Filterung von Suchergebnissen
- Validierung von Eingaben

Diese Patterns wurden mit vier verschiedenen Prototyping Methoden umgesetzt. Wir entschieden uns für handgezeichnete und mit PowerPoint erstellte Papier-Prototypen, digitalen Prototypen, die mit der Software Axure RP Pro erstellt wurden und ausprogrammierte Prototypen auf der Basis von Ajax Technologien.

Die Prototypen wurden in zwei Iterationen mit insgesamt 15 Testteilnehmern mit Hilfe von Usability Walkthroughs getestet. Die Testteilnehmer mussten aus der Menge der 40 Prototyp-Versionen acht Aufgaben lösen, wobei jedes Pattern von einem Benutzer nur einmal getestet wurde. Die Testteilnehmer wurden nach jedem Test aufgefordert, einen kurzen Fragebogen auszufüllen, bei dem die subjektive Wahrnehmung über Aussehen, Verhalten und Schnelligkeit des Prototyps bewertet werden musste. Wir bewerteten unsererseits in einem „objektiven“ Fragebogen das Verhalten der Testteilnehmer bei der Bearbeitung der Aufgaben. Neun Hypothesen leiteten die quantitative Auswertung der Messwerte.

Analyse der Feststellungen

Die Auswertung der direkten Messwerte lieferte nur wenige statistisch bedeutsame Unterschiede bei den verschiedenen Prototyping Methoden. Die Testteilnehmer kamen bei der Bearbeitung der Testaufgaben ähnlich gut zurecht, unabhängig davon ob sie die Aufgabe mit dem Papier-, PowerPoint-, Axure- oder Ajax-basierten Prototypen bearbeiteten. Auch von den Testteilnehmern selbst wurden die verschiedenen Prototyp-Versionen als gleichartig in Bezug auf das Verhalten und Aussehen bewertet. Einzig die langsamere Reaktionszeit der beiden auf Papier getesteten Prototypen im Vergleich zu den beiden digitalen Prototypen war statistisch signifikant.

Die Auswertung der qualitativen Beobachtung und Befragung zeichnete hingegen ein anderes Bild. Bei den papierbasierten Prototypen stellte sich beispielsweise die Geschwindigkeit des Operators bei komplexen Operationen und die fehlenden Rückmeldungen über den Systemstatus als Problem heraus. Bei den mit Axure RP Pro erstellten Prototypen bestand das Problem darin, dass sie erweiterte Mausinteraktionen wie z.B. ein Rechtsklick sowie das Gedrückthalten und Ziehen der Maustaste nicht unterstützten. Die implementierten Umgehungslösungen verhielten sich realitätsfremd und stellten die Testteilnehmer teilweise vor unüberwindbare Probleme. Die ausprogrammierten Prototypen konnten bis auf eine Ausnahme in der ersten Iteration problemlos bedient werden.

Diskussion der Testergebnisse

Für den Vergleich, wie gut sich die vier Prototyping Techniken für die Simulation von RIA eignen, wurden mehrere Kriterien wie Entwicklungsaufwand, Wartungsaufwand, Wiedergabetreue und Interaktivität der Prototypen bewertet.

Handgezeichnete und mit PowerPoint erstellte Papier-Prototypen haben ihre grossen Vorteile in der einfachen Erstellung und ihrer hohen Interaktivität. Für die Interaktivität benötigen sie immer einen Operator und sie muss nötigenfalls erläutert werden. Durch die tiefe visuelle Detailtreue eignen sich Papier-Prototypen weniger für die Spezifikation von Anforderungen.

Die Erstellung der Prototypen mit Axure RP Pro war nach einer kurzen Einarbeitungszeit relativ problemlos. Zwei der zehn Prototypen bereiteten allerdings unlösbare Schwierigkeiten, da notwendige Mausereignisse vom Tool nicht unterstützt wurden. Die entstandenen Umgehungslösungen fielen bei den Benutzertests durch, da die Interaktionsmuster nicht dem mentalen Modell der Benutzer entsprach.

Die Erstellung der ausprogrammierten Prototypen erforderte Kenntnisse in verschiedenen Technologien wie HTML, JavaScript, CSS, DOM und Ajax. Der Erstellungsauwand war verglichen mit den anderen Methoden höher, dafür gab es in Bezug auf die Wiedergabetreue und die Interaktivität keine Einschränkungen.

Möglichkeiten der Erweiterung des Einsatzspektrums der Prototyping Methoden

Die Optimierung von Papier-Prototypen setzt vor allem bei der optimalen Vorbereitung an. Je schneller auf die Interaktionen des Benutzers reagiert und der Zustand der Seite hergestellt werden kann, desto realistischer kann die Benutzererfahrung simuliert werden. RIA typische Feinheiten, wie das Unterstützen der rechten Maustaste oder die Darstellung von Systemfeedback, können durch bestimmte Techniken unterstützt werden.

Die Problematik, dass Axure RP Pro in der aktuellen Version nicht alle Maus- und Tastaturereignisse unterstützt, kann nur gelöst werden, inddem der generierte Prototyp mit anderen Techniken kombiniert wird. Axure bietet dazu ein Integrations-Widget an, mit der eine beliebige Web-Seite integriert werden kann. Die Umsetzung einer Drag und Drop Lösung könnte so beispielsweise mit Flash umgesetzt und in den Axure Prototypen als Komponente integriert werden.

Bei den ausprogrammierten Prototpyen stellt sich weniger die Frage, wie die Methodik in Bezug auf Wiedergabetreue und Interaktivität verbessert werden kann, sondern eher wie die Erstellung der Prototypen vereinfacht und beschleunigt werden kann. Hilfreich ist beispielsweise der Einsatz von Tools wie visuellen RIA Entwicklungsumgebungen, Code Generatoren oder Meta Sprachen. Auch durch Zusammenfügen und Anpassen von Code-Beispielen aus dem Internet kann der Lern- und Entwicklungswand bedeutend reduziert werden.

Fazit

Alle vier getesteten Prototyping Methoden eignen sich in unterschiedlich hohem Masse, RIA Patterns in Prototypen zu simulieren. Papier-Prototypen sind aufgrund ihrer Einfachheit ideal, um erste Design Entwürfe zu evaluieren und vergleichen, während ausprogrammierte Prototypen in späteren Projektphasen unerlässlich sind, um alle Feinheiten einer RIA zu überprüfen. Die verwendete Technologie sollte dabei idealerweise der Zielarchitektur entsprechen, damit Verhalten und Aussehen des Prototyps so realitätsnah wie möglich umgesetzt werden können. Der Einsatz eines spezialisierten Prototyping Tools wie Axure RP Pro ist aufgrund der technischen Limitierungen nur mit Vorbehalten für die Simulation von RIA Patterns zu empfehlen.

1 Einleitung

Rich Internet-Technologien wie Ajax oder Flex erlauben eine Fülle von neuen Interaktionsmöglichkeiten in Web-Anwendungen. Dynamische Elemente und die Möglichkeit, einzelne Bildschirmelemente auszutauschen, ohne die gesamte Seite neu zu laden, erweitern zwar die Möglichkeiten des Interaktionsdesigns, gleichzeitig brechen sie aber mit gewohnten Verhaltensweisen heutiger Nutzer.

Unsere Arbeit untersucht in diesem Spannungsfeld, wie sich RIA Patterns mit verschiedenen Prototyping-Methoden simulieren lassen.

Wir gehen dabei von folgenden Fragestellungen aus:

- Was sind zentrale Merkmale von RIA Interaktions-Design Patterns?
- Wie unterscheiden sich RIA-basierte Interaktionsformen von denen klassischer Web- bzw. Desktop Anwendungen?
- Welche RIA Interaktions-Design Patterns gibt es derzeit?
- Wie können diese RIA Interaktions-Design Patterns klassifiziert werden?
- Wie gut lassen sich verschiedene RIA-Interaktionsarten mit verschiedenen Prototyping Methoden simulieren und in Benutzertests evaluieren?
- Welche Werkzeuge (Tools) eignen sich zur Unterstützung des Prototyping von RIA Patterns? Dabei wird auch ein besonderes Augenmerk auf der Effizienz der verschiedenen Werkzeuge liegen.
- Wie lassen sich die Möglichkeiten verschiedener Prototyping Methoden und Werkzeuge ausweiten, um RIA-basierte Interaktionen realitätsnah darzustellen?

Im ersten Teil dieser Arbeit werden die theoretischen Grundlagen zu Rich Internet Applikationen, RIA Interaktions-Design Patterns, Prototyping und Prototyping Tests geschaffen. Danach beschreiben wir, welche Forschungsfragen wir uns stellten, welchen Untersuchungsgegenstand wir gewählt haben, wie die Prototypen erstellt wurden und mit welchem Untersuchungsdesign wir die Antworten suchten. Im folgenden Kapitel erfolgt die Auswertung der quantitativen Messwerte der objektiven und subjektiven Fragebogen, die im Rahmen der Benutzertests zum Einsatz kamen. In der anschliessenden Diskussion werden die einzelnen Prototyping Methoden bewertet sowie Einsatz- und Optimierungsmöglichkeiten beschrieben. Das Fazit hinterfragt Vorgehensweise und Ergebnisse der Untersuchung und beschreibt die wichtigsten Erkenntnisse der Studie.

Die vorliegende Arbeit wurde im Rahmen des Masterstudiums „Master of Advanced Studies in Human Computer Interaction Design“ der Universität Basel und der Fachhochschule Rapperswil in Zusammenarbeit mit der Firma Stimmt AG aus Zürich erstellt.

2 Theoretischer Hintergrund: Prototyping von Rich Internet Applikationen

Im ersten Teil dieses Kapitels führen wir Rich Internet Applikationen ein. Mit Interaktions-Design Patterns klassifizieren wir im zweiten Teil, welche standardisierten Lösungsmuster sich bisher herausgebildet haben. Die Erstellung und Verwendung von Prototypen besprechen wir im dritten Unterkapitel. Im vierten Teil wird schliesslich der Einsatz von Prototypen im Rahmen von Prototyping Tests zur Evaluation von Lösungsansätzen besprochen.

2.1 Rich Internet Applikationen

Rich Internet Applikationen (RIA) beschreiben eine neue Generation von Anwendungen, welche die Vorteile von traditionellen Web- und Desktop-Applikationen miteinander vereinen. RIA Technologien sind auch die Wegbereiter für neue Konzepte wie „Web 2.0“ oder „Software as a Service“ (SaaS). RIA stellen die Designer vor grosse Herausforderungen, da die neuen Konzepte und Interaktionsmöglichkeiten noch teilweise unbekannt oder noch nicht standardisiert sind. Damit das Potential dieser neuen Technologie voll ausgeschöpft werden kann, muss dem Prototyping und Testing besondere Beachtung geschenkt werden.

2.1.1 Entwicklung von Rich Internet Applikationen

In den Neunziger Jahren wurde mit der Verfügbarkeit der Hypertext Markup Language (HTML) und dem Hypertext Transfer Protocol (HTTP) der Grundstein für den Bau von interoperablen und verteilten Anwendungen über das Internet gelegt. HTML ermöglichte auf eine einfache und effiziente Art die Zusammenstellung von Text und Bildern in einer statischen Web Seite, die mit einem Web Browser dargestellt werden konnte. HTTP bot ein einfaches und zustandsloses Protokoll zur Übertragung der Inhalte. Web Seiten wurden komplett auf einem Server aufbereitet und zum Browser zur Darstellung geschickt, ohne dass dabei Sitzungsinformationen ausgetauscht wurden. Mit HTML Forms war eine einfache Benutzerinteraktion zwischen dem Client und dem Server möglich. HTML öffnete auf der einen Seite das Web für alle, schränkte aber auf der anderen Seite die Interaktionsmöglichkeiten enorm ein. Für Benutzer, die den Komfort von Client/Server-Anwendungen gewohnt waren, war HTML beinahe ein Schritt zurück in die Zeiten der 3270-Terminalanwendungen und bescheidener User-Experience.

Mit zunehmender Popularität des World Wide Web wurden auch die Anforderungen an die Interaktivität der Anwendungen immer grösser. Die Möglichkeiten waren aber durch verschiedene Faktoren limitiert. Da die Daten und die Präsentation auf einem Server aufbereitet und in einem HTML Dokument an den Client geschickt werden mussten, fand die gesamte Verarbeitungslogik auf dem Server statt. So mussten selbst einfachste Aufgaben, wie das Bearbeiten von Listen, sehr umständlich umgesetzt werden. Mit neuen Spezifikationen und Standards wurde versucht, diese Lücke zwischen Desktop- und Web-Anwendungen zu schliessen. Mit Technologien wie Client Side Scripting, Dynamic HTML (DHTML) oder Cascading Style Sheets (CSS) war es möglich, gewisse Aspekte der Verarbeitung und der Präsentation auf den Client zu verlagern. Leider führten aber diese Vorstösse auch zu einer erhöhten Entwicklungskomplexität und zu höheren Testaufwänden. Die Entwickler waren mit verschiedenen Schwierigkeiten konfrontiert. Proprietäre Erweiterungen der verschiedenen Browser und Unterschiede in der Aufbereitung der Dokumente führten dazu, dass die Web-Anwendungen nur so gut sein konnten wie der kleinste gemeinsame Nenner aller verfügbaren Browser.

Bei Client/Server-Anwendungen kommuniziert ein Rich Client mit einem oder mehreren Servern in einem Netzwerk. Rich Clients bieten in Bezug auf das User Interface alle Möglichkeiten die man von Desktop Anwendungen gewohnt ist, wie beispielsweise Drag und Drop, hochauflösende Graphiken und ein User Interface, das unverzüglich auf Eingaben des Benutzers reagiert und veränderte Daten sofort anzeigt. Der Client übernimmt dabei ein Grossteil der Verarbeitung auf der lokalen CPU und kommuniziert nur mit dem Server, um Daten auszutauschen oder einen Dienst auf einem der Server aufzurufen. Beispiele sind Microsoft Outlook oder IBM Lotus Notes.

Die Nachteile von Client/Server-Anwendungen liegen im Bereich der Software-Verteilung (Deployment) und des Versionsmanagements von Client Komponenten. So muss in der Regel bei einer neuen Version der Server-Anwendung auch eine neue Client Software installiert werden. Die Client-Anwendungen werden meistens auch nur für ausgewählte Betriebssysteme angeboten, was die Reichweite der Anwendung einschränkt. Der Einsatz von Client/Server-Anwendungen in einem offenen Netzwerk wie dem Internet gestaltet sich darum als sehr schwierig.

Die Probleme von Web- und Client/Server-Anwendungen sind unterschiedlicher Natur, trotzdem aber eng miteinander verwandt. Die Stärken einer Technologie sind gleichzeitig die Schwächen der anderen. Es liegt also nahe, die Vorteile der Web-Anwendungen (einfache Verteilung, keine erweiterten Anforderungen an die Client-Software) mit den Vorteilen der Client/Server-Anwendungen (reichhaltiges UI, Ausnutzung der Client Hardware, zustandsbehaftetes Protokoll) zu vereinen und gleichzeitig die Nachteile zu eliminieren.

Rich Internet Applikationen (RIA) versprechen, diese Lücke zu schliessen und vereinen Benutzerkomfort, Reaktionsschnelligkeit und Interaktivität von Client/Server-Anwendungen mit der einfachen Verteilung und der globalen Reichweite von Web-Anwendungen.

Die folgende Abbildung illustriert die Evolution von Mainframe-Anwendungen zu RIA (Hein, 2005).

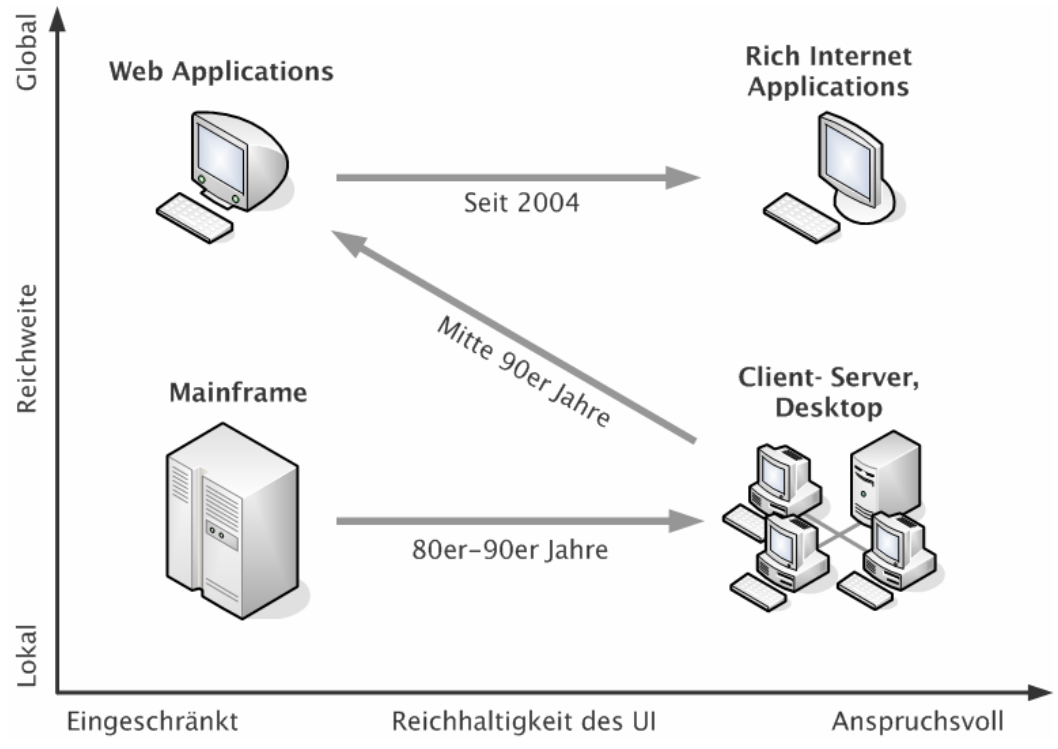

Abbildung 1: Evolution von Rich Internet Applikationen

Der Begriff „Rich Internet Applications“ wurde zuerst in einem Whitepaper von Macromedia im Jahre 2003 erwähnt (Mullet, 2003). Davor gab es schon verwandte Ideen die unter den Begriffen „Thin Clients“, „Remote Scripting“ oder „Rich Web Applications“ bekannt wurden.

2.1.2 Vergleich zwischen Client/Server-, HTML- und RIA

Um die Vor- und Nachteile von Client/Server-, HTML- und Rich Internet Applikationen zu verdeutlichen, werden die drei Technologien anhand von 8 verschiedenen Charakteristiken miteinander verglichen (Monson-Haefel, 2007).

Markterfolg:	Wie gut hat sich die Technologie im Markt durchgesetzt?
Reichweite:	Anzahl der Endgeräte und Betriebssysteme welche die Technologie unterstützen?
Interoperabilität:	Wie gut funktioniert die Technologie in Zusammenarbeit mit anderen Technologien?
Software-Verteilung:	Wie gut kann die Technologie auf den Endgeräten installiert werden?
Management:	Wie gut lassen sich verschiedene Versionen der Software verwalten?
Antwortzeitverhalten:	Wie leistungsfähig ist die Technologie in Bezug auf die Client-seitige Verarbeitungsgeschwindigkeit?
Zustand (State):	Unterstützt die Technologie das Halten von Client-seitigem Zustand?
User Interface:	Wie komfortabel sind die Möglichkeiten zur Darstellung des User Interfaces?

Stellt man die Bewertung der drei Technologien in einem Kiviat Diagramm dar, zeigt sich sehr schön, wie Rich Internet Anwendungen die Vorteile von Client/Server- und HTML-Anwendungen vereinen.

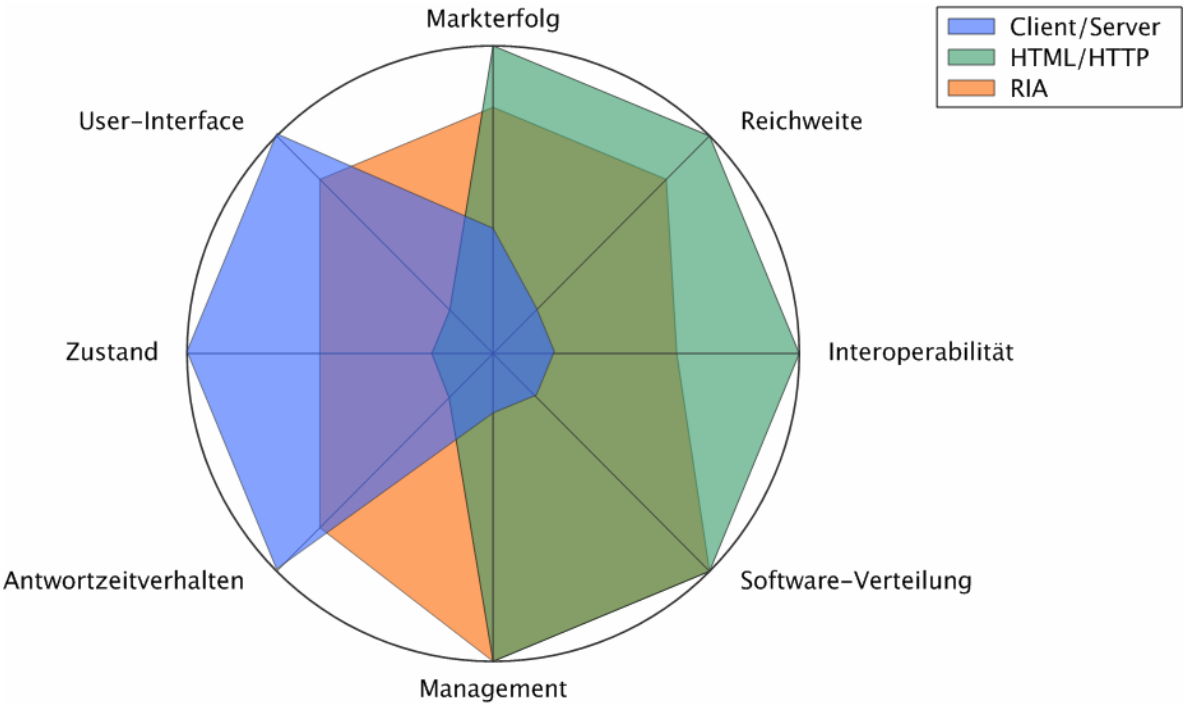

Abbildung 2: Vergleich von Client/Server-, HTML- und RIA (Monson-Haefel, 2007)

RIA Technologien bieten gegenüber HTML das Potential um die Benutzererfahrung von Web-Anwendungen signifikant zu erhöhen. Die folgende Tabelle zeigt anhand einiger Usability Kriterien die Unterschiede der beiden Technologie-Ansätze (Rogowski, 2006):

Usability Kriterien	HTML	RIA
Kontrolle	Benutzer sind bei Prozessen (z.B. Shopping Cart) an einen linearen Prozess gebunden.	Benutzer können sich zwischen den Schritten in einem Prozess schnell und einfach hin- und herbewegen.
Direktmanipulation	Elemente müssen durch Klicken selektiert werden (z.B. Checkbox) und danach eine Serveranfrage gestartet werden.	Objekte können direkt und unbehindert manipuliert werden (z.B. Drag und Drop, Inline Edit).
Fehlerverzeihung	Um Eingaben zu ändern oder rückgängig zu machen, muss sich der Benutzer zurückbewegen, mit dem Risiko gemachte Eingaben zu verlieren.	Benutzer können Aktionen rückgängig machen oder Daten ändern ohne lange Wartezeiten und Risiko des Datenverlusts.
Feedback	Es muss ein Neuladen der Seite abgewartet werden um zu sehen, ob die Benutzereingabe korrekt verarbeitet werden konnte.	Meldungen des Systems aufgrund einer Interaktion des Benutzers sind sofort ersichtlich.
Fehlerbehandlung	Feldvalidierungen verlangen in der Regel eine Serveranfrage. Fehlermeldungen können nicht direkt an der Stelle des Problems angezeigt werden.	Benutzereingaben können in Echtzeit überprüft werden und Fehlermeldungen können vor der Bestätigung des Formulars an der Stelle des Problems angezeigt werden.
Effizienter Task Flow	Prozesse müssen auf mehrere Seiten aufgeteilt werden. Zwischen den Seiten findet jeweils eine Serveranfrage statt und die komplette Seite muss neu geladen werden.	Ganze Prozesse können auf einem einzigen Screen abgebildet werden.

Tabelle 1: Vorteile von RIA in Bezug auf die Benutzererfahrung

2.1.3 Vor- und Nachteile von RIA

Die Vorteile von Rich Internet Anwendungen können wie folgt zusammengefasst werden.

- **Reichhaltigkeit**: RIA bieten bessere Interaktionsmöglichkeiten verglichen mit traditionellen HTML Anwendungen und erreichen heute beinahe den Komfort von Desktop Anwendungen.
- **Reaktionsgeschwindigkeit**: RIA ermöglichen die Aktualisierung von einzelnen Komponenten, ohne dass die ganze Seite neu geladen werden muss. Die Interaktion

wirkt für den Benutzer dadurch schneller und ruhiger. Ein weisser Bildschirm während des Seitenwechsels entfällt.

- **Sofortige Ausführung von Benutzereingaben**: Maus- und Tastatureingaben (Events) können von der Client-seitigen Programmlogik sofort verarbeitet werden. Komplexere Mausereignisse wie Rechtsklicks oder Drag und Drop können unterstützt werden.
- **Automatische Datenaktualisierung**: RIA können kontinuierlich Daten vom Server anfordern und auf dem Bildschirm aktualisieren und ebenfalls Daten, die vom Benutzer eingegeben wurden, automatisch auf dem Server speichern.
- **Entwicklung**: Standardisierte Tags und Programmierschnittstellen vereinfachen die Entwicklung von RIA Patterns. Moderne RIA Frameworks bieten umfangreiche Widget Bibliotheken mit wieder verwendbaren Interaktionselementen an.
- **Installation**: RIA benötigen keine spezielle Software-Installation. Es müssen lediglich ein Browser und je nachdem eine spezielle Laufzeitumgebung bzw. ein Plugin installiert sein.
- **Flexibilität**: Auf die Anwendungen kann von jedem Rechner im Netzwerk zugegriffen werden.
- **Plattformunabhängigkeit**: Die Anwendungen können auf verschiedenen Betriebssystemen ausgeführt werden.
- **Sicherheit**: Web-Anwendungen sind in der Regel weniger anfällig auf Viren als ausführbare Programme.
- **Joy of Use**: Neue Funktionen und Interaktionsmöglichkeiten verbessern die Benutzererfahrung, indem sie eine effizientere Arbeitsweise ermöglichen und mehr Spass machen.

Trotz den genannten Vorteilen haben RIA auch einige Nachteile.

- **Unerwartetes Verhalten**: Da RIA Patterns nicht auf dem Konzept der Seite als atomare Einheit beruhen, wird das mentale Modell des Benutzers verletzt (Nielsen, 2005). Mit RIA wird die Sicht des Benutzers auf die Informationen einer Seite durch eine Sequenz von Navigationsaktionen bestimmt, und nicht wie im klassischen Web durch einen einzigen Klick auf einen Link.
- **Aufforderungscharakter (Affordance)**: Interaktionsmöglichkeiten, die bei Desktop Applikationen intuitiv angewendet werden (z.B. Rechtsklick, Drag und Drop), werden auf einer Web Seite nicht erwartet und benötigen oft einen visuellen oder textuellen Hinweis, um von den Benutzern erkannt zu werden.
- **Sicherheitsgrenzen**: Da RIA in einem Browser ausgeführt werden, laufen die Anwendungen immer in einer so genannten "Sandbox". Dieser Sicherheitsmechanismus verhindert, dass bösartige Programme Schaden am Computer des Benutzers anrichten können. Aus diesem Grund ist der Zugriff auf Systemressourcen (z.B. lokaler Zugriff auf Dateien) sehr limitiert.
- **Technische Voraussetzungen**: RIA setzen in der Regel Scripting Funktionalitäten oder ein installiertes Browser-Plugin voraus. Wurde beispielsweise vom Benutzer Javascript deaktiviert, kann eine Anwendung möglicherweise nicht korrekt ausgeführt werden.
- **Antwortzeitverhalten**: Das Herunterladen der Client-seitigen Programmlogik (z.B. Javascript- oder Java Libraries und Plugins) verursacht bei der erstmaligen Anwendung längere Ladezeiten. JavaScript-reiche Anwendungen haben ebenfalls den Nachteil, dass der interpretierte Code langsamer ausgeführt wird als bei kompilierten Anwendungen.

- **Zustand (State)**: Die Abkehr vom Seitenparadigma hin zu einem zustandsbehafteten Mechanismus kann Probleme mit dem Back-Button oder den Bookmarks des Browsers verursachen, da ein bestimmter Zustand der Seite nicht mehr hergestellt werden kann.
- **Visibilität für Suchmaschinen**: Suchmaschinen funktionieren nach dem Prinzip, dass ein Roboter rekursiv alle Links verfolgt und den Inhalt der Seiten indexiert. Da RIA keine eindeutig adressierbaren Seiten besitzen, kann ihr Inhalt von Suchmaschinen nicht gefunden werden.
- **Barrierefreiheit**: RIA verursachen potentiell mehr Accessibility-Probleme als vergleichbare HTML Anwendungen, da die dynamischen Komponenten einer Anwendung nur schlecht von Screenreadern unterstützt werden können.

Einige der hier beschriebenen Probleme wurden bereits von den Entwicklern von Rich Internet Anwendungen adressiert und ansatzweise gelöst. Mit zunehmender Verbreitung von Web 2.0 werden auch die neuen Interaktionsmöglichkeiten einem breiteren Spektrum von Benutzern bekannt gemacht.

2.1.4 RIA-Plattformen

Heute gibt es vielfältige technische Möglichkeiten, Rich Internet Applikationen zu implementieren. Die verschiedenen Lösungen können in drei Gruppen eingeteilt werden.

Browser-basierte Lösungen

Browser-basierte RIA Implementierungen verwenden HTML, das Document Object Model (DOM) des Browsers, eine Scriptsprache und ein Ajax Framework. Diese Anwendungen benötigen keine spezielle Installation und werden automatisch ausgeführt, wenn JavaScript auf dem Computer des Anwenders aktiviert ist. Browser-basierte Lösungen haben das Problem, dass die Eigenheiten der verschiedenen Browser und Plattformen berücksichtigt werden müssen.

Player-basierte Lösungen

Anwendungen dieses Typs benutzen ein Browser Plugin als Laufzeitumgebung. Auf diese Weise können die RIA auf den Betriebssystemplattformen ausgeführt werden, für die das notwendige Plugin verfügbar ist. Auf Eigenheiten der verschiedenen Browser muss keine Rücksicht genommen werden, was den Testaufwand entsprechend reduziert. Von den Benutzern wird aber verlangt, dass sie die passende Version des Plugins installiert haben müssen. Die bedeutendsten Vertreter dieser Gruppe sind Macromedia Flash/Flex und Microsoft Silverlight.

Desktop-basierte Lösungen

Die dritte Option ist die Installation von Runtime Komponenten ausserhalb der Sicherheitsbeschränkungen des Browsers. Desktop-basierte Lösungen bieten vollen Zugriff auf das lokale Dateisystem und ermöglichen so die Persistierung von Daten auch ohne Internet-Verbindung. Im Weiteren verfügen diese Frameworks über eine solidere Sicherheitsarchitektur und bessere Integration von Multimedia-Inhalten als Ajax Frameworks. Ein Problem stellt bei diesen Lösungen die Verteilung und Wartung dar. Beispiele von Desktop-basierten RIA sind Adobe AIR und JavaFX.

Die folgende Abbildung ordnet die verschiedenen Lösungen einer Skala zwischen Thin Client (wenig Logik auf dem Client) und Rich Client (komplette Logik auf dem Client) zu.

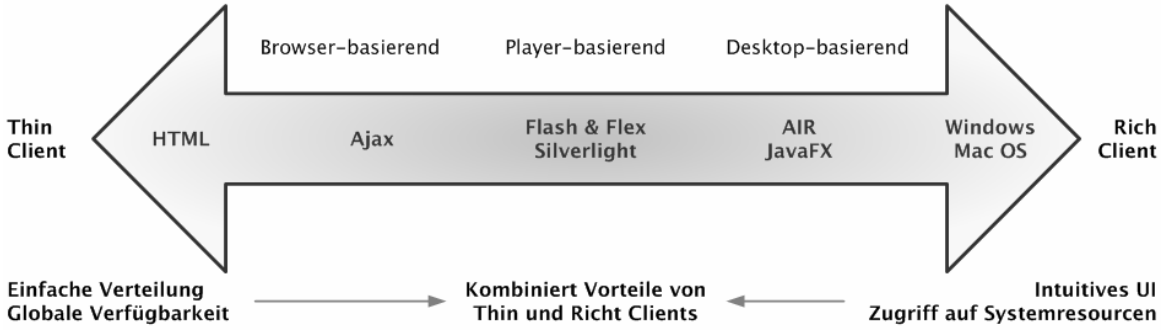

Abbildung 3: RIA Plattformen (Hammond & Goulde, 2007)

Alle Lösungen bieten Vorteile, haben aber auch Nachteile. Welche Lösung in welchem Fall am besten geeignet ist, hängt sehr stark vom Einsatzzweck ab.

2.1.5 RIA-Technologien

Eine Anzahl von RIA Technologien hat sich heute im Markt durchgesetzt. Die wichtigsten Vertreter werden auf den folgenden Seiten vorgestellt.

Ajax

Das Akronym Ajax (kurz für Asynchronous JavaScript and XML) wurde anfangs 2005 zum erstenmal in einem Artikel von Jesse James Garrett erwähnt. Es handelt sich dabei nicht um eine neue Technologie, sondern vielmehr um eine Kombination von bestehenden Web-Technologien wie (X)HTML, CSS, JavaScript, XML, XSLT und das Document Object Model (DOM). Der Begriff „Asynchronous“ steht dabei für die Möglichkeit, Daten zwischen dem Server und dem Browser auszutauschen, ohne dass dabei eine komplette Seite neu geladen werden muss. Ermöglicht hat dies das XMLHttpRequest-Objekt, eine Programmierschnittstelle, die aus JavaScript oder anderen Scriptsprachen verwendet werden kann und den Transfer von beliebigen Daten über das HTTP Protokoll unterstützt. Die XMLHttpRequest-Technik wurde im Internet Explorer 5 eingeführt und später von den anderen Browsern (Mozilla, Firefox, Opera, Konqueror und Safari) übernommen. Das Potential dieser Technik wurde lange Zeit unterschätzt und erst mit Google Maps oder GMail richtig populär.

Bei traditionellen Web-Applikationen muss bei jeder Benutzeraktion ein Request an den Server abgesetzt werden. Auf dem Server findet dann eine Verarbeitung statt (z.B. Zugriff auf eine Datenbank oder ein Backend-System) und es wird eine komplett neu erstellte Seite zurück an den Browser geschickt. Dieses Konzept beruht auf der Tatsache, dass das Web ursprünglich als Hypertext-Medium geplant war. Für den Bau von interaktiven Web-Anwendungen bedeutet dies eine erhebliche Einschränkung in Bezug auf das Design und das Antwortzeitverhalten des Systems.

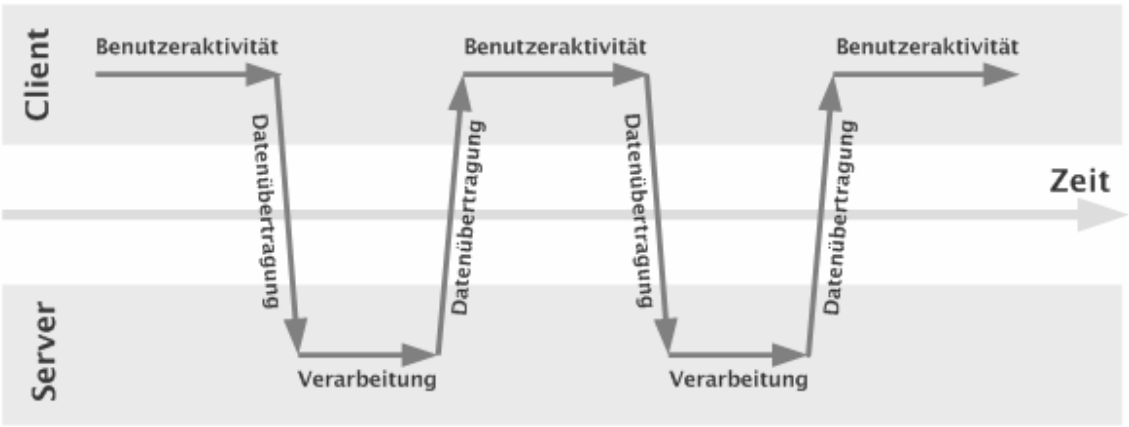

Abbildung 4: Klassisches Web-Applikationsmodell (synchron) (Garrett, 2005)

Bei einer Ajax Anwendung übernimmt ein zusätzlicher Layer auf der Client-Seite die Kommunikation zwischen dem Browser und dem Server. Diese Ajax-Engine ermöglicht es, dass asynchrone Requests an den Server geschickt werden und Daten (meistens in Form von XML Inhalten) zurück an den Browser transferiert werden. Diese Daten werden dann von der Ajax-Engine interpretiert und dynamisch in der aktuellen Seite dargestellt. Auf diese Weise können Fragmente der Seite ausgetauscht werden, ohne dass die gesamte Seite neu geladen werden muss.

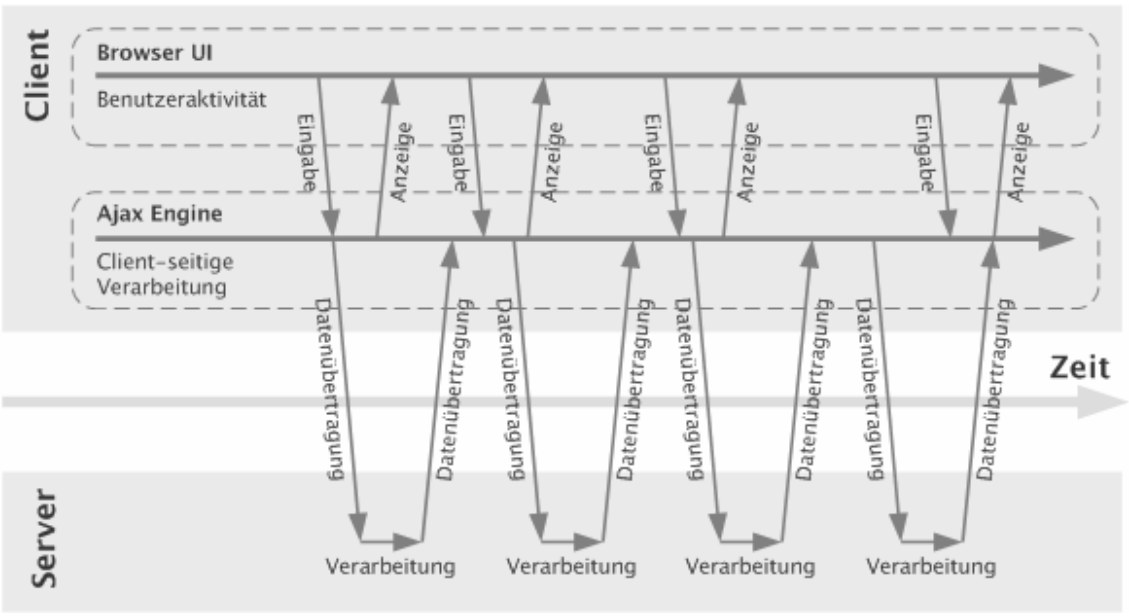

Abbildung 5: Ajax-Applikationsmodell (asynchron) (Garrett, 2005)

In den letzten Jahren kamen einige Ajax-Frameworks auf den Markt, die die Entwicklung von Ajax-Applikationen erheblich vereinfachten. Die Website AjaxPatterns.org (www.ajaxpatterns.org) zählt bereits mehr als 200 Ajax Frameworks.

Die Ajax Frameworks können grob in drei Kategorien eingeteilt werden:

Basisframeworks – Diese Frameworks stellen lediglich die Grundfunktionen für die Kommunikation zwischen dem Web-Browser und dem Server (Kapselung des XMLHttpRequest Objekts) und für die dynamische Manipulation von Inhalten auf dem Client bereit. Beispiele von Basisframeworks sind „Prototype“, „HTMLHttpRequest“ und „AjaxGear“.

Applikationsframeworks – Diese erweitern den Funktionsumfang der Basisframeworks um User Interface-Komponenten für die Entwicklung von Rich Internet Applikationen. Beispiele solcher Widgets sind Schieberegler, aufklappbare Trees, Editierbare Tabellen und animierte Übergänge. Beispiele dieser Kategorie sind „Dojo“, „jQuery“, „Yahoo! User Interface Library (YUI)“, „script.aculo.us“ und „ExtJS“.

Serverframeworks – Diese Frameworks vereinen die Logik auf dem Client mit dem Programmcode auf der Serverseite. So können beispielsweise Server-Komponenten entwickelt werden, die zur Laufzeit den notwendigen Ajax Programmcode für den Client generieren. Beispiele von Serverframeworks sind „Google Web Toolkit (GWT)“, „Direct Web Remoting (DWR)“ und „xajax“.

Adobe Flash/Flex

Adobe Flex ist eine Plattform für die Entwicklung von Rich Internet Anwendungen auf der Basis des Adobe Flash Players. Flex umfasst ein Software Development Kit für die Sprachen MXML (XML-basierte Programmiersprache für die deklarative Beschreibung des User Interfaces) und ActionScript 3.0 (Implementierung der Anwendungslogik) inklusive einer Komponenten-Library, Flex-Builder (Eclipse Plugin für die Entwicklung von Flex Anwendungen), Lifecycle Data Services (Dienste für die Anbindung von Backend-Datenquellen) und Flex Charting (Library für die Darstellung von Diagrammen). Flex Anwendungen können entweder mit dem kostenlosen Flex-SDK oder dem kostenpflichtigen Adobe Flex Builder entwickelt werden. Der Flex Builder bietet ein graphisches Drag und Drop Interface für das Layout und die Anordnung der User Interface Elemente sowie eine Zeitachse für animierte Abläufe. Die Flex Klassenbibliothek enhält Components (UI Elemente), Layout Manager für die Anordnung des User Interfaces, Behaviours (Verhalten und Effekte von Elementen) und Service Components für die Kommunikation mit Web Services (SOAP oder REST). Durch die gute Verfügbarkeit und die grosse Verbreitung des Flash Players sind die technischen Voraussetzungen für die Ausführung von Flex Anwendungen auf den meisten Computern gegeben.

Microsoft Silverlight

Silverlight ist ein browser- und plattformübergreifendes Plugin für Rich Internet Anwendungen das zuvor unter dem Codenamen “WPF/E” (Windows Presentation Foundation/Everywhere) bekannt wurde. In der Tat handelt es sich um eine eingeschränkte Version von WPF, einer neuen User-Interface Programmierschnittstelle (API) für Microsoft Windows. Silverlight-basierte Anwendungen werden mit XAML (extensible Application Markup Language) und JavaScript entwickelt. XAML wird für die deklarative Beschreibung des User Interfaces (grafische Elemente, Benutzeroberflächen, Animationen etc.) verwendet. Die Programmlogik (z.B. die Verarbeitung der Events) wird mittels JavaScript implementiert. Obwohl Silverlight eng mit dem .NET Framework von Windows verknüpft ist, benötigen die Anwendungen für die Ausführung lediglich das Silverlight Plugin. Dieses ist im Augenblick für Internet Explorer 6/7, Firefox 1.5/2.x und Safari auf den verschiedenen Windows Versionen und Mac OS verfügbar.

Adobe AIR

Adobe AIR (Adobe Integrated Runtime) ist eine plattformübergreifende Laufzeitumgebung für die Entwicklung von Rich Internet Applikationen. Die Entwickler können bekannte Web-Techniken wie Flash, Flex, HTML, Javascript, CSS und Ajax einsetzen, die Anwendungen werden aber statt in einem Browser in einer eigenen Laufzeitumgebung auf dem Desktop ausgeführt. Auf diese Weise wird versucht, die Vorteile von Web- und Desktop Anwendungen zu kombinieren. Adobe nennt dieses Verfahren ein „Cross-Operating System“. Ermöglicht werden beispielsweise Drag und Drop zwischen AIR und Desktop-Anwendungen, die Einbindung der Zwischenablage und Desktop- und Systemtastenkürzel. Ebenfalls ist es möglich, AIR Anwendungen offline zu betreiben.

Erste AIR Applikationen gibt es beispielsweise von eBay oder von der New York Times. Bei eBay wird eine Auktion zum eigenständigen Programm auf den Desktop, was eine gezieltere Benutzerführung ermöglicht. Es können z.B. Alarme bei der Beobachtung von Auktionen oder bei der Suche nach bestimmten Waren ausgelöst werden, ohne dass eine Benutzerinteraktion stattfindet. Im Augenblick werden die Betriebssysteme Windows, Mac OS und Linux (Alpha) unterstützt.

JavaFX

JavaFX ist eine neue Plattform von Sun für die Entwicklung von Rich Internet Anwendungen für Desktops, mobile Endgeräte, TV Settop Boxen und Blu-ray Disk Player. JavaFX besteht im Augenblick aus JavaFX Script und Java FX Mobile. JavaFX Script ist eine deklarative, objektorientierte und statisch typisierte Sprache, die auf der Java Plattform aufbaut und darum auch vollen Zugriff auf das Java API hat. Die gesamten GUI Komponenten (z.B. Swing Widgets) werden in Form von Java Objekten bereitgestellt. JavaFX dient dann dazu, diese Komponenten in einem GUI Layout einzusetzen. Aufgrund der reduzierten Komplexität richtet sich die Sprache vor allem an GUI Entwickler und Designer, die Erfahrungen im Umgang mit Scriptsprachen haben. JavaFX Mobile ist ein Betriebssystem für mobile Endgeräte wie PDAs oder Smartphones. Es enthält einen Linux Kernel sowie eine Java Virtual Machine und ermöglicht zusammen mit JavaFX Script die Entwicklung von anspruchsvollen portablen Anwendungen. JavaFX steht vor allem in Konkurrenz mit Adobe AIR, Adobe Flex und Microsoft Silverlight im Markt der Nicht-Ajax Frameworks.

2.1.6 Web 2.0 und RIA

Wir haben gesehen, dass es sich bei Rich Internet Applikationen um ein neues technologisches Paradigma handelt, um reichhaltige und dynamische Anwendungen über das Web zu verbreiten. Technologien wie Ajax oder Flash werden oft in Verbindung mit dem Begriff „Web 2.0“ gebracht. Tatsächlich handelt es sich bei Web 2.0 Anwendungen oft auch um Rich Internet Applikationen, trotzdem können die beiden Begriffe nicht synonym verwendet werden. Web 2.0 ist vielmehr eine Sammlung von Business Strategien, sozialen Trends und neuen Technologien. Anstelle von Web 2.0 werden auch die Begriffe Wisdom Web, People-Centric Web, Partizipatives Web und Read/Write Web genannt.

Es gibt bis heute keine genaue Definition des Begriffes Web 2.0. Allgemein gültig wird darunter ein Sammelbegriff für eine Menge von Eigenschaften verstanden:

- Ermöglichung einer verbesserten Benutzererfahrung durch dynamische und flexible User Interfaces
- Unterstützung von Kollaboration und gemeinsamer Inhaltserstellung und -modifikation
- Bereitstellung von sozialen Netzwerken für Menschen mit gleichen Interessen
- Erstellung von neuen Anwendungen durch die Wiederverwendung, Kombination und Zusammenführung von bestehenden Web-Inhalten (Mashups)

Der Begriff “Web 2.0” wurde von Tim O’Reilly an der O’Reilly Media 2004 Konferenz geboren und später im Artikel “What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software” (O’Reilly, 2005) zusammengefasst.

Im Vergleich zum Web 2.0 war das traditionelle Web (heute auch Web 1.0 genannt) ein statisches Einweg-Publikations-Medium. Der Übergang vom Web 1.0 zum Web 2.0 kann anhand von konkreten Beispielen illustriert werden (vgl. O’Reilly, 2005):

Web 1.0		Web 2.0
Browser	→	Applikations-Client Container
Thin Client	→	Rich Client
Paged Internet Applications (PIA)	→	Rich Internet Applikationen (RIA)
HTML	→	Ajax, Flash/Flex, AIR, Silverlight etc.
Statisch	→	Dynamisch, interaktiv
HTTP (Pull)	→	Publish/Subscribe (Push)
Applikationsserver	→	Mashup Server
Enterprise Application Integration	→	Enterprise Mashup/SOA
Content Management Systeme	→	Wikis, Blogs
Kommerzielle Software	→	Open Source Software
Top-Down Kontrolle (Diktatur)	→	Bottom-Up (Demokratie)
Directories (Taxonomie)	→	Tagging (Folksonomy)

Tabelle 2: Vergleich zwischen Web 1.0 und Web 2.0

Die Popularität von Web 2.0 Anwendungen wurde vor allem durch Sites wie MySpace, Facebook, Flickr, Google Maps und YouTube begünstigt. Heute trifft man vermehrt auch Web 2.0-Charakteristiken in Business- und Enterprise-Anwendungen an.

In einem Artikel der New York Times von Ende 2006 (Markoff, 2006) wurde zum ersten Mal der Begriff Web 3.0 erwähnt. Web 3.0 beschreibt eine dritte Generation von Web-Technologien und –Diensten, welche die maschinenunterstützte Verknüpfung und Aggregation von Informationen im Web unterstützen sollen. Es wird in diesem Zusammenhang auch vom Semantischen Web gesprochen.

2.2 Interaktions-Design Patterns

Design Patterns sind generell anwendbare Lösungen auf ein bestimmtes Problem im Bereich der Gestaltung von Benutzeroberflächen. Um zu beurteilen, wie gut sich verschiedene Techniken für das Prototyping von Rich Internet Applikationen eignen, ist es notwendig, die möglichen Interaktionsmuster von RIA zu identifizieren.

2.2.1 Entwicklung von Design Patterns

Die Idee von Design Patterns stammt ursprünglich aus dem Bereich der Architektur. Christopher Alexander beschreibt in den 70er Jahren in seinem Buch „A Pattern Language. Towns, Buildings, Construction“ (Alexander, Ishikawa, Silverstein, 1977) eine Sammlung von Entwurfsmustern, die komplexe Aspekte und Strukturen in der Architektur logisch zusammenführen.

Alexander beschreibt ein Pattern wie folgt:

> *„Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.“*
>
> *– Christopher Alexander, A Pattern Language, 1977*

Interessanterweise stiess der Ansatz von Alexander in der Architektur nie auf grosse Akzeptanz, stattdessen übernahmen andere Disziplinen das Konzept. 1987 beschrieben Kent Beck und Ward Cunningham (1987) Design Patterns als eine Möglichkeit, um

wiederkehrende Problemstellungen in der objektorientierten Softwareentwicklung zu vereinfachen. Mit dem Buch „Design Patterns: Elements of Reusable Object-Oriented Software“ (Gamma, Helm, Johnson, Vlissides, 1995) der so genannten "Gang of Four", wurde die Idee von Design Patterns für die Softwarearchitektur weiterverfolgt und populär gemacht.

Die ersten Sammlung an Interaktions-Design Patterns wurden von Jenifer Tidwell (1997) oder Martijn van Welie (2001) (van Welie & van der Veer 2003) entwickelt.

2.2.2 Typen von Interaktions-Design Patterns

Ein einzelnes Pattern beschreibt eine bewährte Lösung auf ein Problem in einem bestimmten Design Kontext. Stellt man mehrere verwandte Patterns miteinander in Beziehung, entsteht daraus eine Pattern Sprache. Alexander (Alexander et al., 1977) gruppierte seine Patterns hierarchisch, ausgehend von Städten über Wohnviertel, Häuser bis zu Türen und Fenstern.

Interaktions-Design Patterns können ebenfalls hierarchisch gegliedert werden, ausgehend von einer globalen Systemebene bis zu den einzelnen Interaktionselementen (Widgets). Cooper (2007) gruppiert die Patterns in drei verschiedene Ebenen.

Postural Patterns werden auf einer konzeptionellen Ebene verwendet und bestimmen den allgemeinen Zweck der Anwendung.

Structural Patterns beschreiben Problemstellungen für die Anordung und Gruppierung von Informations- und Interaktionselemente auf dem Bildschirm.

Behavioral Patterns beschreiben die elementaren Informations- und Interaktionselemente (Widgets).

Im Paper von van Welie und van der Veer (2003) wird diese Einteilung weiter verfeinert in die Ebenen Business Goals, Posture Level Patterns, Experience Level Patterns, Task Level Patterns und Action Level Patterns.

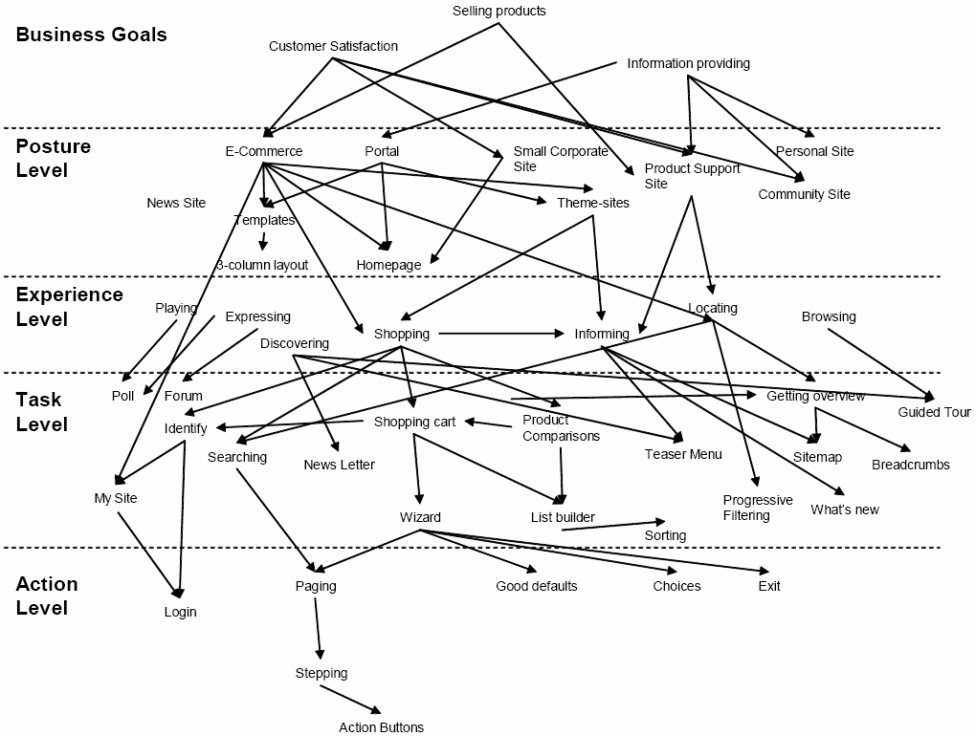

Abbildung 6: Klassifikation von Interaktions-Design Patterns (van Welie et al., 2003)

2.2.3 RIA Interaktions-Design Patterns

Damit wir die Eignung von Prototyping-Methoden für die Evaluation von Rich Internet Applikationen bewerten konnten, mussten wir zuerst definieren was ein RIA Interaktions-Design Pattern auszeichnet und wie es sich von einem traditionellen Web oder Desktop Pattern abgrenzt.

Gemäss Tibbets (2008) können RIA Anwendungen mit den folgenden drei Schlüsselmerkmalen charakterisiert werden.

1. **Aktualisierung von Teilen der Anzeige („Live Look“)**
   Alle Elemente des User Interfaces können isoliert und dynamisch aktualisiert werden, ohne dass dabei die umliegenden Bereiche neu geladen werden müssen. Der Zustand der Seite wird in einer Benutzersitzung gespeichert.
2. **Direkte Verarbeitung von Benutzerinteraktionen („Live Feel“)**
   Tastatur- und Maus-Events werden von der Client-seitigen Logik sofort verarbeitet. Dies ermöglicht es, dass Interaktionen direkt und im Kontext ausgeführt werden können.
3. **Asynchrone Datenkommunikation**
   Daten können in Echtzeit zwischen Client und Server ausgetauscht werden, ohne dass der Benutzer bei der Arbeit blockiert wird.

Bill Scott (2009) geht einen Schritt weiter und beschreibt 6 Design Prinzipien für Rich Internet Applikationen:

1. **Make It Direct**
   Der Benutzer kann direkt mit der Anwendung interagieren und Aktionen im Kontext ausführen. Inhalte können in der Seite editiert und Objekte direkt selektiert und manipuliert werden.
2. **Keep It Lightweight**
   Durch das Anbieten von kontext-spezifischen Werkzeugen (z.B. Kontext-Menus) kann die Komplexität einer Benutzeraufgabe minimiert werden.
3. **Stay on the Page**
   Mit Hilfe von Überlagerungen, dynamisch eingefügten Inhalten und Inline-Prozessen wird verhindert, dass eine Interaktion des Benutzers einen Seitenwechsel verursacht.
4. **Provide an Invitation**
   Hinweise und Aufforderungen sollen dem Benutzer helfen, die neuen Interaktionsmöglichkeiten besser zu verstehen und anzuwenden.
5. **Use Transitions**
   Statusänderungen und Beziehungen zwischen Objekten sollen für die bessere Beachtung hervorgehoben oder animiert werden.
6. **React Immediately**
   Durch die sofortige Reaktion des Systems auf eine Benutzerinteraktion wird die Anwendung als schneller und lebhafter wahrgenommen. Um Fehler frühzeitig vermeiden zu können, werden Meldungen und Fehler unmittelbar angezeigt.

Aus diesen beiden Definitionen lassen sich mehrere Kriterien ableiten. Damit wir von einem RIA Interaktions-Design Pattern sprechen können, müssen eine oder mehrere Bedingungen erfüllt sein.

Das Pattern ...

- wird durch ein Benutzer- oder System-generiertes Ereignis ausgelöst
- unterstützt die direkte Manipulation eines Objekts
- ermöglicht die Ausführung einer Aktion im Kontext des ausgewählten Objekts

- aktualisiert Teile des Bildschirm, nicht die komplette Seite
- unterstützt den asynchronen Austausch von Daten
- bietet notwendige Informationen und Aufforderungen wenn sie benötigt werden
- verhindert Fehler frühzeitig
- reagiert sofort auf Interaktionen des Benutzers
- behält seinen aktuellen Zustand bzw. kann ihn jederzeit wieder herstellen
- animiert Übergänge, Zustandsveränderungen und Beziehungen zwischen Objekten

2.2.4 Design Pattern Sammlungen und ihre RIA-Abdeckung

Es gibt heute eine Vielzahl von Büchern und Online Ressourcen mit Beschreibungen von Interaktions-Design Patterns. Allerdings enthalten erst wenige der Sammlungen Patterns, welche die neuen technischen Möglichkeiten von Rich Internet Applikationen berücksichtigen. Die wichtigsten Pattern-Bibliotheken werden kurz vorgestellt und in Bezug auf ihre RIA Abeckung bewertet.

Designing Interfaces

Jenifer Tidwell beschreibt in ihrem Buch "Designing Interfaces" (Tidwell 2005) 82 verschiedene Interaktionsdesign-Patterns. Etwas mehr als die Hälfte davon sind auch auf der begleitenden Web Site (Tidwell 2005) publiziert. Viele der Patterns beschreiben Problemstellungen von Desktop Applikationen, diese können aber grösstenteils auch auf Rich Internet Applikationen adaptiert werden.

Die Patterns sind nach Problemstellungen strukturiert: “Organizing the Content”, “Getting Around”, “Organizing the Page”, “Doing Things“, “Showing Complex Data”, “Getting Input from Users”, “Builders and Editors” und “Making It Look Good”.

Beispiele von RIA- bzw. Desktop-Patterns: “Extras on Demand”, “Animated Transition”, “Progress Indicator”, “Overview plus Detail”, “Datatips”, “Dynamic Queries”, “Local Zooming”, “Sortable Tables”, “Jump to Item”, “Cascading Lists”, “Tree Table”, “Autocompletion”, “Dropdown Chooser” und “Edit-in-Place”.

Yahoo! Design Pattern Library

Die Yahoo! Design Pattern Library (Yahoo 2006) wurde ursprünglich für interne Zwecke entwickelt, um die Konsistenz der Anwendungen bei Yahoo sicherzustellen. Seit Februar 2006 steht die Library unter einer Open Source Lizenz frei zur Verfügung und beschreibt heute etwa 40 Patterns. In Ergänzung zur Design Pattern Library wird auch die Yahoo! User Interface Library (YUI) angeboten, ein sehr gut dokumentiertes Ajax Framework mit mehr als 30 Komponenten. Bemerkenswerterweise sind nicht alle Komponenten auch als Patterns beschrieben. So gibt es beispielsweise für das Pattern Auto Complete ein entsprechendes YUI Widget, während dem ein Tree View Widget oder ein Rich Text Editor nicht als Patterns erwähnt werden. Durch die starke Web 2.0 Orientierung von Yahoo bietet die Library grösstenteils RIA Patterns und Komponenten.

Die Patterns sind nach folgenden Themen strukturiert: “Search”, “Navigation” (Links, Tabs), “Browsing”, “Selection”, “Rich Interaction”, “Social” (Ratings & Reviews, Reputation).

Beispiele von RIA Patterns sind “Auto Complete”, “Calendar Picker”, “Carousel”, “Drag and Drop”, “Hover Invitation”, “Cross Fade” und “Rating an Object”.

A Pattern Library for Interaction Design

Martijn van Welie (van Welie 2001) publiziert auf seiner Online Library mehr als 130 Web Design Patterns. Die Patterns sind gruppiert in User Needs, Application Needs und Context of Design. Besucher können die Patterns kommentieren und Vorschläge für neue Patterns einreichen. Bis heute sind erst relativ wenig RIA Patterns dokumentiert und Patterns mit

RIA Potenzial werden teilweise noch mit „traditionellen“ Interaktionsformen beschrieben. Die Steuerung im Map Navigator wird beispielsweise noch mit Navigationslinks an den Rändern der Karte statt durch direktes Ziehen der Karte gemacht.

Die Patterns sind in die drei Kategorien “User needs” (Navigating around, Basic interactions, Searching, Dealing with data, Personalizing, Shopping, Making choices, Giving input, Miscelleaneous), “Application needs” (Drawing attention, Feedback, Simplifying interaction) und “Context of design“ (Site types, Experiences, Page types) eingeteilt.

Beispiele von RIA Patterns sind “Accordion”, “Autocomplete”, “Fly-out Menu”, “Overlay Menu”, “Date Selector” und “Pulldown Button”.

UI Patterns

Diese neue Pattern Library von Anders Toxboe (Toxboe 2007) beschreibt bis jetzt etwa 45 Web Design Patterns. Da die Library erst seit 2007 online ist, orientiert sie sich sehr stark an den neuesten technischen Interaktionsmöglichkeiten, die in Web 2.0 Anwendungen eingesetzt werden. Die Patterns sind jeweils mit Beispielen und teilweise mit Programmieranleitungen unterlegt.

Die Patterns sind nach folgenden Themen strukturiert: ”Getting input from users“ (Forms, Explaining the process, Community driven), ”Dealing with data“ (Search, Tables, Images, Browsing, Formatting data), ”Navigation“, ”Tabs“ (Jumping in hierarchy, Fly-out menus), ”Miscellaneous“ (Shopping, Increasing frequency).

Beispiele von RIA Patterns sind “Password Strength Meter”, “Rate Content”, “Inline Help Box”, “Live Preview”, “Input Feedback”, “Live Filter”, “Continuous Scrolling”.

Ajax Patterns

Das Buch von Michael Mahemoff (Mahemoff 2006) beschreibt neben 28 sogenannten Functionality und Usability Patterns auch Ajax Technologie- und Programmier-Patterns. Begleitend zum Buch gibt es ein öffentliches Wiki, welches Ajax Patterns, Frameworks und –Tools sowie Links rund um die Ajax Technologie dokumentiert. Wie der Name des Buchs vermuten lässt, handelt es sich bei den visuellen Patterns ausschliesslich um Komponenten von RIA Patterns.

Die Patterns im Kapitel “Functionality und Usability Patterns“ sind in die Kategorien “Widgets”, “Page Architecture”, “Visual Effects” und “Functionality” gegliedert.

Beispiele von RIA Patterns sind “Slider”, “Progress Indicator”, “Rich Text Editor”, “Live Search”, “Drag and Drop”, “Microlink” und “Virtual Workspace”.

The Design of Sites

Das Buch „The Design of Sites: Patterns for Creating Winning Web Sites“ (van Duyne, Landay et al. 2006) ist bereits in der zweiten Auflage erschienen und dokumentiert 107 Patterns. Es werden vor allem generell anwendbare Web Design Konzepte, wie beispielsweise ein Shopping Cart oder eine Sitemap erläutert. Auf einzelne Interaktionselemente und Widgets wird nur sehr vereinzelt eingegangen. Es sind ebenfalls nur sehr wenige Patterns mit RIA Charakteristiken enthalten.

Die Patterns sind in folgende Kategorien eingeteilt: “Site Genres”, “Creating a Navigation Framework”, “Creating a Powerful Homepage”, “Writing & Managing Content”, “Building Trust & Credibility”, “Basic E-Commerce”, “Advanced E-Commerce”, “Helping Customers Complete Tasks”, “Designing Effective Page Layouts”, “Making Site Search Fast & Relevant”, “Making Navigation Easy”, “Speeding Up Your Site” und “The Mobile Web”.

Beispiele von RIA Patterns sind “Floating Windows”, “Direct Manipulation”, “Predictive Input”, “Progress Bar” und “Jump Menus”.

Designing Web Interfaces

Das Buch “Designing Web Interfaces - Principles and Patterns for Rich Interaction” von Bill Scott (2009) ist zum Zeitpunkt der Abgabe dieser Arbeit noch nicht erschienen. Die begleitende Web Site (Scott, 2009) gibt allerdings schon Einblicke in Sammlung an Patterns. Mit 65 beschriebenen RIA Patterns stellt diese Sammlung die umfangreichste Kollektion ihrer Art dar.

Die Patterns sind in folgende Kategorien eingeteilt: “In-Page Editing“, “Drag and Drop“, ”Direct Selection“, ”Contextual Tools“, “Overlays“, “Inlays“, “Virtual Pages“, “Process Flows“, “Static Invitations“, “Dynamic Invitations“, “Transitional Patterns“, “Purpose of Transitions“, “Lookup Patterns“, “Feedback Patterns”.

Beispiele von RIA Patterns sind “Table Edit”, “Drag und Drop Modulues“, “Hover-Reveal Tools“, “Detail Overlay“, “Inline Paging“, “Hover Invitation“, “Spotlight, “Slide in/Slide out”, “Autocomplete” und “Live Preview”.

2.3 Erstellung und Verwendung von Prototypen

Nach der Einführung von Interaktions-Design Patterns im vorangegangenen Unterkapitel beschreiben wir im Folgenden, durch was sich Prototypen auszeichnen und wozu sie im Rahmen eines durch die ISO normierten Softwareentwicklungsprozesses eingesetzt werden. Dabei gehen wir vertieft auf den Prototyping Prozess von Arnowitz, Arent & Berger (2007) ein. Weiterhin stellen wir verschiedene Prototyping Methoden vor und diskutieren deren Wiedergabetreue als Klassifikationsmerkmal. Schliesslich vergleichen wir (überwiegend elektronische) Werkzuge, mit denen Prototypen erstellt werden können.

2.3.1 Definition von Prototyp

Die ISO Norm 13407 definiert Prototyp wie folgt: „*Representation of all or part of a product or system that, although limited in some way, can be used for evaluation*.“ (ISO, 1999).

Aus dieser Definition lassen sich zwei zentrale Eigenschaften von Prototypen ableiten:

a) Prototypen sind Vorläufer des späteren Systems und weisen noch nicht alle gewünschten Eigenschaften auf. Sie können das gesamte System oder bestimmte Teile davon abbilden.

b) Prototypen können verwendet werden um zu prüfen, inwieweit sie bestimmte Anforderungen erfüllen.

2.3.2 Einbettung von Prototyping in den Entwicklungsprozess von Software

Die Erstellung von Prototypen ist ein integraler Bestandteil des Gestaltungsansatzes des *User-Centered Design*. Bei diesem beruht der Gestaltungsprozess auf Informationen über die Menschen, die das Produkt verwenden werden. Somit stehen die Nutzer während der gesamten Planung, Gestaltung und Entwicklung eines Produktes im Mittelpunkt.

Der internationale Standard ISO 13407 (*Benutzer-orientierte Gestaltung interaktiver Systeme*) normiert das Vorgehen beim *User-Centered Design*. Der Standard beschreibt das idealtypische Vorgehen eines benutzerorientierten Softwareentwicklungsprozesses in vier Teilschritten:

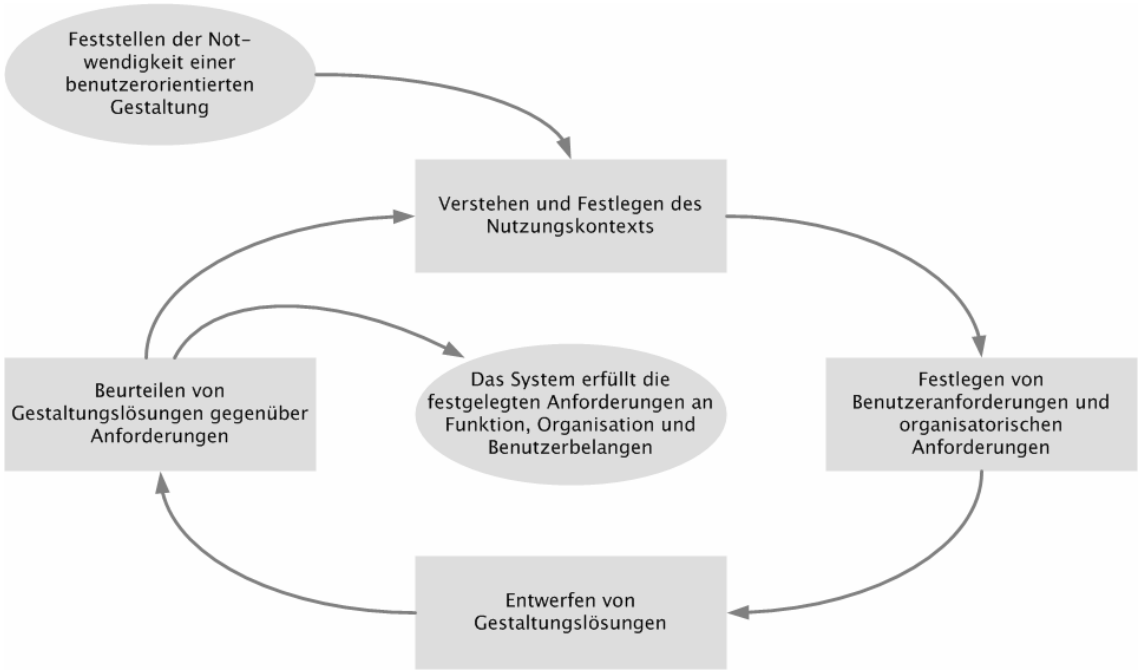

Abbildung 7: Benutzer-orientierter Gestaltungsansatz nach ISO 13407

a) Nutzungskontext verstehen und beschreiben. Dabei wird untersucht, wer später das Produkt verwenden wird, für welche Ziele und unter welchen Umgebungsbedingungen.

b) Anforderungen spezifizieren. Welche Geschäftsanforderungen (Business Requirements) und Nutzerziele müssen erfüllt sein, damit das Produkt erfolgreich sein wird?

c) Gestaltungslösungen erstellen. Diese Lösungen entstehen in Form von Prototypen. Über mehrere Iterationen des Entwicklungsprozesses hinweg nähert man sich so schrittweise einer tragfähigen Lösung an - vom groben Konzept bis zum ausgearbeiteten Design.

d) Gestaltungslösungen bewerten. In diesem wichtigen Teilschritt werden die Gestaltungsentwürfe daraufhin geprüft, inwieweit sie die in Schritt b) definierten Anforderungen erfüllen. Idealerweise werden hierbei Tests mit Nutzern aus der Zielgruppe des Produktes durchgeführt. Art und Ausmass der unerfüllten Anforderungen bestimmen dabei, ob eine weitere Iteration im Entwicklungsprozess notwendig ist und worauf gegebenenfalls beim nächsten Durchlauf der Entwicklungsschwerpunkt gelegt werden sollte.

2.3.3 Zweck und Ziele von Prototyping

Für einen nutzerorientierten Softwareentwicklungsprozess sind somit die Erstellung und Überarbeitung von Prototypen von zentraler Bedeutung. Prototypen sollten früh erstellt und immer wieder überarbeitet werden. Ihre Stärke liegt in der anschaulichen Darstellung von Lösungsansätzen: Anstatt funktionale Anforderungen an das interaktive System nur mit Wörtern zu beschreiben, werden diese visualisiert. Diese Konkretisierung erleichtert erheblich Diskussionen über innovative Problemlösungen.

Weiterhin können Lösungsansätze in einer Form ausprobiert und erforscht werden, die sich ohne grossen Zeit- oder Arbeitsaufwand anpassen lässt. Dies erleichtert es erheblich, verschiedene Lösungsvarianten auszuprobieren und miteinander zu vergleichen. Das Risiko von Fehlentwicklungen sinkt, da sie mit dieser Exploration früher erkannt und beseitigt werden können.

Über den Wert für das eigentliche Entwicklungsteam hinaus erfüllen Prototypen noch eine ganze Reihe von weiteren Funktionen:

- Prototypen ermöglichen, Lösungsansätze frühzeitig mit Kunden zu evaluieren
- Sie erlauben Entwicklern, die technische Umsetzbarkeit und den Aufwand der Umsetzung von Lösungsansätzen fundierter zu bewerten
- Sie veranschaulichen für weitere interne Interessensgruppen (wie z.B. den Auftraggeber) den Entwicklungsstand der Arbeiten
- Sie bilden die Grundlage der Spezifikation einer Benutzeroberfläche. Prototypen erleichtern für Entwickler wesentlich das Verständnis und die Eindeutigkeit von Spezifikationen.

Prototypen sind folglich Kommunikationshilfe, Arbeitsmittel und -ergebnis in einem.

2.3.4 Prototyping Prozess

Die Planung, Vorbereitung, Entwicklung und Evaluation von Prototypen sollte in einem strukturierten Prozess erfolgen. Arnowitz et al. (2007) schlagen hierfür eine Vorgehensweise in vier Phasen und elf Schritten vor:

Phase I: Planung

Die Planungsphase dient dazu, die Prototyping Bedürfnisse zu bestimmen und den Prototyping Prozess entsprechend zu planen. Dabei wird auch festgelegt, welche Teile des interaktiven Systems durch Prototypen modelliert werden sollen.

Schritt 1: Anforderungen prüfen

Der Prozess beginnt mit der Festlegung von Anforderungen an das Prototyping. Dazu werden aus den gesamten Anforderungen an die Software diejenigen ausgewählt, die für das geplante Zielpublikum am wichtigsten sind. Auf diese Weise werden die im Prototyp zu modellierenden w Aufgaben und Inhalte bestimmt. Dazu zählt auch die Entscheidung, ob eher ein horizontaler oder vertikaler Prototyp erstellt werden soll (Beaudouin-Lafon & Mackay, 2003). Ein horizontaler Prototyp stellt eine Software-Schicht (wie zum Beispiel die Benutzeroberfläche) umfassend dar, während darunter liegende Funktionalitäten noch nicht zur Verfügung stehen. Ein vertikaler Prototyp hingegen realisiert alle Software-Schichten einer ausgewählten Funktionalität, so dass diese Anwendungsfälle bereits durchgehend simuliert werden können.

Schritt 2: Aufgaben- oder Bildschirmabfolge erstellen

Interaktive Systeme sind ab einer gewissen Grösse in Seiten oder Bereiche unterteilt. Die Gliederung bzw. Abfolge dieser Einheiten sollten die Ziele der Nutzer möglichst gut unterstützen. Dazu muss eine Aufgaben- oder Bildschirmabfolge bestimmt werden. Sie beruht auf einem Verständnis der strukturellen Zusammenhänge von Handlungsabfolgen des Nutzers. Hierbei kann es sehr hilfreich sein, die Gestaltung des Systems und damit des Prototyps an typischen Handlungen der Nutzer auszurichten. Diese können in Form von Use Cases oder Szenarien modelliert werden (Beaudouin-Lafon & Mackay, 2003). Von der Strukturierung der Inhaltsbereiche können anschliessend auch die Navigationsmöglichkeiten abgeleitet werden.

Schritt 3: Inhalt und Wiedergabetreue (Fidelity) bestimmen

Die zu modellierenden Inhalte eines Prototyps können auf verschieden Dimensionen in unterschiedlicher Wiedergabetreue ausgearbeitet werden:

- Informationsdesign
- Interaktionsdesign
- Navigationsmodell
- Visuelles Design

- Text- und Dateninhalte
- Markenidentität (Branding)
- Systemleistung und -verhalten

Die sonst so verbreitete Kennzeichnung eines Prototyps als einen mit geringer oder hoher Wiedergabetreue („low/ high fidelity prototype“) ist auf diesem Hintergrund eine äusserst grobe und damit wenig aussagekräftige Vereinfachung.

Phase II: Spezifikation

In der zweiten Phase des Prototyping Prozesses werden die Entscheidungen der ersten Phase genutzt, um die gewünschten Merkmale des Prototyps zu bestimmen.

Schritt 4: Prototype Merkmale bestimmen

Beim Prototyping führt sowohl eine zu grobe als auch eine zu detaillierte Ausarbeitung zu einem Zeitverlust: Durch vage und ungenügende Rückmeldungen bzw. durch einen zu hohen Aufwand bei der Erstellung des Prototyps. Deswegen sollte sich der Detaillierungsgrad der verschiedenen Dimensionen von Schritt 3 an folgenden Merkmalen orientieren:

- Zielgruppe (intern/ extern)
- Stadium des Entwicklungsprozesses (früh/ Mitte/ spät)
- Erstellungsgeschwindigkeit (schnell/ sorgfältig)
- Langlebigkeit (kurz/ mittel/ lang)
- Darstellung (konzeptuell/ konkret)
- Präsentationsstil (erklärend/ interaktiv)
- Medium (physisch/ digital)
- Wiedergabetreue (gering/ mittel/ hoch; siehe Schritt 3)

Schritt 5: Prototyping Methode wählen

In diesem Schritt wird bestimmt, welche Prototyping Methode (siehe Abschnitt 2.3.5) am besten zu den gegebenen Umständen passt.

Schritt 6: Prototyping Werkzeug wählen

Nun kann entschieden werden, welches Werkzeug am besten genutzt wird, um die in Schritt 5 gewählte Prototyping Methode umzusetzen. Dabei sollte unbedingt mit berücksichtigt werden, welche Fähigkeiten bei den Erstellern der Prototypen hinsichtlich der Bedienung der einzelnen Werkzeuge bereits vorhanden sind.

Phase III: Erstellung des Prototyps

Nach der Festlegung der Zielsetzung und der Wahl der Vorgehensweise erfolgt in dieser Phase die eigentliche Erstellung des Prototyps.

Schritt 7: Gestaltungsrichtlinien formulieren

Gestaltungsentscheidungen sollten stets nachvollziehbar und begründet erfolgen. Dabei können bekannte Gestaltungsrichtlinen aus den Bereichen kognitive Psychologie, Graphisches Design und Informationsdesign sehr hilfreich sein.

Schritt 8: Prototyp erstellen

In diesem Schritt fliessen Anforderungen und Gestaltungsrichtlinien zusammen. Letztendlich wird die Qualität eines Prototyps von der Qualität der Nutzerforschung, einer genauen Formulierung der Anforderungen sowie einer sorgfältigen Analyse der Gestaltungsmöglichkeiten in mehreren Iterationen bestimmt.

Phase IV: Ergebnisse

In dieser Phase wird der Prototyp Bewertungen (Reviews) und Usability Tests unterzogen. Nachdem letztmals Erkenntnisse in die Überarbeitung des Prototyps eingeflossen sind,

dient der Prototyp schliesslich zusammen mit der Spezifikation als Grundlage zur Umsetzung.

Schritt 9: Prototyp bewerten
Bevor Prototypen durch Nutzer getestet werden, sollten sie durch interne Interessensgruppen bewertet und optimiert werden.

Schritt 10: Design validieren
Prototypen sollten in jeder Iteration auch mit Hilfe von externen Interessensgruppen, vor allem den Nutzern, getestet werden.

Schritt 11: Design umsetzen
In diesem letzten Schritt wird der Prototyp so aufbereitet, dass er möglichst anschaulich und eindeutig den Entwicklern aufzeigt, wie das Endprodukt aussehen und sich verhalten soll.

Obwohl dieser Prototyping Prozess von Arnowitz et al. (2007) sicher umfassend und ausführlich formuliert ist, haben die Autoren es versäumt, den iterativen Charakter in Ihrer Vorgehensweise herauszuarbeiten. So ist auch bezeichnend, dass das Stichwort „Iteration“ nicht einmal im Schlagwortverzeichnis des Buches aufgelistet ist. Deswegen muss bei diesem Vorgehensmodell unbedingt berücksichtigt werden, dass viele Schritte oder gar Phasen bei der Erstellung eines Prototyps im Sinne der ISO 13407 Norm (siehe Seite 17) mehrfach durchlaufen werden sollten.

2.3.5 Beschreibung verschiedener Prototyping-Methoden

Das Erstellen von Prototypen ist eine Praxis, die in praktisch allen gestalterischen Fachbereichen zum Einsatz kommt. Die Methoden des Prototypings sind somit so verschieden wie die Interaktionsarten, die mit Ihrer Hilfe veranschaulicht und simuliert werden sollen. Im Folgenden werden gängige Protoyping-Methoden vorgestellt (Arnowitz et al., 2007). Andere Autoren wie Beaudouin-Lafon und Mackay (2003) verwenden eine leicht andere Kategorisierung, die sich aber nicht grundlegend von der von Arnowitz et al. unterscheidet und sich leicht in deren überführen lässt. Die Reihenfolge der nachfolgenden Vorstellung folgt grob der chronologischen Abfolge des Einsatzes der einzelnen Methoden im Laufe einer Softwareentwicklung.

Card Sorting

Card Sorting dient dazu, eine grosse Anzahl an Funktionen oder Objekten (z.B. Bücher) nutzerorientiert zu kategorisieren. Card Sorting wird zumeist früh im Softwareentwicklungsprozess eingesetzt und bildet die Grundlage für die Struktur und Bezeichnung (*Labeling*) der Informationsarchitektur. Konkret wird so vorgegangen, dass Nutzer vorbeschriftete Karten erhalten, die sie in inhaltlich ähnliche Stapel sortieren sollen. Schliesslich werden die Nutzer gebeten, die jeweiligen Stapel noch mit einem passenden Kategoriennamen zu versehen. Mit dieser abstrakten Methode erhält man Einblick in das Empfinden der Nutzer bezüglich der Ähnlichkeit bzw. Unterschiedlichkeit von gegebenen Inhalten, sowie in die Terminologie von Kategorien.

Storyboard Prototyping

Ein Storyboard kann man sich als illustriertes Drehbuch vorstellen, das eine Interaktion des Nutzers mit dem zu gestaltenden System visualisiert. Es dient in einer frühen Phase des Softwareentwicklungsprozesses dazu, die Aufgabenteilung zwischen Nutzer, Software und System zu beschreiben. Das Ziel und das grobe Verhalten des interaktiven Systems soll verdeutlicht werden, ohne auf Details der Gestaltung der Benutzeroberfläche einzugehen. Storyboards bestehen je nach Ausarbeitungsgrad in Textbeschreibungen oder comic-ähnlichen Zeichnungen der Interaktion zwischen Nutzer und System.

Video Prototyping

Video Prototyping kann als optisch ansprechende Weiterführung eines Storyboards verstanden werden. Mit Hilfe von Videotechnik wird das zu gestaltende System so vorgestellt, als ob es bereits existieren würde. Auf diese Weise sollen innovative Interaktionsideen vorgestellt werden, ohne das entsprechende System vorab entwickeln zu müssen. Video Prototyping dient somit in einer frühen Projektphase der Veranschaulichung und Bewertung von Ideen.

Wireframe Prototyping

Ein Wireframe (englisch für „Drahtgestell“) Prototyp wird am Anfang des konkreten Designs einer Benutzeroberfläche angefertigt. Er besteht aus einer groben Skizze, wie einzelne Ansichten der Benutzeroberfläche aussehen könnten. Auf die Ausgestaltung von Farben, Formen und Grafiken wird bewusst verzichtet. Statt deren wird mit symbolischen Platzhaltern gearbeitet. Die verschiedenen Ansichten werden auf einen konkreten Anwendungsfall (ein Use Case oder Szenario) ausgerichtet. Ein Wireframe Prototyp dient dazu, konzeptuelle Annahmen über die Produktstruktur und grundlegende Interaktionsformen zu visualisieren.

Blank Model Prototyping

Das Blank model Prototyping (in etwa: „Rohlings-Prototyp“) weist im Vergleich zu allen anderen Methoden zwei wesentliche Unterschiede auf. Zum einen ist es nur für haptische, dreidimensionale Bedienelemente geeignet. Zum anderen wird der Prototyp nicht im Vorfeld durch Experten erstellt, sondern während einer Zusammenkunft mit Nutzer durch diese. Nutzer erhalten typischerweise die Aufgabe, aus Bau- und Bastelmaterial (wie Papier und Knetmasse) ein Prototyp eines Bedienelementes zu erstellen. Dazu werden der Nutzungskontext und eine konkrete Angabe der gewünschten Interaktionsform vorgegeben. Der Vergleich der von verschiedenen Teilnehmern erstellten Bedienelemente (und die mündlichen Erläuterungen, während sie erstellt wurden) gibt Aufschluss über geteilte Auffassungen bezüglich grundlegenden Formen und Eigenschaften der Bedienelemente. Blank model Prototyping kommt in frühen Projektphasen zum Einsatz und bringt in der Regel grobe Entwürfe hervor.

Paper Prototyping

Ein Papier-Prototyp definiert sich über das Medium, auf dem er präsentiert wird: Papier. Ein Papier-Prototyp ist in aller Regel interaktiv, d.h. sämtliche Systemreaktionen werden durch Hilfsmittel auf Papier simuliert. So kann ein Dropdown durch ein kleines Post-it, oder ein Seitenaufruf durch die Überdeckung des bisherigen Bildschirmes durch einen neuen dargestellt werden. Papier-Prototypen werden in erster Linie für Nutzertests angefertigt – durch die Interaktivität dieser Prototyp-Methode kann man reichhaltige Erkenntnisse über geeignete Strukturen und Verhaltensweisen des interaktiven Systems sammeln. Die besondere Stärke liegt im vergleichsweise geringen Aufwand bei der Erstellung und Überarbeitung dieser Prototyp-Form.

Wizard-of-Oz Prototyping

Beim Wizard-of-Oz Ansatz werden mangelhafte oder noch gar nicht existierende Fähigkeiten eines interaktiven Systems durch einen Mensch übernommen, der bei einer Testsitzung hinter den Kulissen agiert. Auf diese Weise können Fähigkeiten bestimmter Technologien (z.B. Sprachsteuerung) simuliert werden, ohne auf ein voll funktionsfähiges System angewiesen zu sein. Der Wizard-of-Oz Ansatz kommt genau genommen nie alleine zum Einsatz, sondern immer im Rahmen von Prototype Tests weiterer Benutzeroberflächen.

Digital Prototyping

Digitale Prototypen werden zur Präsentation am Bildschirm mit herkömmlichen Büro-Anwendungen wie Photoshop, Word, Excel oder PowerPoint erstellt. Der Ausarbeitungsgrad reicht dabei von einer Serie von groben, noch durch mündliche Erklärungen zu ergänzenden Bildschirmentwürfen bis hin zu einem ausgereiften Design mit einer begrenzten Interaktivität. In der Regel müssen im Rahmen von Nutzertests digitale Prototypen noch von einem Testleiter vorgestellt werden, da es ihnen für eine freie Exploration noch an programmierter Funktionalität mangelt.

Coded Prototyping

Ein ausprogrammierter Prototyp wird mit einer Programmier- oder Skriptsprache erstellt und kann von Testnutzern interaktiv bedient werden. Ein ausprogrammierter Prototyp verwendet oftmals die Programmiersprache, die auch für die spätere Umsetzung vorgesehen ist. Der Code (oder zumindest Teile davon) kann somit die Grundlage für die Erstellung des späteren produktiven Systems bilden. Ausprogrammierte Prototypen werden spät im Entwicklungsprozess erstellt und weisen eine hohe Wiedergabetreue auf. Da die Erstellung und Änderung dieser Prototypen sehr aufwändig ist, sollten im Vorfeld alle wichtigen Designentscheide gefällt worden sein. Ausprogrammierte Prototypen können sowohl zur Prüfung der technischen Machbarkeit als auch für vollwertige Usability-Tests eingesetzt werden.

Wie aufgezeigt gibt eine Vielfalt an Prototyping Methoden, die jeweils ihre eigenen Merkmale aufweisen. Arnowitz et al. haben die vorgestellten Prototyping Methoden hinsichtlich ihrer Eignung für bestimmte Anwendungsfälle verglichen (siehe Tabelle 3 auf Seite 27). Wenn auch die Bewertungskriterien der Autoren für diese Einschätzung nicht transparent sind, gibt diese Expertensicht wertvolle Hinweise darauf, welche Prototyping Methoden sich in ihren Augen für welche Fälle eignen und für welche nicht.

2.3.6 Wiedergabetreue von Prototypen

Die Merkmale von Prototypen werden unabhängig von der eingesetzten Prototyping-Methode oftmals nach ihrer Wiedergabetreue (englisch „Fidelity“) klassifiziert. Die Wiedergabetreue beschreibt nach Walker, Takayama und Landay (2002), „how easily prototypes can be distinguished from the final product and can be manipulated to emphasize aspects of the design.“ (S.1). Die Wiedergabetreue eines Prototyps lässt sich demnach an seinem Aussehen und seinem Verhalten ablesen.

Diese Dimensionen der Wiedergabetreue werden allerdings nicht einheitlich beschrieben. Walker et al. (2002) nennen die Merkmale „interaction style, visual appearance and/or level of detail“ (S.1). Beaudouin-Lafon und Mackay (2003) klassifizieren auf vier Dimensionen die Wiedergabetreue und die Methode eines Prototyps: „Representation“ (Medium des Prototyps), „Precision“ (Detailgenauigkeit), „Interactivity“ (Interaktionsmöglichkeiten) und „Evolution“ (Lebenszyklus und Zweck des Prototyps).

Arnowitz et al. (2007) beschreiben hingegen die Eigenschaften eines Prototyps sogar in sieben Dimensionen (vgl. Seite 20):

- Informationsdesign
- Interaktionsdesign
- Navigationsmodell
- Visuelles Design
- Text- und Dateninhalte
- Markenidentität (Branding)
- Systemleistung und –verhalten

Über diese unmittelbaren Eigenschaften eines Prototyps hinaus führen sie eine Kategorisierung seines Einsatzzweckes ein (vgl. S. 20):

- Zielgruppe (intern/ extern)
- Stadium des Entwicklungsprozesses (früh/ Mitte/ spät)
- Erstellungsgeschwindigkeit (schnell/ sorgfältig)
- Langlebigkeit (kurz/ mittel/ lang)
- Darstellung (konzeptuell/ konkret)
- Präsentationsstil (erklärend/ interaktiv)
- Medium (physisch/ digital)

Diese Übersicht verdeutlicht zwei Ungereimtheiten bei der Klassifikation der Wiedergabetreue von Prototypen:

Zum einen wird nicht konsequent zwischen den Eigenschaften des Prototyps an sich und dem Zweck bzw. den Rahmenbedingungen seines Einsatzes unterschieden. Während Arnowitz et al. (2007) diese beiden Aspekte getrennt aufführen, beziehen sich die vier Dimensionen von Beaudouin-Lafon und Mackay explizit auf beide Aspekte. Die Absicht der konzeptionellen Trennung dieser beiden Aspekte ist leicht nachvollziehen. Dennoch betonen Arnowitz et al. zu recht, dass die Wiedergabetreue des Prototyps von seinem Einsatzzweck abgeleitet werden sollte. Demnach sind die beiden Aspekte der Wiedergabetreue und des Einsatzzwecks eng voneinander abhängig, auch wenn sie zur besseren Übersichtlichkeit getrennt aufgeführt werden sollten.

Zum anderen werden die Dimensionen der Wiedergabetreue von verschiedenen Autoren unterschiedlich klassifiziert. Dennoch lässt sich feststellen, dass die beiden Merkmale „Aussehen“ und „Verhalten“ eines Prototyps zu seinen Kerneigenschaften zählen. Sie kehren in den verschiedenen Beschreibungen der Dimensionen stets wieder, wenn sie auch mit unterschiedlichen Begriffen bezeichnet werden.

2.3.7 Prototyping Werkzeuge

Die Auswahl einer Prototyping Methode bestimmt das physische Medium und das Erscheinungsbild des Prototyps. Wie beim Vorgehensmodell der Erstellung von Prototypen auf Seite 20 dargestellt wurde, können Prototypen Methoden jeweils mit unterschiedlichen Werkzeugen (Tools) umgesetzt werden. Somit sind die Prototyping Werkzeuge nur bedingt von der Prototyping Methode abhängig.

Nachfolgend werden gängige Prototyping Werkzeuge vorgestellt. Strukturell ähnliche werden dabei zu Gruppen zusammengefasst.

Papier und Bleistift

Der Ausdruck „Papier und Bleistift“ steht stellvertretend für alle Büromaterialien, die zur Herstellung eines Papier-Prototypen verwendet werden. Im einfachsten Fall sind dies wortwörtlich Papier und Bleistift. In der Regel werden jedoch Stifte in verschiedenen Arten, Farben und Breiten, Haftnotizen, transparente Folien, Scheren, Klebestifte, Klebebänder und andere Hilfsmittel eingesetzt.

Bildschirmpräsentations-Programme

Die prominentesten Vertreter dieser Gruppe sind PowerPoint und Keynote. Die Präsentationssoftware PowerPoint ist Teil des Microsoft Office Paketes. Sie ist mit über 250 Millionen Installationen weltweit das am weitesten verbreitete Präsentationsprogramm (Wikipedia, 2008). Obwohl es ursprünglich nicht als Prototyping Werkzeug konzipiert war, hat es als solches eine weite Verbreitung gefunden. Dies liegt zum einen an der hohen Vertrautheit vieler Nutzer mit diesem Programm, zum anderen an seiner Eigenschaft, dass man mit ihm vergleichsweise einfach Text- und Grafikelemente erstellen und anordnen kann.

Programme zur Erstellung von Diagrammen

Zu dieser Gruppe zählen Visualisierungs-Programme wie Microsoft Visio (für Windows) und OmniGraffle (für Mac OS). Sie wurden ursprünglich für Zeichnungen und Flussdiagramme konzipiert, werden aber vergleichbar mit den genannten Programmen für Bildschirmpräsentationen gerne zur Erstellung von Prototypen genutzt.

Programme zur Erstellung von Wireframes

Zu diesen oftmals browserbasierten Angeboten zählen Anwendungen wie Balsamiq, MockupScreens, Protoshare und OverSite. Sie fokussieren auf die (grobe) grafische Erstellung von Benutzerschnittstellen und sind nur begrenzt in der Lage, Interaktivität zu simulieren.

Grafikprogramme

Diese Programme sind auf die Erstellung von Grafiken spezialisiert. Zu ihnen zählen Adobe Photoshop, Adobe Fireworks und Adobe Illustrator. Auch sie sind nicht bis wenig in der Lage, interaktive Bildschirmelemente zu simulieren.

Prototyping-Tools

Auf dem Softwaremarkt sind zahlreiche auf die Erstellung von Prototypen spezialisierte Programme erhältlich. Axure RP Pro, iRise Studio und Serena Prototype Composer sind Vertreter dieser Gruppe. Axure RP Pro ist dabei das weitverbreitetste „professionelle“ Werkzeug für die rasche Erstellung von Wireframes, Prototypen und Spezifikationen für Anwendungen und Internetseiten (Warfel, 2009).

HTML Editoren

HTML-Editoren wie Adobe Dreamweaver und Microsoft FrontPage ermöglichen die Erstellung von interaktiven Prototypen. Da sie lauffähigen HTML Code produzieren, können Prototypen, die mit HTML Editoren erstellt wurden, unter Umständen sogar in das finale Produkt überführt werden.

Programme zur Erstellung von Benutzeroberflächen

Programme wie Adobe Flash, Microsoft Expression Blend und VisualBasic sind darauf spezialisiert, interaktive Benutzeroberflächen herzustellen. Viele erlauben es zudem, multimediale Inhalte zu erstellen, beziehungsweise bereits erstellte Inhalte in die Benutzer-oberfläche zu integrieren.

Programmiersprachen

„Vollwertige“ Markup-, Script- und Programmiersprachen wie HTML, JavaScript, Java oder Ruby werden üblicherweise in der Entwicklung des produktiven Systems eingesetzt. Es ist allerdings auch möglich, sie zur Entwicklung von Prototypen, d.h. Vorstufen des finalen Systems, zu verwenden. Je nach Zweck und verfügbare Mittel kann der Prototyp in Bezug auf Interaktivität und Designtreue dem finalen Produkt beliebig nahe kommen. Wie bei HTML Editoren lässt sich der Programmcode von Prototypen oftmals in den des produktiven Systems überführen, so dass dieser nicht mehr von Grund auf neu erstellt werden muss. Im Zusammenhang mit einem HTML Prototypen wird auch oft von "Rapid Prototyping" gesprochen, da der Prototyp laufend bis zum Endprodukt verfeinert wird.

Nach dieser Übersicht stellt sich die Frage, welche Verbreitung die einzelnen Gruppen von Prototyping Werkzeugen finden.

Warfel (2009) führte 2008 eine Umfage unter nahezu 200 Berufstätigen aus dem User Experience Feld durch. Dabei wurde auch die Frage gestellt, welche Prototyping Werkzeuge in der Praxis verwendet werden. Mit 81% der Befragten waren Papier-Prototypen am weitesten verbreitet. An zweiter Stelle fand sich manuell erstellter HTML

Code (58%), gefolgt von automatisch generiertem HTML Code von Programmen wie beispielsweise Axure, iRise, Visio oder Fireworks (34%).

In einer älteren Umfrage aus dem Jahr 2002 (Guuui, 2008) waren HTML-Editoren wie Microsoft FrontPage mit 28% am weitesten verbreitet, gefolgt von konventionellen Programmen zur Erstellung von Diagrammen (25%) wie Microsoft Visio. Papier-Prototypen fanden sich hier mit nur 8% Verbreitung auf den hinteren Rängen.

Umfagen wie diese sind allerdings von zweifelhafter Repräsentativität und haben damit nur sehr begrenzte Aussagekraft: Weder sind die Teilnehmer genau beschrieben, noch wurden sie systematisch ausgewählt. Die verwendeten Werkzeuge dürften sich somit je nach Ausbildungshintergrund und beruflicher Funktion der Anwender erheblich unterscheiden.

In diesem Unterkapitel erläuterten wir, was Prototypen sind und wozu sie dienen. Im nachfolgenden Abschnitt stellen wir vor, wie sie im Rahmen von Prototyping Tests zur Evaluation von Lösungsansätzen eingesetzt werden können und geben eine Übersicht über die verwendeten Methoden.

2.4 Prototyping Tests

2.4.1 Zweck und Ziele der Evaluation von Prototypen

Wie in Kapitel 2.3.1 bereits erwähnt, können die erstellten Prototypen verwendet werden, um den Gestaltungsentwurf daraufhin zu prüfen, ob er die Anforderungen erfüllt.

Nach Sarodnick & Brau (2006) gilt die ISO Norm 9241 *Ergonomische Anforderungen für Bürotätigkeiten an Bildschirmgeräten* "als die massgebliche Norm für eine Gestaltung von Systemen mit hoher Usability und setzt Massstäbe für die Evaluation von Dialogsystemen." Teil 110 (*Grundsätze der Dialoggestaltung*) der Norm listet 7 Grundsätze, die eingehalten werden sollten: Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit und Lernförderlichkeit.

Mit der Evaluation von Prototypen kann also neben der Erfüllung von lösungsspezifischen Geschäfts- und Nutzeranforderungen auch die Erfüllung von allgemein gültigen Nutzeranforderungen geprüft werden.

Da der Gestaltungsansatz des User Centered Design ein iteratives Vorgehen verlangt, wird nicht nur die fertige Lösung abschliessend geprüft (summative Evaluation). Die Evaluation wird heute üblicherweise zur Bewertung während der Entwicklung und zur Sammlung von Informationen eingesetzt (formativ). Sie kann also auch zu weiteren Iterationen und Verbesserungen der Gestaltungslösung führen.

2.4.2 Übersicht der Evaluationsmethoden

Die verschiedenen Prototypen unterscheiden sich in ihren Merkmalen, beispielsweise an welche Zielgruppe sie sich wenden, oder in welcher Entwicklungsphase sie eingesetzt werden (vgl. Seite 19). Zur Erstellung werden daher unterschiedliche Prototyping Tools eingesetzt. Entsprechend sind die resultierenden Prototypen von völlig unterschiedlicher Art: Begriffsgruppen, Handskizzen oder Webseiten.

Die folgende Übersicht (Tabelle 3) nach Arnowitz et al. (2007) zeigt die Merkmale von Prototyping Tools.

	Card Sorting	Wireframe	Story-board	Papier	Video	Wizard of Oz	Digital Interaktiv	auspro-gram-miert
Zielgruppe	intern / extern	intern	intern / extern	intern/ extern	intern/ extern	extern	intern / extern	Intern / extern
Stadium des Entwicklungs-prozesses	früh	früh / mittel	früh / mittel	mittel	früh / mittel	mittel	mittel / spät	spät
Erstellungs-geschwindig-keit	schnell	schnell	schnell	schnell	schnell / sorgfältig	schnell	schnell / sorgfältig	sorgfältig
Langlebigkeit	kurz	kurz / mittel	mittel	mittel	mittel / lang	mittel	mittel / lang	lang
Darstellung	konzep-tuell	konzep-tuell	konzep-tuell	konzep-tuell/ konkret	konzep-tuell/ konkret	konzep-tuell/ konkret	konzep-tuell/ konkret	konkret
Präsentations-stil	interaktiv	erklärend	erklärend	interaktiv	erklärend / interaktiv	interaktiv	interaktiv	interaktiv
Medium	physisch / digital	physisch/ digital	physisch/ digital	physisch	digital	digital	digital	digital
Wiedergabe-treue	gering	gering/ mittel	mittel/ hoch	mittel	mittel/ hoch	mittel/ hoch	mittel/ hoch	hoch

Tabelle 3: Merkmale von Prototyping Tools

Legende:

Zielgruppe intern: Entwickler, Management, Marketing, Verkauf, Technische Autoren, etc.
extern: Endnutzer, Kunden, Finanzielle Stakeholder, Domain Spezialisten/Analysten

Darstellung konzeptuell: Aussehen entspricht nicht dem Produkt. Schwerpunkt liegt auf der Darstellung der Ideen/des Konzeptes
konkret: Aussehen entspricht dem Produkt. Visuelle Wiedergabetreue kann tief, mittel oder hoch sein.

Entsprechend eignen sich je nach Tool, das zur Erstellung eingesetzt wurden, andere Evaluationsmethoden zur Prüfung des resultierenden Prototypen. Tabelle 4 zeigt die Eignung nach Arnowitz et al. (2007)

	Card Sorting	Wire-frame	Story-board	Papier	Video	Wizard of Oz	Digital Inter-aktiv	auspro-gram-miert
Interne Reviews	++	++	++	--	++	++	+	++
Externe Reviews	n/a	-	++	n/a	++	+	++	++
Usability Test	n/a	n/a	n/a	++	n/a	++	++	++
Fokus Gruppen Test	+	-	+	+	+	+	++	++
Experten Review	+	++	++	+	++	++	++	++
Kunden-feedback	n/a	--	+	+	+	+	+	++
Umfrage	n/a	n/a	n/a	n/a	n/a	n/a	+	++
Kognitiver Walk-through	n/a	+	n/a	+	n/a	-	+	++

Tabelle 4: Eignung Evaluationsmethode nach Prototyping Tools

Legende:
++ sehr geeignet
+ geeignet
- akzeptabel
-- nicht geeignet
n/a nicht anwendbar

Generell unterscheidet man zwischen Methoden, welche von Experten durchgeführt werden und denen, welche Nutzer einbeziehen (Heuer in Heinsen & Vogt, 2003). Letztere werden dabei empirische Methoden genannt, da die Erkenntnisse aus Beobachtungen/Experimenten stammen. Die Versuchspersonen sind dabei echte Nutzer oder zumindest repräsentativ. Sie verfügen in der Regel über keine speziellen Kenntnisse

der Usability und geben daher keine direkten Bewertungen ab, sondern führen vordefinierte Aufgaben mittels der zu evaluierenden Software bzw. dessen Prototypen aus.

Bewertungen von Experten werden meistens Inspektionsmethoden genannt (Heuer in Heinsen & Vogt, 2003; Sarodnik & Brau, 2006). Nielsen (1994) verwendet Inspektionsmethoden als Dachbegriff und bezeichnet Expertenbewertungen als heuristische Methoden. Gemeinsam ist allen Varianten, dass einer oder mehrere Evaluatoren anhand ihrer Erfahrung und mittels vorgegebener Richtlinien die Benutzeroberfläche der Software bzw. des Prototyps beurteilen.

Nach Sarodnik & Brau (2006) gehören die Heuristische Evaluation (Nielsen, 1993) und der Kognitive Walkthrough zu den bekanntesten Inspektionsmethoden. Neben den reinen Inspektionsmethoden und den reinen Nutzertests gibt es auch Mischformen.

Nachfolgend werden einige Beispiele genauer beschrieben.

Heuristische Evaluation

Inspektionsmethode. Spezialisten beurteilen die Benutzeroberfläche checklistenartig aufgrund von Heuristiken, dh. allgemein akzeptierten Prinzipien, meist basierend auf Erfahrungswerten oder Gestaltungsrichtlinien.

Zielgruppe: Experten. Intern oder extern.

Stadium des Entwicklungsprozesses: beliebige Projektphase.

Aufwand: Relativ kostengünstig, verhältnismässig geringer zeitlicher Aufwand.

Zu beachten: Es muss sich bei den Durchführenden wirklich um Experten handeln, idealerweise mit Erfahrung. Es ist eher unwahrscheinlich, dass sich ein Experte auch in der Problem Domain auskennt. Experten decken meist andere Probleme auf als Benutzer. Unterschiedliche Experten finden meist unterschiedliche Probleme, aber je mehr Experten man einsetzt, desto eher gehen die Vorteile des geringen Aufwandes verloren.

Kognitiver Walkthrough

Inspektionsmethode. Experte versucht, aus dem Blickwinkel des Benutzers, eine Aufgabe, bzw. die zur Lösung nötigen Abläufe zu analysieren.

Zielgruppe: Experten. Intern oder extern

Stadium des Entwicklungsprozesses: frühe Projektphase, während des Designs, auch mit frühen Prototypen von geringer Wiedergabetreue.

Aufwand: materiell geringer bis mässiger Aufwand (je nach Prototyp keine zusätzliche Infrastruktur nötig), zeitlich geringer bis mittlerer Aufwand.

Zu beachten: Diese Methode ist vor allem geeignet, um zu erfahren, wie einfach erlernbar ein Interface ist. Für Interfaces, welche eine intensive vorgängige Schulung erfordern, nicht geeignet.

Pluralistischer Walkthrough

Inspektionsmethode/Mischform. Eine gemischte Gruppe (Usability Experten, Benutzer, Entwickler, sonstige Stakeholders) geht ein Aufgabenszenario aus der Sicht des Benutzers durch, während die Gruppenmitglieder die einzelnen Schritte für sich notieren, dann miteinander diskutieren und kommentieren.

Zielgruppe: Experten, Benutzer, Entwickler, weitere Stakeholder. Primär Intern

Stadium des Entwicklungsprozesses: frühe Projektphase, während des Designs, mit frühen Prototypen.

Aufwand: personell mässiger Aufwand, zeitlich mittlerer bis grosser Aufwand

Zu beachten: Der zeitliche Aufwand orientiert sich immer an der langsamsten Person, da mit der Diskussion zugewartet wird, bis alle für sich die Schritte notiert haben.

Usability Walkthrough

Benutzer versuchen, ihnen gestellte Aufgaben ohne vorgängiges formelles Training zu lösen. Während des Walkthroughs kommentieren sie ihre Handlungen, Absichten und Erwartungen. Dabei werden sie vom Versuchsleiter(team) begleitet, das ihnen klärende Fragen zu ihren Handlungen und Kommentaren stellen kann.

Zielgruppe: idealerweise echte Benutzer, auf keinen Fall Mitglieder des Entwicklungsteams. Intern oder extern

Stadium des Entwicklungsprozesses: frühe Projektphase, während des Designs, auch mit frühen Prototypen von geringer Wiedergabetreue.

Aufwand: materiell geringer bis mässiger Aufwand (je nach Prototyp keine zusätzliche Infrastruktur nötig), zeitlich mittlerer bis grosser Aufwand

Zu beachten: Diese Methode ist vor allem geeignet, um zu erfahren, wie einfach erlernbar ein Interface ist. Für Interfaces, welche eine intensive vorgängie Schulung erfordern, nicht geeignet.
Gedankengänge werden ausformuliert und angesprochen, dadurch werden auch Motivationen und Überlegungen offengelegt.
Keine Effizienzmessung möglich, mögliche Beeinflussung der Testpersonen durch die Versuchsleiter

(Formeller) Usability Test

Benutzer versuchen, ihnen gestellte Aufgaben ohne vorgängiges formelles Training und ohne Hilfestellung seitens der Versuchsleitung zu lösen. Sie lernen das Interface, während sie Aufgaben damit zu lösen versuchen. Wenn keine Effizienzmessung gewünscht ist, kommentieren sie während des Tests ihre Handlungen, Absichten und Erwartungen. Versuchsleiter sollten möglichst nicht eingreifen und beschränken sich auf die Beobachtung.

Zielgruppe: idealerweise echte Benutzer, oder solche, die den echten Benutzern in testrelevanten Kriterien entsprechen. (z.B. Computererfahrung, Kenntnisse der Problem Domain, Alter)

Stadium des Entwicklungsprozesses: funktionierendes System oder Prototypen hoher Wiedergabetreue in einer späten Projektphase

Aufwand: Sehr aufwändig in Infrastruktur (Labor, ev. mit getrennten Räumen für Testbenutzer und Beobachter, oder portable Einrichtungen mit Kamera) und Zeit- und Kostenaufwand (Rekrutierung, Beobachtung, Nachbearbeitung)

Zu beachten: Da diese Usability Test sehr spät in der Projektphase stattfinden, wird es sehr teuer, die Erkenntnisse in einer verbesserten Benutzeroberfläche umzusetzen. Ev. ist dies sogar erst für einen späteren Release möglich. Aufgrund von beobachteten Handlungen werden Rückschlüsse auf die Motivation, die Absichten und die Gedankengänge gezogen, die nicht notwendigerweise zutreffen.

2.4.3 Reliabilität und Validität der Untersuchung

Generell stellt sich bei Usability Evaluationen die Frage nach der Reliabilität, da sich die Testpersonen naturgemäss enorm unterscheiden und dadurch nicht gegeben ist, dass zwei Testpersonen dasselbe Resultat erzielen. Reliabilität wird hier meist durch eine entsprechende Anzahl Testpersonen erreicht. (vgl. auch Kapitel 2.4.4)

Bezüglich der Validität stellt sich die Frage, ob die Evaluation von Prototypen geringer und hoher Wiedergabetreue dasselbe Ergebnis erzielen. In der Literatur ist man sich generell einig, dass Prototypen visuell geringer Wiedergabetreue und solche visuell hoher Wiedergabetreue in unterschiedlichen Projektphasen zu unterschiedlichen Zwecken eingesetzt werden sollen. (Rudd, Stern, Isensee, 1996; Snyder, 2003; Arnowitz et al., 2007) Daraus könnte geschlossen werden, dass eine Evaluation auch nicht zu demselben Ergebnis führen würde. Dies würde jedoch bedeuten, dass sie nicht dasselbe evaluieren, die Ergebnisse also nicht vergleichbar wären.

Für Usability Evaluationen gibt es jedoch verschiedene Untersuchungen, die darauf hinweisen, dass sowohl grobe wie auch detailgetreue Prototypen zu denselben Ergebnissen führen im Hinblick auf die Erkennung von Usability Problemen.

Virzi, Sokolov & Karis (1996) stellen fest: „[...] we found substantially the same set of usability problems in both the low- and high-fidelity groups.“ (S. 241). Grundsätzlich zur selben Aussage, dass nämlich die Resultate von Papier-Prototypen und elektronischen Prototypen nicht wesentlich voneinander abweichen, kommen auch Liu & Khoshabee (2003) und Mäuselein (2007).

Da Prototypen sowohl hoher wie auch tiefer visueller Wiedergabetreue zu denselben Ergebnissen führen, bedeutet dies für unsere Arbeit, dass verschiedene Prototypen-Arten vergleichbar sind und ein direkter Vergleich zu einem validen Ergebnis führen kann.

2.4.4 Anzahl Versuchspersonen

Gemäss Nielsen und Landauer (1993) entdeckt man mit 5-6 Testteilnehmern bis zu 80% der grössten Probleme. Dies natürlich unter der Voraussetzung, dass sich an Testanordnung und Aufgabenstellung nichts ändert. Abbildung 8 zeigt, dass mit zunehmender Zahl von Testpersonen (schwarze Punkte) oder Usability Experten (weisse Punkte) nicht immer mehr Usability Probleme gefunden werden.

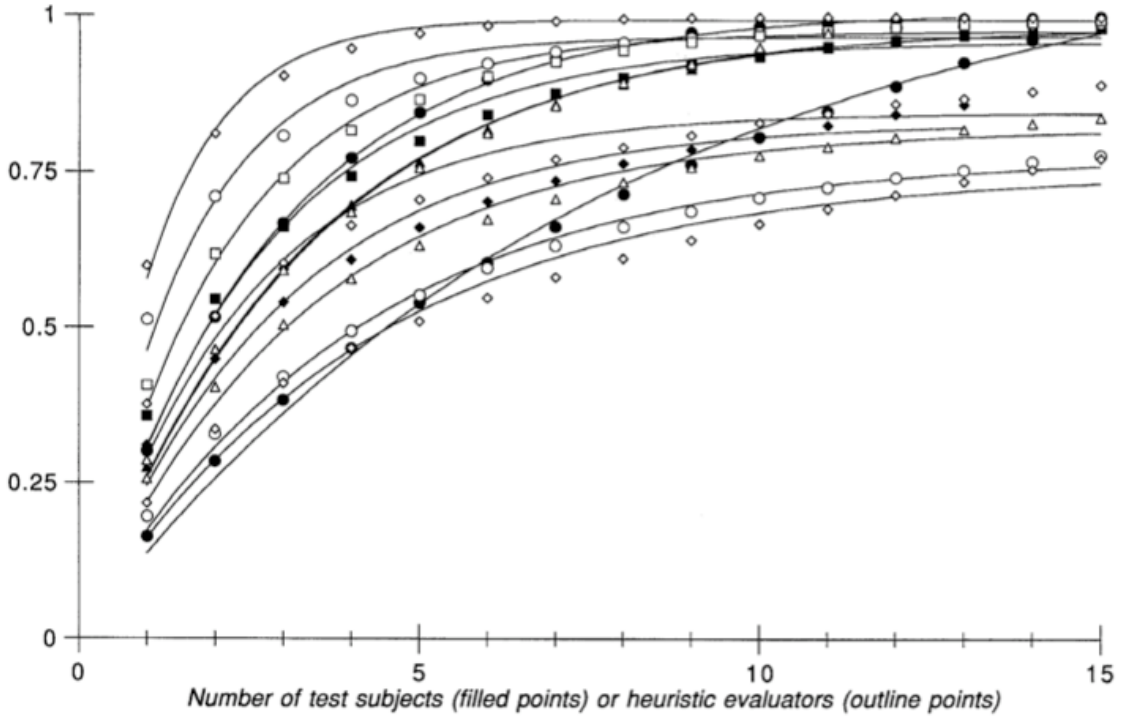

Abbildung 8: Anzahl Versuchspersonen, Nielsen & Landauer (1993)

Neuere Untersuchungen widersprechen der Aussage, dass 5-6 Versuchspersonen genügen. Nach Molich et al. (1999) haben 9 unabhängige Teams eine Usability Evaluation durchgeführt, wobei keine zwei Teams dieselben Probleme aufgedeckt haben. Ebenso haben Spool & Schroeder (2001) festgestellt, dass bei einem Usability Test mit 18 Benutzern jeder Benutzer grosse Fehler entdeckt hat, die sich von den Entdeckungen der anderen unterschieden.

Die Frage ist in der Literatur nicht abschliessend beantwortet und wird immer noch heftig diskutiert. (Bevan et al., 2003) Nicht zuletzt aus Gründen der Praktikabilität (Durchführbarkeit, Kosten) stützen sich nach wie vor viele Tests auf die Ergebnisse von Nielsen. Nielsen (1993) räumt ein, dass die Resultate nicht vollständig zuverlässig sind, meint aber, dass "this level of accuracy might be enough for many projects." (S. 169) und verweist immer wieder auf Kosteneffektivität (ebda.).

3 Methodik der empirischen Untersuchung

In diesem Kapitel beschreiben wir, welche Forschungsfragen wir uns stellten (3.1), welchen Untersuchungsgegenstand wir gewählt haben (3.2), wie die Prototypen erstellt wurden (3.3) und mit welchem Untersuchungsdesign wir Antworten auf sie suchten (3.4).

3.1 Forschungsfragen und Hypothesen

Unsere Forschungsfrage lautete: Wie gut eignen sich verschiedene Prototyping-Methoden, um Interaktionsformen von RIA Patterns zu simulieren?

Um diese Frage zu beantworten, setzten wir wie beschrieben zehn RIA Patterns in vier verschiedene Prototypen-Versionen um und führten mit diesen Prototyping Tests durch.

Im nachfolgenden Abschnitten stellen wir unsere Prädiktor- und Zielvariablen vor sowie die Hypothesen, die wir mit ihnen untersucht haben.

3.1.1 Unabhängige Variable

Unserer Studie lag eine einzige Prädiktorvariable (siehe Glossar) zugrunde: Die Prototyp-Version. Wir untersuchten vier Ausprägungen (siehe Seite 38):

1. Papier-Prototyp, erstellt von Hand
2. Papier-Prototyp, erstellt mit PowerPoint
3. Digitaler Prototyp, erstellt mit Axure RP Pro
4. Kodierter Prototyp, erstellt mit Ajax

3.1.2 Abhängige Variablen

Die Auswirkungen der Prädiktorvariablen haben wir anhand von acht Zielvariablen (siehe Glossar) gemessen.

Vier davon bestanden in den folgenden, quantifizierten Beobachtungen des „objektiven Fragebogens“ (siehe Seite 59):

1. Zögern des Testteilnehmers
2. Schwierigkeiten des Testteilnehmers
3. Langsamkeit des Testteilnehmers
4. Zurechtkommen des Testteilnehmers insgesamt

Die anderen vier Zielvariablen bestanden in den Bewertungen der Testteilnehmer, die sie mittels des „subjektiven Fragebogens“ (siehe Seite 60) ausdrücken konnten:

5. Grad, in dem sie der Prototyp bei der Erledigung der Aufgabe störte
6. Aussehen der Bildschirmelemente
7. Verhalten der Bildschirmelemente
8. Schnelligkeit des Prototyps

3.1.3 Forschungshypothesen

Wie in Kapitel 2.4.3 dargestellt spricht der aktuelle Forschungsstand für eine Gleichwertigkeit verschiedener Prototyping Methoden und unterschiedlich hoher

Ausprägungen an Wiedergabetreue. Von dieser Ausgangslage haben wir unsere Forschungshypothesen abgeleitet. Gemäss der genannten Gleichwertigkeit sind die meisten unserer Hypothesen Nullhypothesen, d.h. wir gehen davon aus, im Vergleich der vier Prototypen-Versionen keine bedeutsamen Unterschiede der Zielvariablen festzustellen. Signifikante Unterschiede erwarten wir nur in zwei nahe liegenden und leicht nachvollziehbaren Punkten:

- Testteilnehmer empfinden, dass der per Hand erstellte Papierprotoyp weniger wie ein fertiges System aussieht als die anderen drei Prototypen-Versionen. (Hypothese 6)
- Testteilnehmer empfinden die beiden Papier-Prototypen langsamer als die beiden elektronischen Prototypen. (Hypothese 8)

Zudem sagen wir voraus, dass sich Aussehen, Verhalten und Geschwindigkeit eines Prototypen darauf auswirken, wie störend er von Testteilnehmern empfunden wird (Hypothese 9).

Nachfolgende die Hypothesen im Überblick.

Hypothese 1 (Nullhypothese)	Bei der Aufgabenbearbeitung zögern Testteilnehmer bei den vier Prototypen-Versionen gleich stark.
Hypothese 2 (Nullhypothese)	Testteilnehmer bearbeiten die Testaufgaben mit den vier Prototypen-Versionen gleich schnell.
Hypothese 3 (Nullhypothese)	Testteilnehmer haben bei der Bearbeitung der Testaufgaben mit den vier Prototypen-Versionen gleich viele Schwierigkeiten.
Hypothese 4 (Nullhypothese)	Testteilnehmer kommen bei der Bearbeitung der Testaufgaben mit den vier Prototypen-Versionen gleich gut zurecht.
Hypothese 5 (Nullhypothese)	Testteilnehmer fühlen sich bei der Bearbeitung der Testaufgaben durch die vier Prototypen-Versionen gleich stark gestört.
Hypothese 6 (Alternativhypothese)	Testteilnehmer empfinden, dass der per Hand erstellte Papierprotoyp weniger wie ein fertiges System aussieht als die anderen drei Prototypen-Versionen.
Hypothese 7 (Nullhypothese)	Testteilnehmer empfinden das Verhalten der vier Prototypen-Versionen als gleich nahe zu dem des späteren Systems.
Hypothese 8 (Alternativhypothese)	Testteilnehmer empfinden die beiden Papier-Prototypen langsamer als die beiden elektronischen Prototypen.
Hypothese 9 (Zusammenhangs-hypothese)	Es besteht ein Zusammenhang zwischen dem Ausmass, in dem sich Testteilnehmer an einer Prototyp-Version stören und ihrem Aussehen, Verhalten und Geschwindigkeit.

Tabelle 5: Hypothesen

Von einer statistischen Auswertung der quantitativen Messwerte auf Ebene der RIA Patterns haben wir abgesehen, da die Kombination von Prototyp-Version und RIA nur noch einen Stichprobenumfang von jeweils n=3 aufwies. Somit wurden die Auswirkungen der verschiedenen Prototyping-Methoden immer nur auf die Gesamtheit aller zehn RIA Patterns betrachtet, und nicht auf bestimmte RIA Patterns. Die Stärken und Schwächen einzelner RIA Patterns wurden vielmehr durch die qualitativen Beobachtungen abgedeckt.

3.2 Untersuchungsgegenstand

Dieses Unterkapitel beschreibt, welche RIA Patterns wir in Prototypen simuliert haben. Zudem gehen wir auf die Prototyping Methoden ein und die Werkzeuge, die bei der Erstellung der Prototypen zum Einsatz kamen.

3.2.1 Auswahl der Patterns

Bei der Auswahl der Patterns die wir als Prototypen umsetzen und testen wollten haben wir uns folgende Vorgaben gesetzt:

- Wir wählen 10 verschiedene Patterns aus verschiedenen Kategorien aus
- Wir beschränken uns auf Behavioral- bzw. Action-Level Patterns (vgl. 2.2.2)
- Die Patterns besitzen RIA-Charakteristiken (gemäss Kapitel 2.2.3)
- Wir konzentrieren uns auf Patterns, bei denen die Interaktion des Benutzers im Vordergrund steht
- Die Patterns müssen in eine konkrete Benutzeraufgabe integriert werden können

Neben der Analyse der bestehenden Pattern Libraries haben wir ebenfalls viele Web 2.0 Anwendungen auf die Verwendung von RIA Patterns untersucht. Daraus entstand unser eigener RIA Pattern Katalog mit ca. 40 RIA Patterns in 9 Kategorien. Die Liste der Patterns erhebt keinen Anspruch auf Vollständigkeit.

Eine Übersicht der Patterns folgt auf den folgenden Seiten. Die von uns ausgewählten Patterns sind farblich hervorgehoben. Die detaillierte Beschreibung der Patterns inkl. Beispiele ist im Anhang ersichtlich.

Dateneingabe

Pattern	Problem
Editierbare Tabellen	Eine oder mehrere Zeilen einer Tabelle sollen selektiert werden können.
Inline Textbearbeitung	Ein Text soll direkt an der Stelle, an der er angezeigt wird, bearbeitet werden können.
Rich-Text Editor	Ein Text soll bearbeitet und formatiert werden können, ohne dass Markup eingegeben werden muss.
Eingabevervollständigung (Auto-Complete)	Der Benutzer soll eine Eingabe in einem Textfeld machen, ohne den gesamten Begriff eingeben zu müssen.
Eingabevorschläge (Type & Suggest)	Der Benutzer soll eine Eingabe in einem Textfeld machen und das System soll ihn dabei unterstützen.
Tastaturkürzel (Keyboard Shortcuts)	Aktionen sollen ohne die Benutzung der Maus aufgerufen werden können.
Rückgängig (Undo)	Vom Benutzer ausgeführte Eingaben und Aktionen sollen rückgängig gemacht werden können.

Datenauswahl

Pattern	Problem
Selektion in Tabellen	Eine oder mehrere Zeilen einer Tabelle sollen selektiert werden können.

Schieberegler (Slider)	Der Benutzer soll einen Wert innerhalb eines definierten Wertebereichs auswählen können.
Bewertung von Objekten	Der Benutzer soll ein Objekt auf einer fixen Skala bewerten können.
Statusänderung	Elemente sollen durch einen einzigen Mausklick gekennzeichnet bzw. ihr Zustand verändert werden können.

Datenaktualisierung

Pattern	Problem
Filterung von Suchergebnissen	Eine Liste von Suchergebnissen soll automatisch gefiltert werden, wenn der Benutzer die Suchkriterien ändert.
Ticker	Informationen sollen laufend und ohne Interaktion des Benutzers auf der Seite eingeblendet werden.
Periodischer Update	Veränderte Daten sollen ohne Benutzerinteraktion aktualisiert werden.

Navigation

Pattern	Problem
Pulldown Menu	Der Benutzer soll jederzeit direkten Zugang zu mehreren Ebenen der Navigation haben.
Expandierbare Bäume	Der Benutzer soll hierarchisch organisierte Navigationsstrukturen durch auf- und zuklappen von Elementen erforschen können.
Kontext Menu	Aktionen sollen im Kontext eines selektierten Objekts ausgeführt werden können.
Akkordeon	Die oberste Navigationsebene sowie die Substruktur eines Punktes sollen permanent sichtbar sein.

Bilder und Karten

Pattern	Problem
Lupe	Durch das Bewegen einer Lupe über einem Bild wird der Bildausschnitt vergrössert.
Kartenausschnitte verschieben	Der Benutzer kann den sichtbaren Ausschnitt einer Karte verändern können.
Zoomen	Ausschnitte eines graphischen Element sollen vergrössert oder verkleinert werden können.

Objektmanipulation

Pattern	Problem
Drag und Drop	Der Benutzer soll Elemente auf einer Seite durch Ziehen und Loslassen umorganisieren können.
Skalierung von Objekten	Elemente sollen durch eine direkte Manipulation in der Grösse verändert werden können.

Verschieben/Scrollen von Objekten	Elemente sollen durch eine direkte Manipulation auf dem Bildschirm bewegt werden können.

Feedback

Pattern	Problem
Fortschrittsanzeige (Progress Bar)	Der Benutzer soll über die Gesamt- und verbleibende Dauer einer Operation informiert werden.
Aktivitätsanzeige	Bei länger andauernden Operationen soll der Benutzer Feedback über den Systemstatus erhalten.
Validierung von Eingaben	Der Benutzer soll Unterstützung bei der Korrektur von Eingabefehlern erhalten.
Anzeige eingegebener/ verbleibender Zeichen	Es soll dem Benutzer laufend signalisiert werden, wieviel Zeichen in einem Textfeld noch eingegeben werden können.

Anzeige

Pattern	Problem
Detailanzeige bei Mouseover	Zusätzliche Informationen sollen beim Bewegen der Maus über ein Element dargestellt werden.
Extras on Demand	Bei Bedarf sollen erweiterte Informationen oder Formularelemente eingeblendet werden können.
Micro Link	Der Benutzer möchte Links auf der Seite verfolgen, ohne dabei durch ein Neuladen des ganzen Inhalts unterbrochen zu werden.
Live Preview	Dem Benutzer wird das Ergebnis von Eingaben in Formularfeldern sofort in einer Vorschau angezeigt.
Modaler Dialog	Es sollen Informationen in einem neuen Fenster über dem "eingefrorenen" Hauptinhalt dargestellt werden.
Popup	Informationen sollen in einem neuen Fenster über dem Hauptinhalt dargestellt werden.
Portlet	Fragmente der Seite sollen vom Benutzer beliebig kombiniert und arrangiert werden können.
Split Panels	Bereiche der Seite sollen vom Benutzer in der Grösse verändert werden können.
Inline Hilfestellung	Der Benutzer erhält bei Bedarf einen Hilfetext am Ort der Eingabe eingeblendet.

Visuelle Effekte

Pattern	Problem
Animierte Übergänge	Dem Benutzer soll eine Statusänderung eines Objektes visuell verdeutlicht werden.
Hervorheben von Objekten	Die Aufmerksamkeit des Benutzers soll auf ein bestimmtes Objekt gelenkt werden.

3.2.2 Auswahl der Prototyping Methoden und -Werkzeuge

Folgende Anforderungen leiteten uns bei der konkreten Auswahl der Prototyping Methoden und Werkzeuge:

- Die verwendeten Prototyping Methoden beruhen auf unterschiedlichen Medien, mit denen grafische Benutzerschnittstellen visualisiert werden
- Die verwendeten Prototyping Methoden werden in verschiedenen, d.h. frühen und späten Phasen der Entwicklung von Benutzerschnittstellen eingesetzt
- Die verwendeten Prototyping-Werkzeuge sind in ihrer Anwendung weit verbreitet
- Die verwendeten Prototyping-Werkzeuge sind gut dafür geeignet, die zuvor ausgewählten Prototyping Methoden umzusetzen

Auf diesem Hintergrund entschlossen wir uns, im Rahmen dieser Arbeit folgende drei Prototyping-Methoden und vier Prototyping-Werkzeuge zu untersuchen:

Prototyping-Methode	Prototyping-Werkzeug(e)
Paper Prototyping	Papier, PowerPoint
Digital Prototyping	Axure RP Pro
Coded Prototyping	Ajax

Tabelle 6: Auswahl der Prototyping Methoden und -Werkzeuge

Die Wahl der Werkzeuge spiegelt deren individuellen Stärken bei der Umsetzung der jeweiligen Prototyping Methoden wider. So wurden die Papier-Prototypen einmal mittels Papier und einmal mittels PowerPoint hergestellt. Beim digitalen Prototyp wurde die professionelle Prototyping-Software „Axure RP Pro“ verwendet. Der kodierte Prototyp wurde schliesslich mit Ajax umgesetzt.

3.3 Erstellung der Prototypen

Dieses Kapitel beschreibt, wie wir die Prototypen für die drei gewählten Methoden (Papier, Digital, kodiert) mit den vier Prototyping Werkzeugen Handzeichnung auf Papier, PowerPoint, Axure RP Pro und Ajax erstellt haben. Wegen der grossen Ähnlichkeit werden die beiden Papier-Prototypen gemeinsam behandelt.

3.3.1 Per Hand und PowerPoint erstellte Papier-Prototypen

Die Erstellung von handgezeichneten oder elektronisch erstellten Papier-Prototypen erfordert keine speziellen technischen Kenntnisse. Die benötigten Zeichenkünste für Wireframes, wie sie üblicherweise in Papier-Prototypen verwendet werden, beschränken sich im Wesentlichen auf Linien und Vierecke. Auch bei Programmen für die Bildschirmpräsentation oder Erstellung von Diagrammen (zB. PowerPoint, Keynote, Visio, OmniGraffle) genügen Grundkenntnisse der Handhabung.

Entsprechend wird in der Literatur immer wieder darauf hingewiesen, wie schnell und einfach Papier-Prototypen erstellt werden können. Tatsächlich fanden wir alle benötigten Materialien für handgezeichnete Prototypen im normalen Bürofachhandel, sofern wir sie nicht bereits vorrätig hatten. Dies waren Stift, Papier, Haftnotizen in verschiedenen Farben, Korrekturstreifen, Schere, Schneidmesser, Leim (wiederablösbarer und normaler) Plastikhüllen und Karton. Auch für elektronisch erstellte Papier-Prototypen kann man davon ausgehen, dass ein Präsentations- oder Zeichenprogramm im normalen Büroumfeld bereits installiert ist.

Es stellten sich aber dennoch Herausforderungen bei der Erstellung:

- Gewisse Bildschirmelemente mussten aufwändig aus verschiedenen Grundformen zusammengesetzt werden, wofür PowerPoint wenig Hilfe bot (zB. Skala)
- Wir mussten herausfinden, wie man dynamische Vorgänge abbilden kann
- Wir mussten Benutzerhandlungen möglichst genau vorhersagen
- Für alle Vorgänge auf dem Computerscreen mussten wir festlegen, was genau geschieht und was davon abgebildet werden kann/muss und was davon weglassen werden kann/muss
- Jede Änderung im Aufgaben- oder Prototypdesign führte dazu, dass der Prototyp in Teilen oder (beim handgezeichneten) vollständig neu gezeichnet werden musste

Es ist an sich problemlos möglich, fehlende Elemente während des Tests an Ort und Stelle zu fertigen. Dies resultiert aber in Arbeits- und Zeitaufwand des Operators (also der Person, die als Computer agiert) sowie die entsprechende Wartezeit durch den Testteilnehmer. Dies kann sich vorhersehbar negativ in der Bewertung auswirken und den Ablauf massiv verzögern. Deshalb wollten wir nach Möglichkeit den Prototypen vollständig mit allen Elementen bereit halten und in einer Form, die nicht bereits bei Beginn des Testes nach Flickwerk aussah.

Abgesehen von der Darstellung (Handzeichnung vs. elektronisch gezeichnet) waren beide Papier-Prototypen ähnlich in der Herstellung und Funktion. In wenigen Fällen haben wir die beiden Papier-Prototypen technisch abweichend erstellt, um mehrere Möglichkeiten testen zu können.

Um den Charakteristiken von Rich Internet Applikationen möglichst gerecht zu werden, wurde nur der Basisbildschirm einmal vollständig erstellt. Alle anderen Elemente wurde einzeln zum Austausch bereitgelegt. Bei der elektronisch erstellten Version war das Browserfenster im Basisbildschirm integriert. Bei der handgezeichneten diente ein auf Karton aufgezogener Ausdruck des Browserfenster als Basis, auf den jeweils der Bildschirminhalt gelegt wurde.

Die Prototypen

Der Schieberegler war einer der komplexesten Prototypen auf Papier: Beim ersten Durchgang sollte lediglich die Obergrenze mit einem Schiebereglerelement gesetzt werden können. Da es nur ein bewegliches Element gab, das der Benutzer manipulieren konnte (der Schieberegler), war dieses selbst nicht das Problem. Die Benutzerhandlung war aber schlecht vorhersagbar. Der Benutzer konnte die Obergrenze auf einen beliebigen Punkt setzen. Dies machte es auch schwierig vorherzusagen, welche Punkte gleichzeitig auf der Karte erscheinen würden und welche Listenresultate auf einem Element zusammengefasst werden könnten.

Hier haben wir für die beiden Prototypen zwei verschiedene Ansätze gewählt: beim handgezeichneten legten wir jeden Kartenpunkt einzeln an. Beim elektronisch erstellten haben wir für die wahrscheinlichsten Kombinationen je eine Karte erstellt. Bei beiden haben wir die Listenresultate zu den kleinstwahrscheinlichen Einheiten zusammengefasst. Als Hilfsmittel für die korrekte Platzierung der Elemente haben wir einen Gesamtausdruck benutzt.

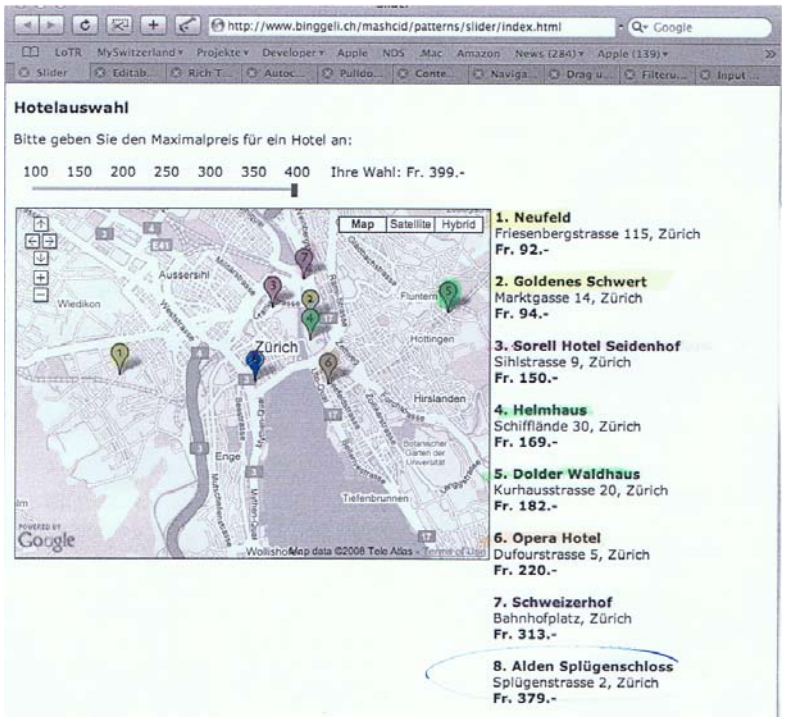

Abbildung 9: Gedächtnisstütze. Kartenpunkte für Pattern "Schieberegler"

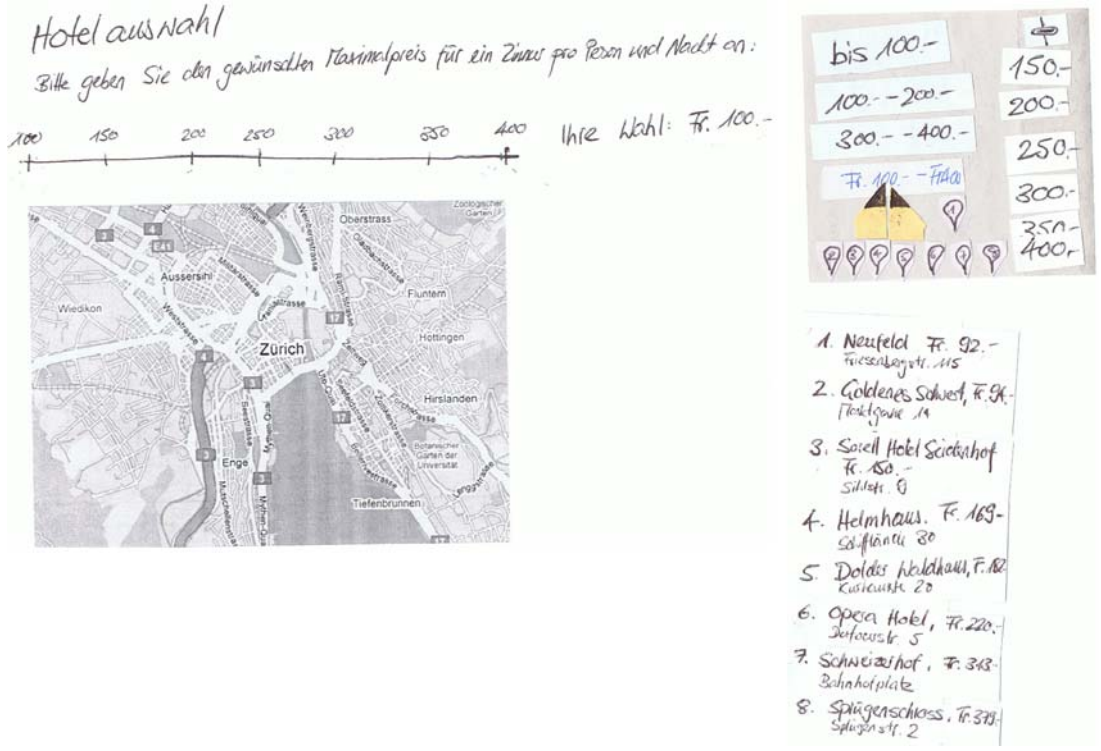

Abbildung 10: Elemente des Papier-Prototyps für Pattern "Schieberegler"

Beim zweiten Durchgang konnte der Benutzer Wertebereiche mit einer Unter- und einer Obergrenze setzen. Da dies zu einer unübersehbaren Fülle von Kombinationen geführt hätte, wurden in der Aufgabenformulierung gezielt genau definierte Bereiche genannt. Dies waren „weniger als Fr. 100.-", „Fr. 100.- bis Fr. 200.-" und „Fr. 300.- bis Fr. 400.-".

Bei der elektronisch erstellen Version des Papier-Prototypen haben wir für genau die in der Aufgabenstellung genannten Einstellungen je eine Karte mit den angezeigten Kartenpunkten erstellt. Beim handgezeichneten blieben wir bei den einzelnen Punkten.

Zum Vergleich der PowerPoint Versionen vom 1. und 2. Durchgang s. Abbildung 12 und Abbildung 13.

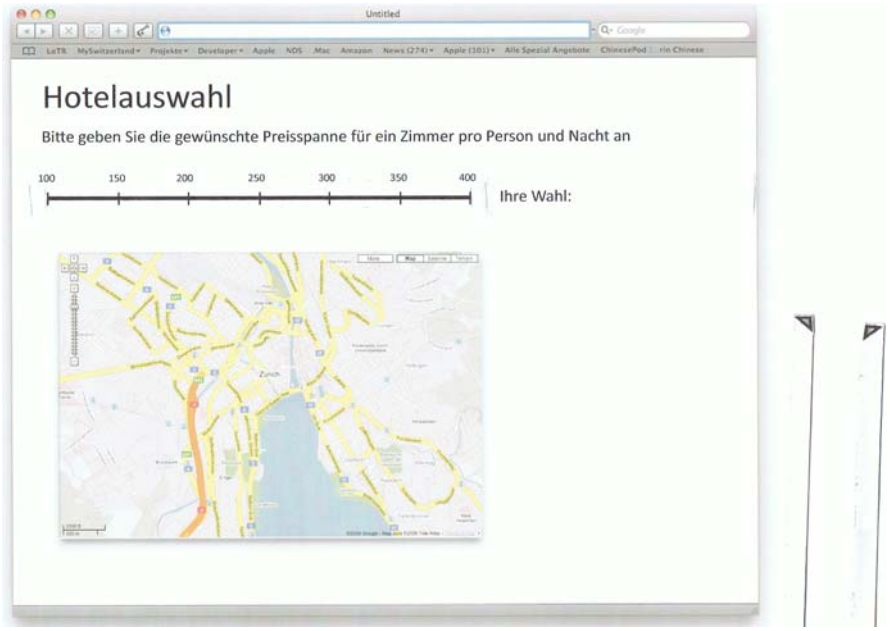

Abbildung 11: Basis PowerPoint-Prototyp für Pattern "Schieberegler"

Abbildung 12: Elemente PowerPoint-Prototyp für Pattern "Schieberegler" 1. Durchgang.

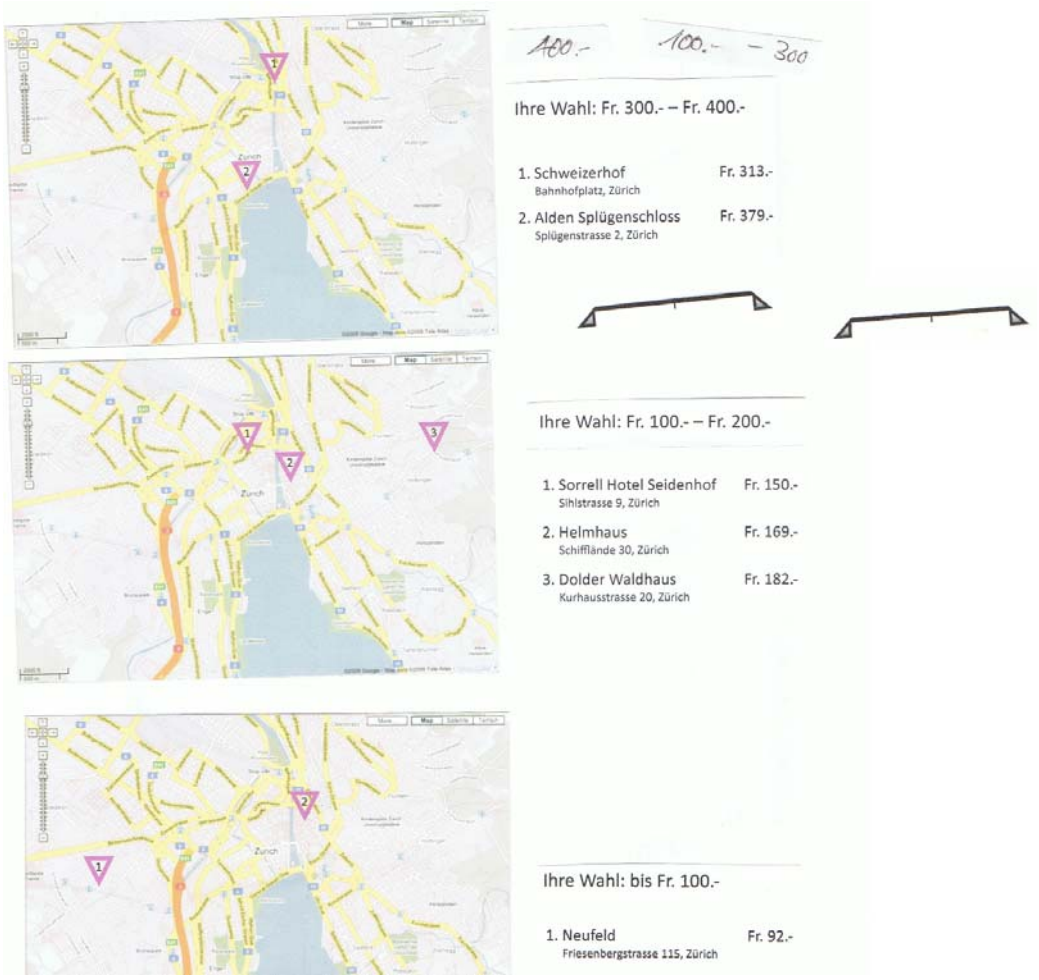

Abbildung 13: Elemente PowerPoint-Prototyp für Pattern "Schieberegler" 2. Durchgang.

Die Überlegung, dass der Benutzer möglicherweise einen Wertebereich einstellt (z.B. oberer und unterer Wert Fr. 100.- auseinander liegend) und diesen dann als gesamtes verschieben möchte, und nicht den oberen und den unteren Wert einzeln, führte zu einem Experiment. Die Absicht war, den oberen und den unteren Schieberegler so einzubauen, dass sie mit einem Element verbunden werden konnten. Beim elektronischen, in PowerPoint erstellten Papier-Prototypen war der Schieberegler daher nicht mehr ein bzw. zwei kleine Dreiecke, sondern zwei lange Streifen, die man in fixem Abstand zueinander hätte fixieren können. Beim Handgezeichneten blieb es bei zwei kleinen Dreiecken, um diese zwei Varianten vergleichen zu können.

Für die Patterns Editierbare Tabellen, Pulldown Menüs, Drag und Drop und Validierung von Eingaben waren die Prototypen aufwändig in der Arbeit, da sie alle viele einzelne kleine Elemente benötigten. Es war jedoch einfach vorherzusagen, was Benutzerhandlungen sein könnten. Für die editierbare Tabelle war in der Aufgabe klar formuliert, welche Daten eingegeben und welche Texte geschrieben werden sollten. Bei den restlichen Patterns mussten einfach alle Möglichkeiten bereitliegen. Da diese aber finit waren (z.B. alle Menüpunkte beim Pulldown Menü), waren sie einfach vorherzusagen und einigermassen einfach in sinnvollen Gruppierungen bereitzulegen.

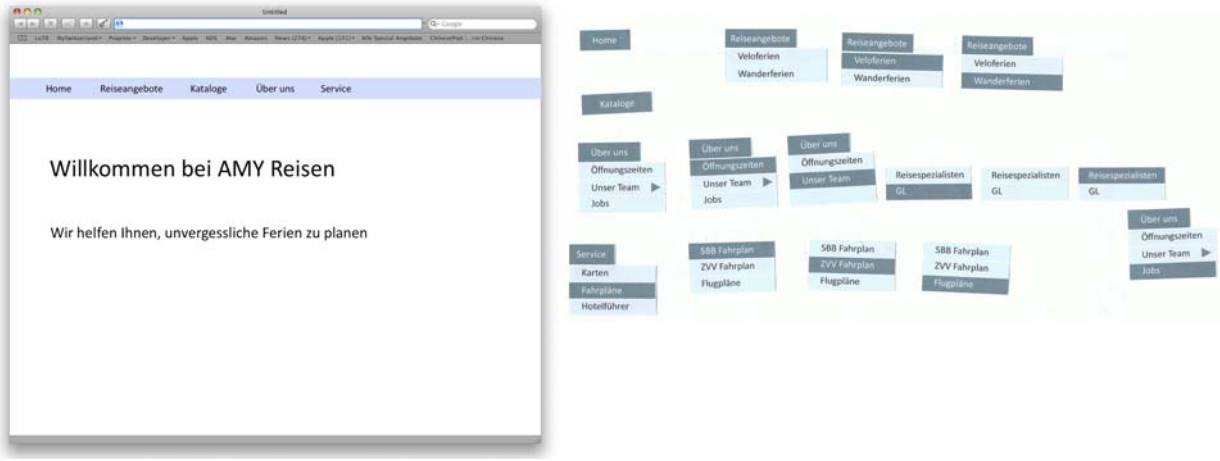

Abbildung 14: Elemente PowerPoint-Prototyp für Pattern "Pulldown Menü"

Beim Pattern Pulldown Menüs zeigte sich ein grundlegender Unterschied zwischen den handgezeichneten und den elektronisch erstellten Protoypen: für die zweite Iteration änderten wir die Navigation, da die Informationsarchitektur beim ersten Testdurchgang zu viel Verwirrung geführt hatte. In PowerPoint mussten nur die Menüpunkte geändert und alles neu ausgedruckt werden, während beim handgezeichneten der Prototyp vollständig neu gezeichnet werden musste. Andererseits war es beim farbigen, mit PowerPoint erstellten Protoypen viel aufwändiger, all die Menüpunktvariationen auszuschneiden und mit wiederablösbarem Klebstoff zu bestreichen, als einige wenige, da farblich identische, Menüpunkte auf Post-it Zettel zu schreiben.

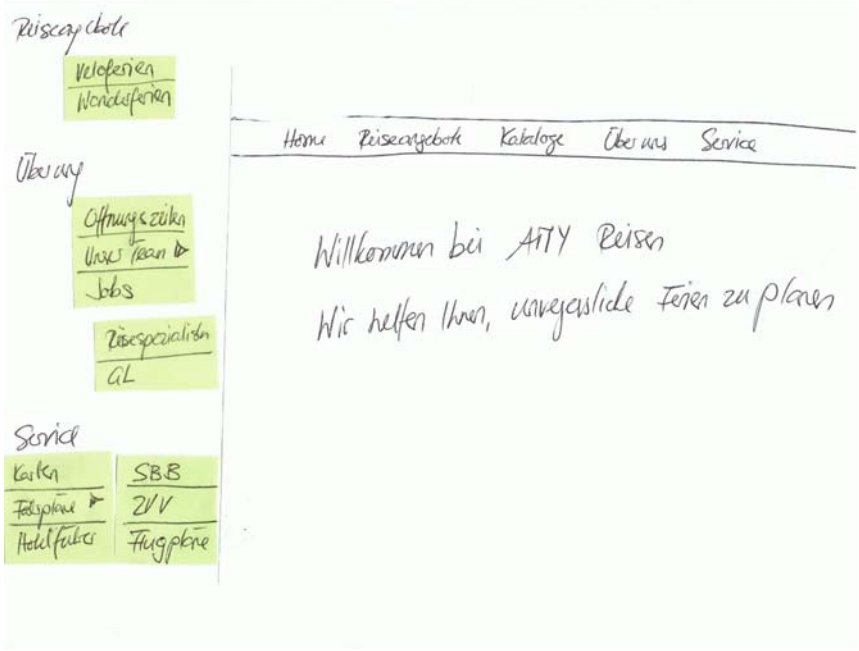

Abbildung 15: Elemente Papier-Prototyp für Pattern "Pulldown Menüs"

Die dynamischen Vorgänge abzubilden, war bei diesen Pattern einfach: entweder war es trivial (Drag und Drop) oder nicht machbar (Editierbare Tabelle). Eine Cursoränderung anzuzeigen ist beispielsweise ausgeschlossen, weil einerseits Cursor generell nicht gezeigt werden und andererseits die Reaktionsgeschwindigkeit des menschlichen Computers niemals schnell genug wäre.

Obwohl Drag und Drop trivial umzusetzen war, indem ein bewegliches Element angeboten wurde, war die Entscheidung dafür nicht trivial: allein der Umstand, dass da ein einzelnes, bewegliches Element liegt, war bereits ein Hinweis darauf, dass es vom Testteilnehmer bewegt werden soll. Dies entspricht nicht dem Aufforderungscharakter im echten System. Da uns aber keine weniger auffällige Alternative einfiel, haben wir es im Prototypen auf diese Weise umgesetzt.

Bei den Patterns Rich Text Editor, Eingabevorschläge, Kontext Menü und Filterung von Suchergebnissen gab es recht wenige Elemente umzusetzen und auch die Benutzerhandlungen waren sehr vorhersagbar. Die dynamischen Elemente waren einfach umzusetzen, da sie keine Änderungen in Grösse, Form oder Farbe beinhalteten, sondern nur in Text (z.B. Dropdown) bzw. ganzen Elementen (z.B. Kontextmenü-Popup).

Das Pattern Navigation in Karten schliesslich stellte ein grobes Problem dar. Scheinbar extrem simpel sowohl in der Darstellung der dynamischen Vorgänge, der Benutzerhandlungen wie auch der Bildschirmelemente, war es extrem schwierig, die Idee mechanisch umzusetzen. Es sollte eine Karte sein, die in einem Ausschnitt des "Bildschirms" (ein darüberliegendes Papier) sichtbar sein sollte und dahinter herumgeschoben werden konnte. Die Reibung Papier auf Papier liess sich nur sehr schwer ausschalten und konnte erst in der 2. Iteration erreicht werden, indem die Karte in eine Plastikhülle gesteckt und der "Bildschirm" auf die Plastikhülle geklebt wurde.

Generell war es in der handwerklichen Umsetzung möglich, handgezeichnete und elektronisch erstellte Papier-Prototypen gleich zu behandeln. Die Elemente konnten sich im Einzelfall unterscheiden, wie beim Pulldown Menü beschrieben, wenn Farbe eine Rolle spielte. Dabei war es natürlich immer einfacher, gleiche oder ähnliche Elemente elektronisch per Copy/Paste zu erstellen oder durch mehrfaches Ausdrucken, während sie von Hand immer einzeln gezeichnet werden mussten. Andererseits mussten beim handgezeichneten die einzelnen Bestandteile nicht aus dem Ausdruck ausgeschnitten und mit wiederablösbarem Leim versehen werden. Der Aufwand für beide Papier-Prototypen-Arten hielt sich dadurch ungefähr die Waage.

3.3.2 Mit Axure RP Pro erstellte digitale Prototypen

Axure RP Pro ist eine spezialisierte Prototyping-Software (vgl. Seite 25). Sie dient dazu, Wireframes, Prototypen und Spezifikationen für Applikationen und Internetseiten zu erstellen.

Das nachfolgende Bildschirmfoto zeigt den grundlegenden Aufbau von Axure RP Pro.

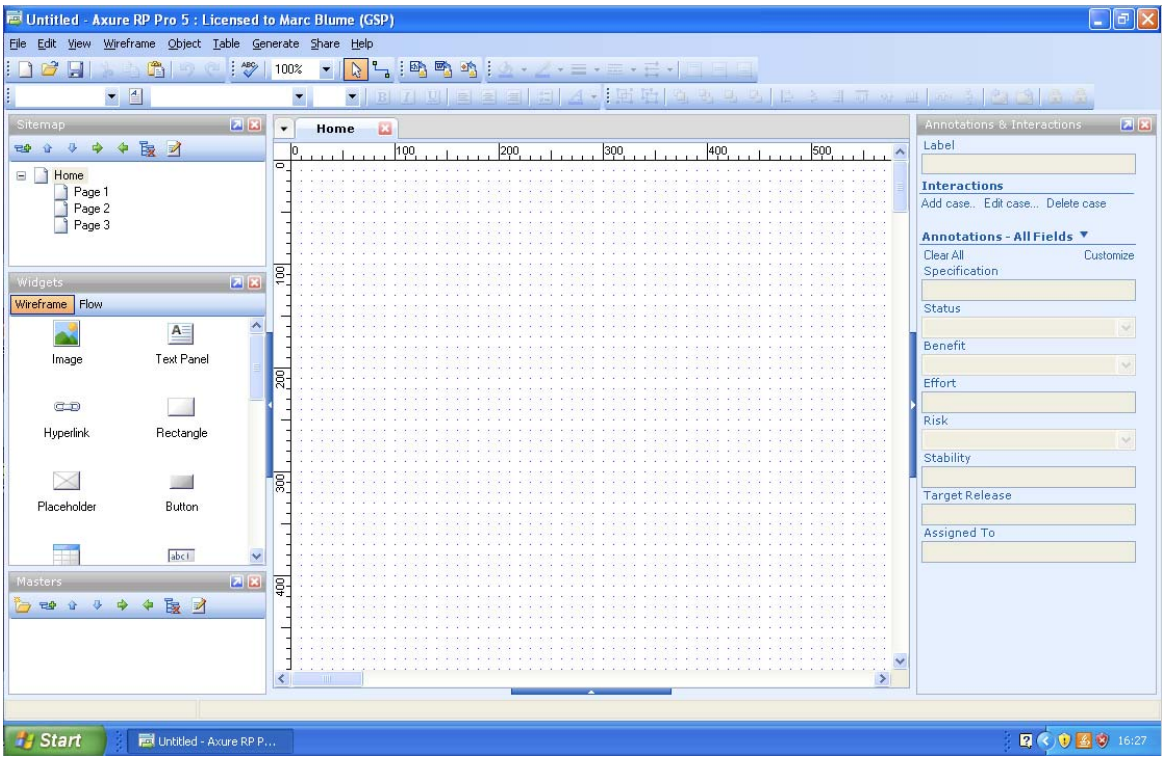

Abbildung 16: Bildschirmaufbau und Arbeitsoberfläche von Axure RP Pro

Axure RP Pro hat einen dreispaltigen Aufbau.

- In der linken Spalte sind folgende Bereiche zugänglich:
  - Eine hierarchische Seitenstruktur („Sitemap“) der erstellten Internetseite bzw. Applikation
  - Eine Sammlung von Interfaceelementen („Widgets“) zur Erstellung der konkreten Benutzerschnittstelle
  - Eine Sammlung von selbst erstellten Vorlagen („Masters“), deren Inhalte sich auf alle Seiten auswirkt (z.B. Navigationselemente, Kopf- und Fusszeilen)
- In der mittleren Spalte befindet sich die zentrale Arbeitsfläche. Auf ihr werden die einzelnen Bildschirme erstellt und bearbeitet. Der Zugang zu verschiedenen Bildschirmen erfolgt über Laschen („Tabs“) am Kopf der mittleren Spalte, oder über die Seitenstruktur der linken Spalte.
- In der rechten Spalte können Systemreaktionen auf Benutzerhandlungen bestimmt werden. Zudem lassen sich Erläuterungen und auch Spezifikationen zu Bildschirmelementen hinterlegen.

Die Erstellung eines Prototypen mit Axure folgt folgendem Muster:

1. Der Nutzer erstellt eine neue Seite und fügt sie am passenden Ort in die bestehende Seitenstruktur ein
2. Der Nutzer zieht Bildschirmelemente aus der Sammlung von Interfaceelementen in der linken Spalte auf die Arbeitsfläche und bearbeitet sie dort
3. Systemreaktionen auf erwartete Nutzerhandlungen werden in der rechten Spalte „programmiert“
4. Der Nutzer erstellt einen interaktiven HTML-Prototyp, testet ihn und korrigiert daraufhin bei Bedarf fehlende oder fehlerhafte Systemreaktionen

Axure RP Pro weist hinsichtlich der Flexibilität der Bildschirmdarstellung im Vergleich mit den Programmen des Microsoft Office Paketes zwei bedeutende Vorteile auf.

Zum einen verfügt Axure RP Pro über sogenannte dynamische Bildschirmelemente („*Dynamic Panels*“). Diese Bildschirmflächen können mit verschiedenen Zuständen hinterlegt werden, so dass mit einer einzelnen Seite eine grosse Vielfalt von Bildschirmzuständen abgedeckt werden kann. Bereits mit dieser bescheidenen Systemlogik ist es nicht mehr notwendig, für sämtliche Kombinationsmöglichkeiten an möglichen Bildschirmzuständen individuelle Bildschirme zu erstellen.

Zum anderen können bedingte Systemreaktionen in einem „wenn-dann“ Format bestimmt werden. Mit dieser Funktion kann die Systemlogik um Abhängigkeiten erweitert werden. Dabei ist auch möglich, einer Nutzerhandlung mehrere Systemreaktionen folgen zu lassen, so dass sich mehrere Bildschirmelemente gleichzeitig anpassen.

Bei der Erstellung der zehn Rich Internet Application Prototypen zeigte sich rasch, dass Axure RP Pro die verschiedenen Prototypen der RIA Patterns stark unterschiedlich gut unterstützt. Um die Spannweite dieser Unterstützung aufzuzeigen, stellen wir nachfolgend drei RIA Pattern Prototypen vor, die sich mit Axure RP Pro gut, mittelmässig bzw. nur schlecht erstellen lassen.

Das RIA Pattern „Dropdown-Menü“ wird bereits standardmässig als Interfaceelement angeboten, wie auf dem nachfolgenden Bildschirmfoto zu sehen ist:

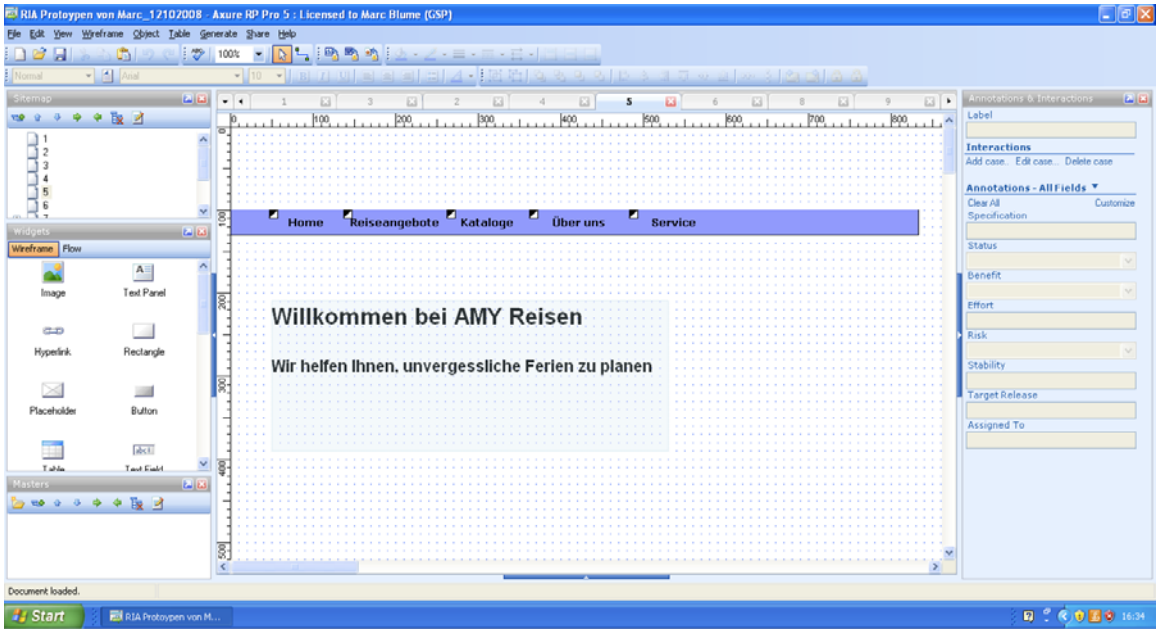

Abbildung 17: Standard-RIA Pattern „Dropdown-Menü“ von Axure RP Pro

Dabei kann die Anzahl und die Beschriftung der einzelnen Menüeinträge in den Dropdowns frei gewählt werden. Mit Untermenüs können Informationshierarchien eingeführt werden. Zudem kann das reguläre Aussehen wie auch das Aussehen, wenn der Mauspfeil darüber geführt wird („Mouseover“) hinsichtlich Schriftart, Schriftgrösse, Textfarbe, Hintergrundfarbe usw. bestimmt werden. Selbstredend lassen sich die einzelnen Menüeinträge mit anderen Bildschirmseiten verlinken.

Der Prototyp des RIA Patterns „Validierung von Eingaben“ liess sich ebenfalls umsetzen, wenn auch mit ungleich höherem Aufwand. Insbesondere die dynamischen Bildschirmelemente kamen dabei intensiv zum Einsatz. Sie sind hier als gelbe Flächen sichtbar:

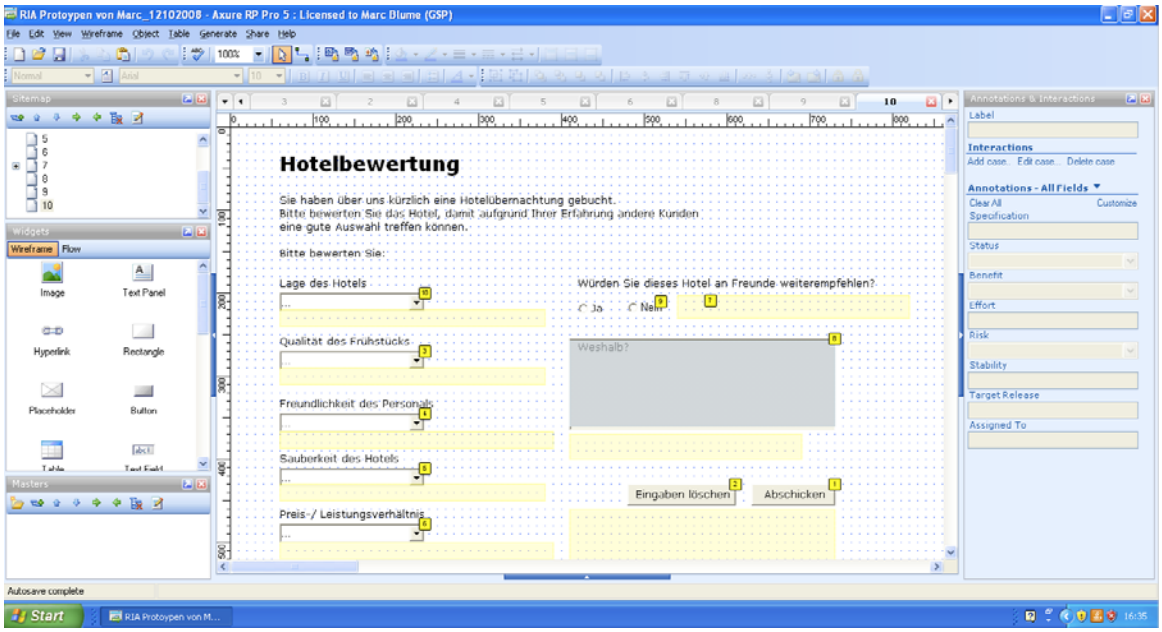

Abbildung 18: Einsatz von dynamischen Bildschirmelementen in Axure RP Pro

Die dynamischen Bildschirmelemente beinhalten Fehlermeldungen und Hinweise, die bei bestimmten Nutzerhandlungen angezeigt (sichtbar) bzw. wieder ausgeblendet werden. Wird zum Beispiel die zweite Eigenschaft des Hotels - die Qualität des Frühstücks - bewertet während die erste Eigenschaft – die Lage des Hotels – noch nicht bewertet wurde, so wird die bei der Lage des Hotels hinterlegte Fehlermeldung sichtbar. Erst mit der Auswahl einer Bewertung für die Lage des Hotels wird das dynamische Bildschirmelement der entsprechenden Fehlermeldung wieder auf „unsichtbar“ gesetzt.

Das RIA Pattern „Validierung von Eingaben“ liess sich wie beschrieben mit den Funktionen von Axure RP Pro umsetzen, wenn auch die Abbildung der komplexen Abhängigkeitsverhältnisse der Systemreaktionen aufwändig war.

Als drittes und abschliessendes Beispiel soll ein Prototyp eines RIA Patterns angeführt werden, das sich nur sehr unbefriedigend mit Axure RP Pro simulieren liess. Obwohl der Schieberegler (Slider) auf diesem Bildschirmfoto des von Axure RP Pro generierten HTML unverdächtig aussieht, liess seine Verhaltensweise sehr zu wünschen übrig.

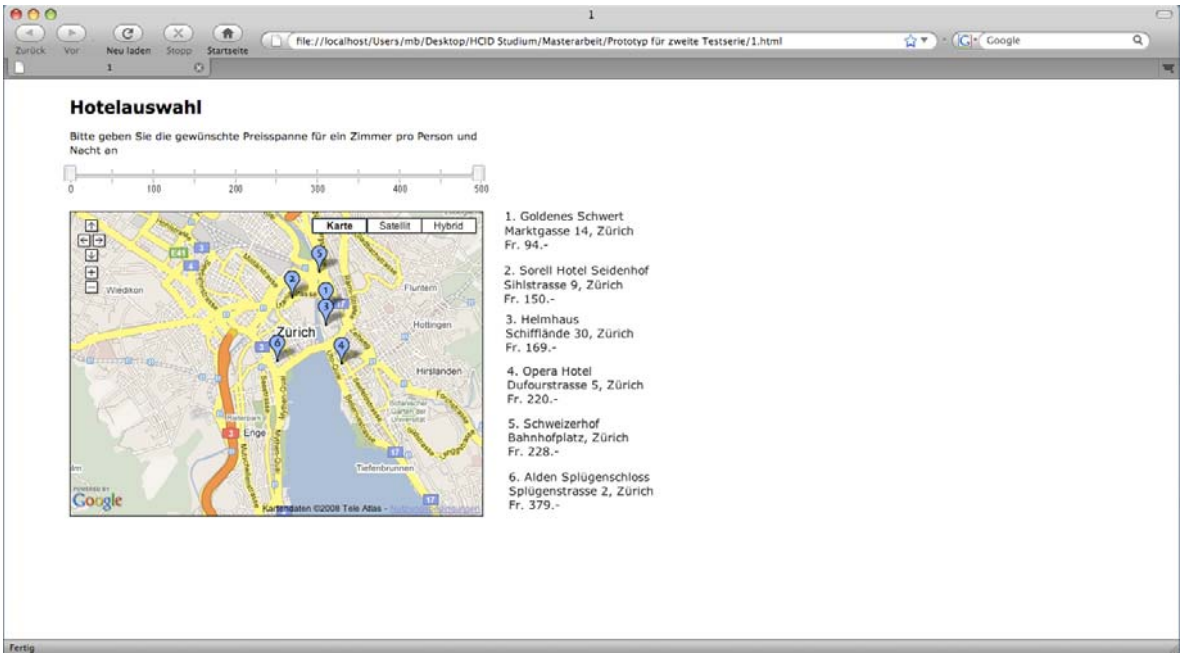

Abbildung 19: Bildschirmfoto eines durch Axure RP Pro generierten HTML-Prototyps eines Schiebereglers

Systemisch betrachtet ist ein Schieberegler ein Bildschirmelement, das sich durch Ziehen mit gedrückt gehaltener Linkstaste der Maus innerhalb eines vordefinierten Bereiches verschieben lässt. Die neue Position dient dabei zugleich als Nutzereingabe und Anzeige eines neuen Systemzustands.

Axure RP Pro mangelt es an zwei zentralen Eigenschaften, um diese Funktionen zu erfüllen:

1. Axure RP Pro unterstützt keine dynamische Verschiebung von Bildschirmelementen mit der Maus. Demzufolge können auch keine Bereiche definiert werden, in denen eine Bewegung eines bestimmten Elementes erlaubt ist.
2. Axure RP Pro „kennt“ den Ort eines Bildschirmelementes nicht. Somit kann eine Ortsveränderung eines Elementes auch nicht als Eingabe eines neuen Wertes interpretiert werden.

Nur nach langem Überlegen und mit viel Kreativität konnte eine Lösung erstellt werden, die zumindest mit viel Wohlwollen eine Annäherung an ein Verschieben von Bildschirmelementen darstellte. Das Verhalten der Schieberegler wurde so programmiert, dass sie sich bei einer Berührung mit dem Mauspfeil („Mouseover“) *scheinbar* in eine vorbestimmte Richtung bewegen, indem der Schieberegler der aktuellen Position ausgeblendet und der rechts bzw. links benachbarte Schieberegler eingeblendet wurde. Gleichzeitig mit dieser Veränderung wurden die Hotels der entsprechenden neuen Preisklasse mit Hilfe von dynamischen Bildschirmelementen auf der Karte und der Ergebnisliste angezeigt.

Obwohl damit in einem sehr bescheidenen Masse eine Bewegung eines Bildschirmelementes per Maus erreicht wurde, war die Lösung in mehrerlei Hinsicht unbefriedigend. So interpretiert Axure RP Pro jeden Mouseover über einen Schieberegler als Absicht, ihn vom Rand weg zu bewegen. Somit befinden sich die Schieberegler auf einer „Einbahnstrasse“, und lassen sich nicht wieder zurück in die entgegengesetzte Richtung verschieben. Zudem sind die beiden Schieberegler voneinander unabhängig, so dass bei einer entsprechenden Manipulation der Schieberegler des Maximalwertes plötzlich links von dem des Minimalwertes steht – ein Zustand, der bei einer Werteeingabe

natürlich gegen die konventionelle Leserichtung von links nach rechts (im Sinne des „von... bis...“) verstösst und auf Nutzer stark irritierend wirkt.

Diese fehlende Eigenschaft von Axure RP Pro, Bildschirmelemente dynamisch mit der Maus verschieben zu können hat bei mehreren Prototypen von RIA Patterns zu erheblichen Problemen geführt – sowohl bei ihrer Erstellung wie auch erwartungsgemäss bei deren Konfrontation mit Testteilnehmern. Betroffen waren neben dem Schieberegler auch der Prototyp des „Drag und Drop“ RIA Patterns, sowie das Verschieben von Kartenausschnitten.

Über dieses Problem hinaus erkennt Axure RP Pro keine Rechtsklicke auf der Maus, so dass das Kontextmenü des entsprechenden Prototyps des RIA Patterns nur unbefriedigend simuliert werden konnte. Diese Beispiele unterstreichen, dass Axure RP Pro viele RIA Patterns nur unzureichend in Prototypen simulieren kann.

Nach der Erstellung aller zehn Prototypen der RIA Patterns in Axure RP Pro wurden sie in einen lauffähigen HTML-Code überführt. Die Testteilnehmer konnten somit die durch Axure RP Pro erstellten Prototypen in einer vertrauten Browserumgebung nutzen.

3.3.3 Mit Ajax erstellte, kodierte Prototypen

Für die ausprogrammierten Prototypen haben wir entschieden, ausschliesslich Ajax Technologien und keine Player-basierten Technologien wie Flash oder Silverlight einzusetzen. Nach der Festlegung der zu erstellenden Prototypen haben wir verschiedene Open-Source Ajax Frameworks auf ihren Abdeckungsgrad hin untersucht. Da keines der Frameworks Unterstützung für alle der notwendigen Patterns bot, wurde der Ansatz gewählt Komponenten aus verschiedenen Quellen wie Widgets aus Ajax Libraries und Beispielcode zu verwenden. Dieses Vorgehen wird auch als „Patchwork Prototyping“ verstanden (Floyd, Jones, Rathi & Twidale, 2007).

Als Entwicklungstools wurden einzig ein normaler Text-Editor und ein JavaScript Debugger (Firebug) eingesetzt. Da einige der Prototypen auch eine in PHP geschriebene Server-Komponente benutzten, mussten die Prototypen auf einem Web-Server mit installierter PHP Laufzeitumgebung installiert werden. Da die angezeigten Daten in einfachen Dateien oder im Programmcode selbst abgebildet wurden, war der Einsatz von Datenbanken nicht erforderlich.

Einige der Prototypen liessen sich relativ schnell realisieren, da vorhandener Beispielcode jeweils nur für die entsprechenden Szenarien modifiziert werden musste. Andere Prototypen erforderten aufgrund der komplexeren Szenarien mehr Entwicklungszeit.

Die Pulldown-Menus basierten auf einer vorhanden Lösung bei welcher strukturierte HTML Listen mit Cascading Stylesheets (CSS) formatiert und mit Hilfe von JavaScript animiert wurden.

Für die Kontext-Menus wurde “Proto.Menu” (http://code.google.com/p/proto-menu) eingesetzt, eine einfache Lösung die auf der „Prototype“ Ajax Library basiert. Die Menus werden in Form von JSON Objekten spezifiziert. Eine Callback Funktion definiert was im Falle eines Klicks auf das Menu Item passieren soll.

Als Rich Text Editor wurde die frei verfügbare FreeRichTextEditor Komponente (http://www.freerichtexteditor.com) eingesetzt. Diese musste lediglich in die Seite integriert und geringfügig konfiguriert werden.

Der Prototyp mit den editierbaren Tabellen wurde mit DHTML Grid (http://dhtmlgrid.sourceforge.net), einer JavaScript Implementierung eines Grid Controls, realisiert. Inhalte in Tabellenzellen können damit durch einen Klick editierbar gemacht

werden. Die Funktionalität des Controls wurde dahingehend erweitert, dass beim Klick auf ein Datumsfeld ein Kalender geöffnet wird, um einen Tag auszuwählen.

Abbildung 20: Ajax Prototyp „Editierbare Tabellen“

Für den Drag und Drop Prototyp wurde eine Beispielanwendung auf der Basis der „Prototype“ und „script.aculo.us“ Frameworks adaptiert und erweitert. Das Skript wurde komplett in PHP geschrieben, welches den notwendigen HTML und Ajax Code generierte. Mit der Verwendung einer PHP Session konnte die Anwendung statusbehaftet werden, sodass Produkte, die in den Warenkorb gezogen wurden, bei einem Reload der Seite bestehen blieben.

Der Prototyp „Eingabevorschläge (Type & Suggest)“ wurde ebenfalls auf der Basis des „Prototype“ und „script.aculo.us“ Frameworks entwickelt. Bei jedem Tastendruck des Benutzers in einem der beiden Eingabefelder wird der eingegebene String mit einem Ajax Request an den Server geschickt und mit der Liste der Orte verglichen. Die Treffer werden in Form einer HTML Liste an den Browser zurückgeschickt und in einem Auswahlfeld unterhalb der Texteingabe angezeigt.

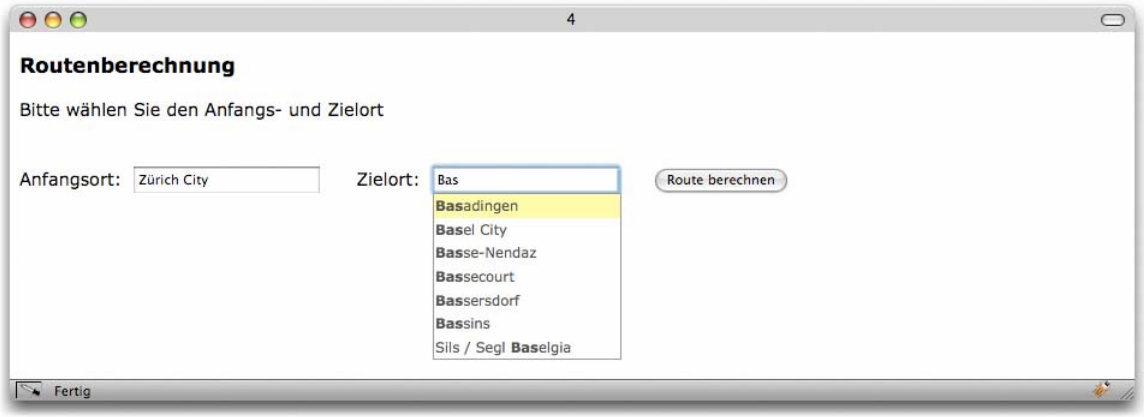

Abbildung 21: Ajax Prototyp „Eingabevorschläge“

Der notwendige JavaScript Code auf der Client Seite beschränkt sich dabei auf die Registrierung der Eingabe- und Ausgabefelder und dem Namen des Skripts auf der Serverseite.

Für den Schieberegler Prototyp sollte ein Preisbereich für die Anzeige von passenden Hotels gewählt werden können. Bei jeder Veränderung des Minimal- oder Maximalwerts sollte die Karte und die Liste automatisch angepasst werden. Für die Umsetzung wurde das Slider Widget aus der Yahoo! User Interface Library (YUI) und das Google Maps API in Verbindung mit der GPlotter Erweiterung eingesetzt. Bei jedem Klicken oder Ziehen des Schiebereglers wurde die selektierte Preisspanne mit einem Ajax Request an den Server

übermittelt, wo die Liste der Hotels gefiltert und die Treffer in Form eines XML Dokuments zurück an den Client geschickt wurden. Diese XML Liste enthielt die Koordinaten und Daten der Hotels, die dann mit dem Google Maps API verarbeitet werden konnten.

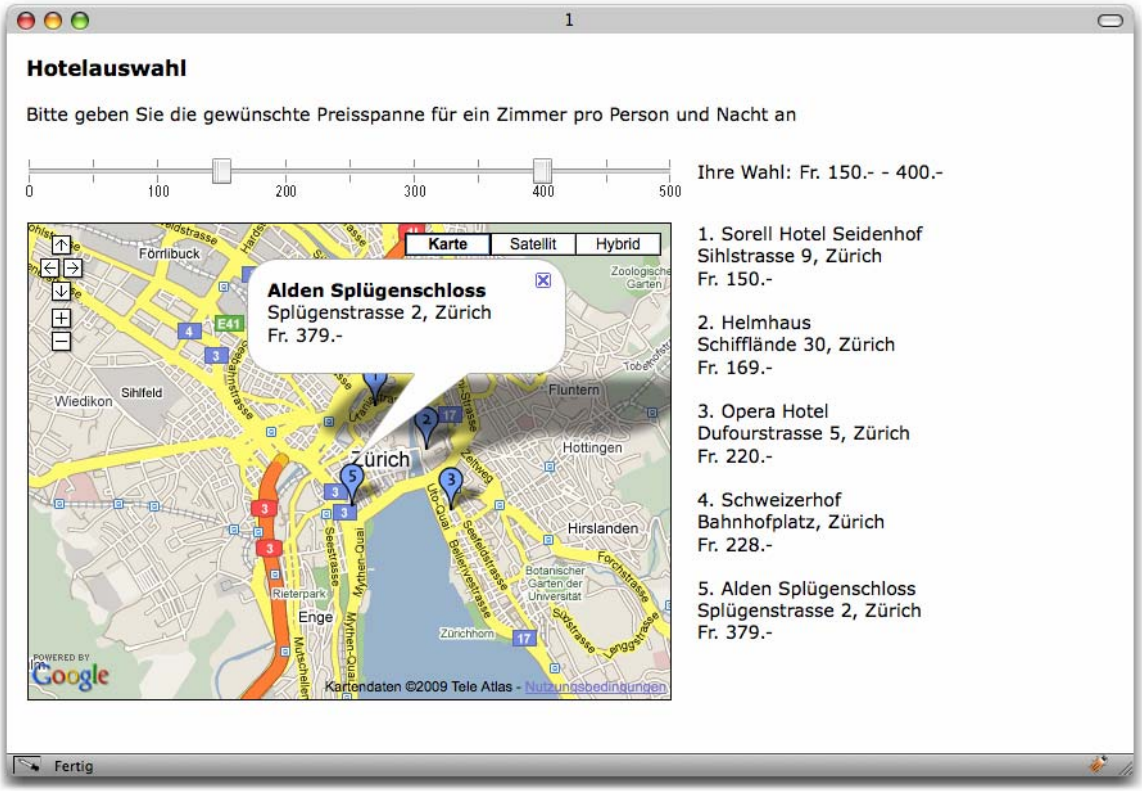

Abbildung 22: Ajax Prototyp „Schieberegler“

Beim Prototyp „Kartenausschnitte verschieben“ wurde ebenfalls das Google Maps Interface integriert.

Der Prototyp „Kartenausschnitte verschieben“ verwendete die gleichen Technologien, wobei die Punkte auf der Karte bereits fest eingezeichnet wurden und keine Interaktion zwischen dem Browser und dem Server stattfand.

Für die sofortige Filterung von Suchergebnissen wurde für die Ajax Kommunikation ebenfalls die „Prototype“ Library eingesetzt. Bei jeder Interaktion des Benutzers wurden die Filterkriterien mit einem Ajax Request an den Server übermittelt, wo der HTML Code dynamisch generiert und an den Browser zurückgeschickt wurde. Die Antwort des Servers kann dann mit einer DOM-Manipulation direkt in die Seite geschrieben werden. Während der Dauer des Requests wird dem Benutzer eine Fortschritts-Meldung eingeblendet.

Der Prototyp „Validierung von Eingaben“ wurde mit eine Formular-Validierungs-Erweiterung für das „Prototype“ Framework realisiert (http://tetlaw.id.au/view/javascript/really-easy-field-validation). Dieses Skript erlaubt die Programmierung von Validierungsregeln mit der entsprechenden Fehlermeldung, die am Ort des Fehlers angezeigt wird. Die Fehlermeldungen erscheinen jeweils, wenn das entsprechende Eingabefeld den Fokus verliert.

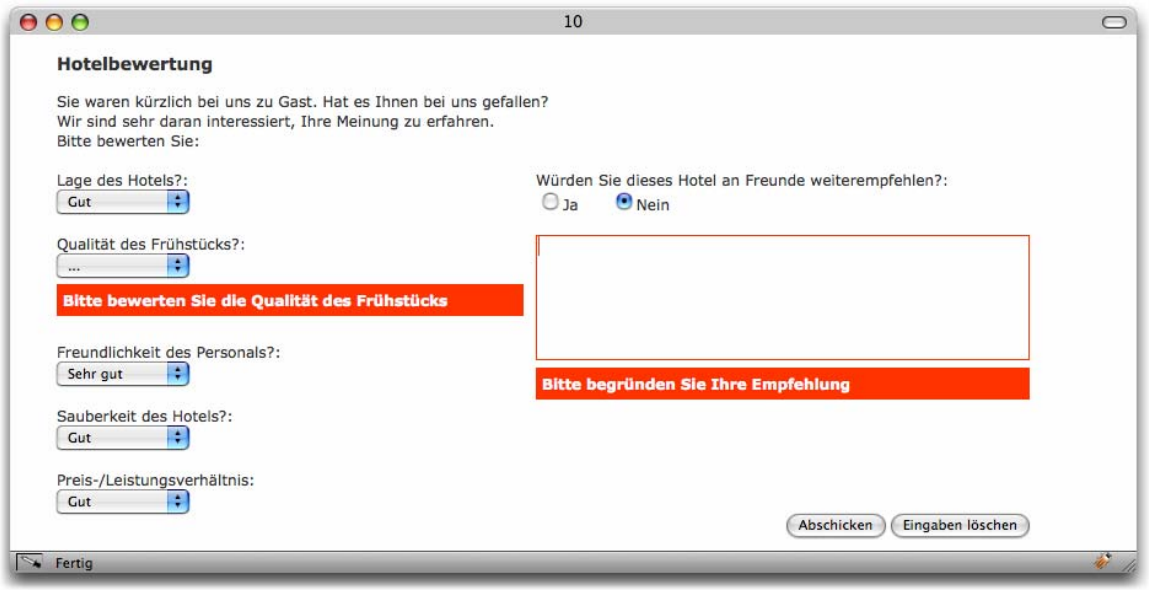

Abbildung 23: Ajax Prototyp „Validierung von Eingaben“

3.4 Untersuchungsdesign

Dieses Unterkapitel beschreibt, welche methodische Vorgehensweise wir zur Beantwortung der Forschungsfragen gewählt haben.

Anzahl Iterationen

Es war uns ein Anliegen, iterativ vorgehen zu können, da dies dem Ansatz des User Centered Design entspricht. Wir entschieden uns für 2 Iterationen. Dies ist einerseits die minimale Anzahl, um überhaupt von Iterationen sprechen zu können. Andererseits, unter Berücksichtigung der benötigten Planungs-, Vorbereitungs-, Rekrutierungs- und Auswertungsdauer für einen Usability Test, war diese Beschränkung nötig, um die Untersuchung erfolgreich durchführen zu können.

Ebenfalls unter Berücksichtigung der Vorbereitungsdauer mussten wir uns für eine Anzahl Pattern entscheiden, welche wir als Prototypen umsetzen wollten. Wir stellten einige Kombinationsmöglichkeiten auf.

Ein Prototyp musste für jede Kombination von Prototyp-Art und Pattern erstellt werden. Diese Gesamtzahl an Prototypen musste gleichmässig auf die Anzahl Testteilnehmer verteilt werden können. Die Faktoren waren daher immer eine Kombination von

Anzahl Pattern x Anzahl Prototyping Arten x Anzahl Testteilnehmer

Nach Nielsen und Landauer (1993) genügen 5-6 Testteilnehmer, um etwa 70% - 80% der grossen Usability Probleme aufzudecken (vgl. Seite 31).

Von den durchgespielten Möglichkeiten entschieden wir uns für

10 Pattern x 4 Prototyp-Arten = 40 Prototypen

Da wir jeden Prototypen (mindestens) einmal testen wollten, ergab dies 40 Aufgaben, aufzuteilen auf die Anzahl Testteilnehmer. Bei 5 Testteilnehmern bedeutete dies 8 Aufgaben pro Testsitzung. Bei einer üblichen Evaluationsdauer von einer Stunde pro Sitzung schienen uns dies zumutbar zu sein. Jeder der Testteilnehmer würde dadurch ein Pattern maximal einmal erhalten und jede Prototyp-Art zweimal.

Die erste Iteration führten wir planmässig durch. Bei einigen Prototypen zeigten sich bei den Tests erhebliche Probleme. Dies lag teilweise an Fehlern des Prototyps:

- der Papier-Prototyp für das Pattern „Navigation in Karten“ war gar nicht bedienbar. Die Karte liess sich nicht bewegen.
- der ausprogrammierte Prototyp für „Eingabevorschläge (Type & Suggest)“ reagierte zu langsam und führte dazu, dass die Eingabevorschläge gar nicht erschienen.
- für alle Prototypen-Arten zu „Schieberegler (Slider)“ wurde von den Testteilnehmern die Möglichkeit gewünscht, einen Wertebereich einstellen zu können

Teilweise lag der Fehler aber auch in der Aufgabenformulierung. Wir beschlossen daher nach dem ersten Durchgang, die entsprechenden Testaufgaben zu ändern und die Prototypen zu verbessern.

Dies hätte für gewisse Patterns bedeutet, dass wir bei einem abweichenden Resultat nicht hätten bestimmen können, ob dies an der Änderung von Aufgabe und/oder Prototyp lag. Deshalb beschlossen wir, für den 2. Durchgang 10 Testteilnehmer zu rekrutieren, um trotz der Änderung der Aufgaben und der Designs für jede Kombination von Aufgabe und Prototyp zumindest 2 vergleichbare Resultate zu erhalten.

Da aber der Prototyp bzw. die Aufgabenstellung nur verbessert und nicht grundsätzlich neu entworfen wurde, wurde die Auswertung dennoch über alle 15 Testteilnehmer der 2 Iterationen vorgenommen, ohne zwischen den zwei Iterationen zu unterscheiden.

3.4.1 Testaufgaben

Für jedes der ausgewählten Patterns wurde eine Aufgabe erstellt. Wir entschlossen uns, zwar einen Kontext zu formulieren, diesen jedoch relativ generisch zu halten.

Ein Prototypentest völlig ohne Kontext erschien uns nicht zweckmässig, da es bei der Usability immer auch um Aufgabenangemessenheit geht (DIN EN ISO 9241-110). Ein Bedienelement völlig losgelöst von jedem Inhalt zu testen, ist dafür nicht geeignet.

Jedoch sollten sich die Testteilnehmer auf das umgesetzte RIA Pattern und nicht auf die visuelle Umsetzung konzentrieren. Wir wollten daher kein Design kreieren oder eine gesamte Site erstellen. Jedoch wollten wir zur einfacheren Orientierung den Testteilnehmern einen Aufgaben-Rahmen anbieten.

Entschieden haben wir uns für den Rahmen eines Reiseplaners. Dieser Aufgabenbereich ist jedem Testteilnehmer in der anvisierten Benutzergruppe vertraut und bietet unterschiedlichste Aufgabenstellungen.

Wir haben daher für jedes der 10 Pattern einen Kontext und eine Aufgabe innerhalb des gewählten Rahmens eines Reiseplaners geschaffen. Es war jedoch nicht vorgesehen, dass sich die Testteilnehmer in einer bestimmten Reihenfolge, analog einer echten Reiseplanung, durch die Aufgaben bewegen sollten.

RIA	Pattern Name	Kategorie	Kontext	Aufgabe
1.	Schieberegler (Slider)	Datenaus-wahl	Variation von Eingabewerten	Hotels unterschiedlicher Preiskategorien darstellen
2.	Editierbare Tabellen	Datenein-gabe	Direkte Manipulation von Tabellenwerten, z.B. durch Anklicken und Überschreiben der Werte	Tabellarischen Reiseplaner bearbeiten

3.	Rich Text Editor	Datenein-gabe	Eingabefeld eines Formulars, das verschiedene Formatierungsmöglich keiten bietet	Hotelfeedback ausfüllen
4.	Eingabevor-schläge (Type & Suggest	Datenein-gabe	Trefferliste, die sich nach jeder Zeicheneingabe in ein Suchfeld dynamisch anpasst	In einem Fahrplan suchen
5.	Pulldown Menüs	Navigation	Horizontale Navigationsleiste, die auf Anklicken ein Pulldown Menü mit Auswahlmöglichkeiten anbietet	Auf einer Website den Fahrplan finden
6.	Kontext Menüs	Navigation	Aufruf eines Kontextmenüs mittels Rechtsklick auf Arbeitsfläche	Veranstaltungsinfos für eine Reisedestination aufrufen
7.	Kartenaus-schnitte verschieben	Bilder und Karten	Verschieben eines Kartenausschnittes mittels Klicken und Ziehen	Sich geographisch über die Reise informieren
8.	Drag und Drop	Objektma-nipulation	Sortieren von Fotos durch Drag und Drop von Fotos/ Fotostapeln	Reiseartikel in einen Warenkorb legen
9.	Sofortige Filterung von Suchergebnissen	Daten-aktuali-sierung	Eingrenzung der Suchkriterien, die sich sofort auf Suchergebnisse auswirkt	In einem Fahrplan die Suchergebnisse eingrenzen
10.	Validierung von Eingaben	Feedback	Rückmeldung über (ungültige) Eingabewerte bei einer Eingabemaske	In einem Formular eine Hotelbewertung abgeben

Tabelle 7: Testaufgaben

Zu jeder Aufgabe wurde eine Webseite erstellt, die in jeder Prototyp Art identisch umgesetzt wurde. Soweit es das Prototyping Tool zuliess, war die visuelle Wiedergabetreue daher weitgehend vergleichbar. Da gewisse Tools einen genau gescripteten Ablauf verlangten, um Interaktivität vortäuschen zu können, wurde die Aufgabenformulierung auf diesen Ablauf zugeschnitten. Die genaue Formulierung findet sich im Anhang.

Zwischen dem ersten und dem zweiten Durchgang mussten wir Änderungen an der Aufgabenformulierung vornehmen. Dies betraf in erster Linie die Aufgaben zu den Pattern "Kartenausschnitte verschieben" und "Validierung von Eingaben ".

Bei der Aufgabe "Kartenausschnitte verschieben" war den Testteilnehmern nicht klar, welche Handlungen aufgrund der Aufgabe erwartet wurden und wann genau die Aufgabe als gelöst betrachtet werden konnte.

Die Aufgabenstellung bei Aufgabe "Validierung von Eingaben" gab absichtlich ungenügend Angaben zum Ausfüllen eines Formulars, damit die Fehlermeldungen sicher angezeigt wurden. Dies verärgerte die Testteilnehmer so sehr, dass wir den Eindruck hatten, dass dies auch die Bewertung des Prototyps negativ beeinflusste. Dies führte dazu, dass wir das Aufgabendesign im zweiten Durchgang etwas abwandelten.

3.4.2 Aufgabenmatrix

10 Patterns und 4 Prototypenarten ergaben 40 mögliche Kombinationen. Bei 5 Testteilnehmern pro Durchgang resultierte dies in 8 Aufgaben, die jeder Testteilnehmer (TN) lösen musste. In der ersten Iteration wurde ein Durchgang mit 5 Testteilnehmern (TN 1 - TN 5), in der 2. Iteration mit 10 Teilnehmern (TN 6 - TN 15) durchgeführt. Jede Aufgaben-Pattern-Prototyp-Kombination wurde also von 3 Testteilnehmern gelöst und kam pro Durchgang genau einmal vor.

Um Einflüsse durch die Reihenfolge der Aufgaben oder der Prototyp-Arten zu vermeiden, haben wir die Abfolge randomisiert. Die Aufgaben wurden daher nicht in der Abfolge dargereicht, wie sie bei einer realistischen Reiseplanung angefallen wären.

TN 1/6/11	TN 2/7/12	TN 3/8/13	TN 4/9/14	TN 5/10/15
Papier 1	PowerPoint 9	Axure 7	Ajax 5	Ajax 3
PowerPoint 2	Papier 10	Papier 8	Axure 6	Axure 4
Axure 3	Ajax 1	Ajax 9	PowerPoint 7	Papier 5
Ajax 4	Axure 2	Axure 10	Ajax 8	PowerPoint 6
PowerPoint 5	PowerPoint 3	PowerPoint 1	Papier 9	Papier 7
Papier 6	Papier 4	Ajax 2	PowerPoint 10	PowerPoint 8
Ajax 7	Axure 5	Papier 3	Axure 1	Axure 9
Axure 8	Ajax 6	PowerPoint 4	Papier 2	Ajax 10

Tabelle 8: Aufgabenmatrix

Alle Testteilnehmer haben ein Pattern höchstens einmal erhalten und in jeder Prototyp-Art zwei Aufgaben gelöst.

3.4.3 Auswahl der Testteilnehmer

Als Testteilnehmer wollten wir nur normale Benutzer rekrutieren, welche keine Vorkenntnisse oder Vertrautheit mit Usability bzw. den Methoden der Usability Evaluationen haben. Es waren daher alle Personen ausgeschlossen, welche beruflich auf dem Gebiet Usability und Usability Evaluationen tätig sind.

Die insgesamt 15 Testteilnehmer rekrutierten wir im Kollegen- und Bekanntenkreis. Das Verhältnis männliche (7) und weibliche (8) Testteilnehmer war ausgeglichen. Altersmässig gab es durch einen grossen Anteil Personen in Ausbildung (Praktikanten) eine Dominanz von jüngeren Testteilnehmern. Wir konnten jedoch auch im Altersbereich 40 - 60 einige Testteilnehmer rekrutieren.

Während alle Testteilnehmer sowohl beruflich wie auch privat das Internet benutzen, sind vier davon im engeren Sinn technisch oder inhaltlich mit der Erstellung von Webseiten oder Software vertraut.

Diese soziodemographischen Angaben sind jedoch nicht in die Auswertung eingeflossen. Als Faktor für die quantitative Auswertung haben wir vielmehr die Vertrautheit mit RIA erfragt.

Teilnehmer	Geschlecht	Alter	Beruf
TN 1	w	26	Praktikantin, Beratungsbranche
TN 2	m	23	Praktikant, Beratungsbranche
TN 3	w	28	Content Manager Intranet Wiki
TN 4	w	24	Praktikantin, Beratungsbranche
TN 5	m	25	Praktikant, Beratungsbranche
TN 6	m	30	Web Publisher
TN 7	w	22	Praktikantin, Tourismusbranche
TN 8	m	25	Praktikant, Tourismusbranche, Java-Entwickler
TN 9	w	43	Admin. Assistentin
TN 10	w	22	Admin. Assistentin
TN 11	w	27	Admin. Assistentin
TN 12	m	58	Content Manager, ex-Web Content
TN 13	m	29	Web Publisher
TN 14	w	43	Psychologin
TN 15	m	55	Account Manager

Tabelle 9: Testteilnehmer

3.4.4 Qualitative Beobachtungen während der Tests

Als Evaluationsmethode entschieden wir uns für einen klassischen Usability Test. Die Testteilnehmer sollten möglichst selbständig eine Aufgabe lösen und dabei laut kommentieren, weshalb sie eine Handlung vornahmen bzw. was ihre Absichten und Erwartungen waren. Testleiter und Beobachter befanden sich im gleichen Raum, konnten klärende Fragen stellen und wenn nötig eingreifen.

Alle Tests fanden in Sitzungszimmern statt. Zwei Personen fungierten während der Tests jeweils als Beobachter. Eine Person leitete und moderierte den Test. In diesen Rollen haben wir uns abgewechselt. Eine Person war bei den papierbasierten Prototypen der Operator, also der "Computer". Diese Rolle wurde immer von derselben Person wahrgenommen, war also entweder identisch mit einem der Beobachter oder dem Moderator.

Die Testteilnehmer wurden um Erlaubnis gebeten, dass Videoaufnahmen von den Tests gemacht wurden. Bei den elektronischen Prototypen wurde dabei sowohl der Bildschirm wie auch der Testteilnehmer aufgenommen; dies durch eine Software, die auf dem Testcomputer installiert war und auch die eingebaute Webcam benutzte.

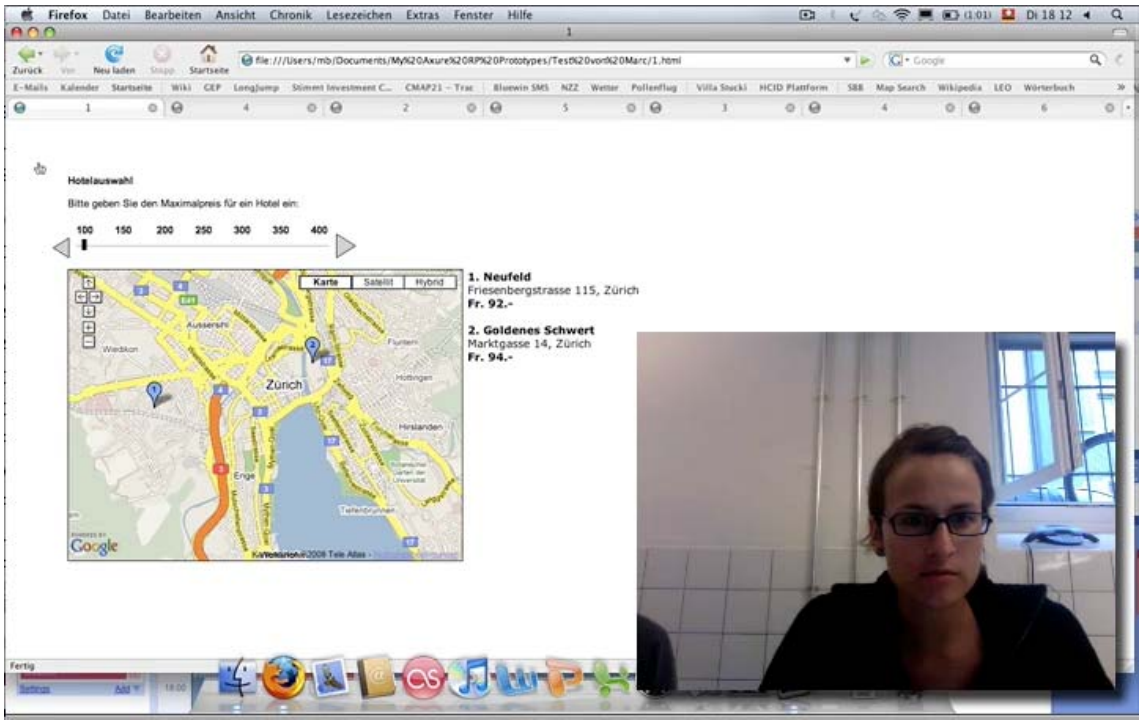

Abbildung 24: Bildschirmaufnahme während Test eines Axure Prototypen

Bei den papierbasierten Prototypen wurde nur der Prototyp mit einer auf dem Tisch installierten Videokamera aufgenommen. Alle Aufnahmen dienten aber nur als Gedächtnisstütze. Grundlage für die qualitative Auswertung waren Handnotizen.

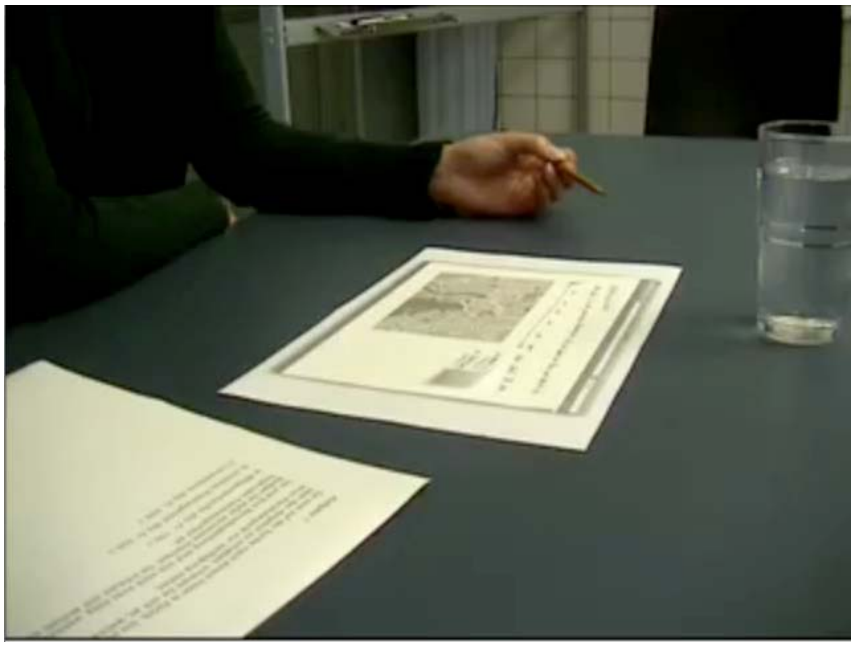

Abbildung 25: Videoaufnahme während Test eines Papier-Prototypen

Während der Tests kamen zwei Sitzordnungen zur Anwendung, je nachdem, ob die Beobachter einen zweiten Bildschirm zur Verfügung hatten.

Gemeinsam war beiden, dass der Testteilnehmer (T) zwischen zwei Plätzen wechseln musste: einem Arbeitsplatz für elektronische und einem für papierbasierte Prototypen.

Der Moderator (M) sass jeweils neben Testteilnehmer und wechselte ebenfalls den Platz, wenn der Testteilnehmer sich umplatzieren musste

Wenn nicht der Moderator selber der Operator (O) war, sass dieser für papierbasierte Prototypen auf der anderen Seite des Testteilnehmers

Die Beobachter (B) sassen schräg gegenüber dem Testteilnehmer auf der anderen Seite des Konferenztisches, wenn sie ihren eigenen Bildschirm hatten. Dies war in der ersten Iteration durchgehend der Fall, in der zweiten nur punktuell.

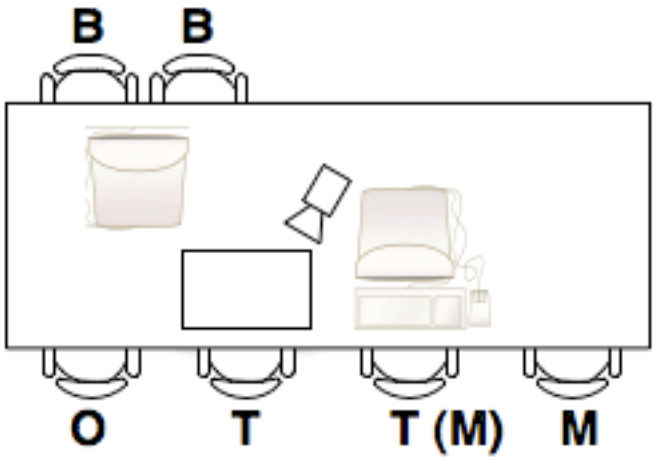

Abbildung 26: Sitzordnung mit 2. Bildschirm

Die Beobacher (B) sassen hinter dem Testteilnehmer, wenn sie seinen Bildschirm sehen mussten.

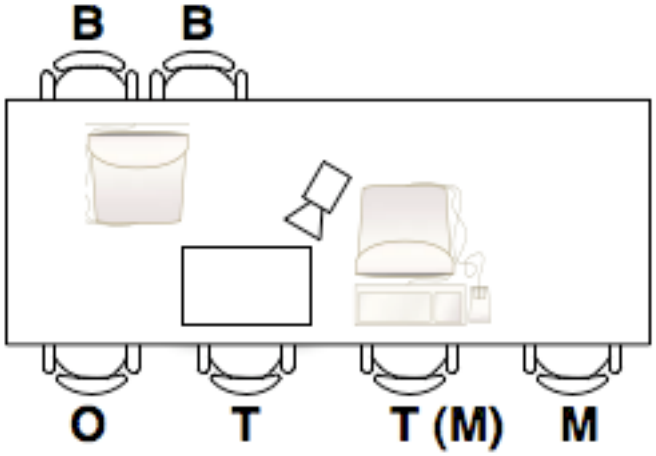

Abbildung 27: Sitzordnung ohne 2. Bildschirm

Abbildung 28: Durchführung einer Testsitzung

Die Beobachter hielten in Freiform-Notizen Testsituationen fest, in denen Teilnehmer die Testaufgaben nur mit Schwierigkeiten oder gar nicht lösen konnten. Ebenso notiert wurden Emotionsausdrücke wie Ärger, Überraschung und Freude in Mimik, Gestik oder Sprache. Diese Notizen wurden nach beiden Testserien in einem gemeinsamen Dokumenteraster zusammengetragen und nach Gemeinsamkeiten und Erkenntnissen ausgewertet.

3.4.5 Quantitative Datenerhebung mittels Fragebogen

Die eben beschriebene qualitative Beobachtung wurde durch zwei quantitative Fragebögen ergänzt. Zum einen diente den Beobachtern ein „objektiver“ Fragebogen dazu, bestimmte Aspekte der Beobachtung in Zahlenwerten auszudrücken. Zum anderen wurde nach jeder Aufgabe den Testteilnehmern ein „subjektiver“ Fragebogen vorgelegt, um ihre persönlichen Bewertung des Prototyps zu erheben. Die Adjektive „objektiv“ und „subjektiv“ implizieren keinerlei Wertung der Stichhaltigkeit der Daten, sondern weisen ausschliesslich auf die Quellen der Bewertungen hin.

„Objektiver“ Fragebogen

Dieses Erhebungsinstrument umfasste fünf Aussagen, mit denen Beobachter das Verhalten des Testteilnehmers nach der Bearbeitung einer bestimmten Aufgabe einstufen konnten. Diese Aussagen lauteten:

1. *"Er/ Sie zögerte bei der Aufgabenbearbeitung."*
2. *"Er/ Sie hatte Schwierigkeiten bei der Aufgabenbearbeitung."*
3. *"Er/ Sie war langsam bei der Aufgabenbearbeitung."*
4. *"Insgesamt kam er/ sie ... zurecht."*

Inwieweit diese Aussagen zutrafen wurde von den Beobachtern auf einer fünfstufigen Likert-Skala bewertet. Die Antwortmöglichkeiten reichten dabei bei den ersten drei Fragen

von „*gar nicht*“ bis „*ausserordentlich*“ und bei der letzten Frage von „*sehr g*ut“ bis „*sehr schlecht*“. Sämtliche Zwischenabstufungen waren ebenfalls mit einer verbalen Beschreibung der entsprechenden Antwortmöglichkeit versehen, wie die folgende Abbildung des Erhebungsbogens aufzeigt:

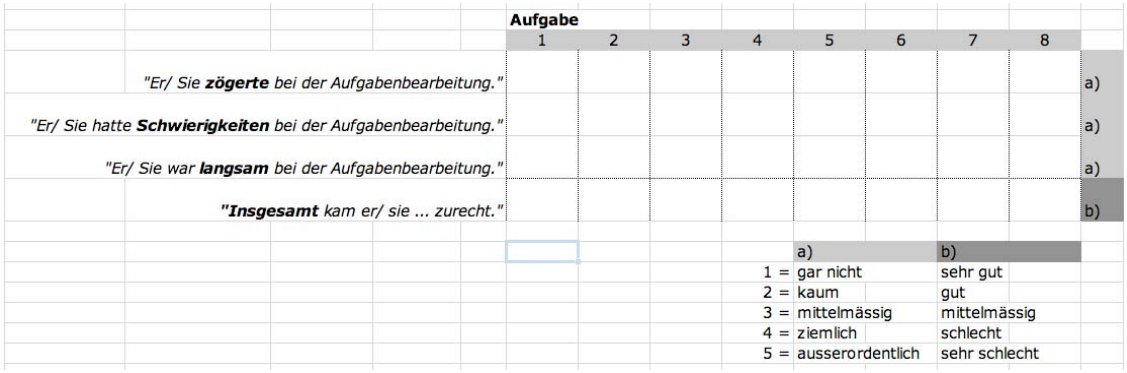

Abbildung 29: Objektiver Fragebogen

Während sich die erste Frage auf Situationen bezog, in denen der Testteilnehmer in seinem Handlungsablauf stockte, wies die dritte Frage auf eine generell verlangsamte Bearbeitungsgeschwindigkeit des Testteilnehmers. Die zweite Frage bezog sich auf Verständnisschwierigkeiten und Situationen, in denen der Testteilnehmer durch seine Handlungen ungewünschte Systemzustände verursachte. Die vierte und letzte Frage fasste zusammen, wie reibungslos der Testteilnehmer die gestellte Aufgabe mit dem Prototyp insgesamt lösen konnte.

„Subjektiver“ Fragebogen

Dieser Fragebogen wurde von uns eigens für diese Untersuchung erstellt. Er umfasste folgende fünf Aussagen, auf die der Befragte zu reagieren hatte:

1. *„Ich kann mich daran erinnern, diese Art der Bedienung bereits früher schon einmal kennen gelernt zu haben.“*
2. *„Irgend etwas an diesem Prototypen hat mich bei der Erledigung der Aufgabe gestört. (Was genau war dies?)“*
3. *„Die Bildschirmelemente des Prototyps sehen genau so aus wie bei einem fertigen System.“*
4. *„Die Bildschirmelemente verhalten sich genau so wie bei einem fertigen System.“*
5. *„Der Prototyp reagierte auf Befehle genau so schnell wie ein fertiges System.“*

Der oder die Testteilnehmer(in) konnte auf einer fünfstufigen Likert-Skala angeben, inwieweit er oder sie der entsprechenden Aussage zustimmt. Die Antwortmöglichkeiten deckten dabei das Spektrum von „Stimme gar nicht zu“ bis „Stimme voll und ganz zu“ ab.

Hier eine Abbildung des verwendeten Fragebogens:

	Stimme gar nicht zu				Stimme voll und ganz zu
Ich kann mich daran erinnern, diese Art der Bedienung bereits **früher schon einmal** kennen gelernt zu haben.					
	1	2	3	4	5
Irgend etwas an diesem Prototypen hat mich bei der Erledigung der Aufgabe **gestört**. (**Was genau** war dies?)					
	1	2	3	4	5
Die Bildschirmelemente des Prototypen **sehen genau so aus** wie bei einem fertigen System.					
	1	2	3	4	5
Die Bildschirmelemente **verhalten sich genau so** wie bei einem fertigen System.					
	1	2	3	4	5
Der Prototyp reagierte auf Befehle genau **so schnell** wie ein fertiges System.					
	1	2	3	4	5

Abbildung 30: Subjektiver Fragebogen

Die fünf Fragen des Fragebogens dienten unterschiedlichen Zwecken.

Mit der ersten Frage wurde erhoben, ob die Testteilnehmer(in) die eben erlebte Rich Internet Application (RIA) bereits zuvor bei ihrer bisherigen Internetnutzung kennen gelernt hatte. Es ist wichtig diese Vorkenntnisse zu berücksichtigen, da es wesentlich leichter ist mit einer bestimmten RIA umzugehen wenn man bereits in der Vergangenheit Lernerfahrungen mit ihr machen konnte.

Die zweite Frage kann als Schlüsselfrage betrachtet werden: Wies der Prototyp Eigenschaften auf, die es dem oder der Testteilnehmer(in) erschwert haben, die Aufgabe zu lösen? Prototypen sollten eine Benutzerschnittstelle so simulieren, dass sich Benutzer ganz auf ihre Aufgabe und die dazu notwendigen Bildschirme konzentrieren können. Stehen dabei inhärente Eigenschaften der Prototypen-Methode im Wege, ist sie nur eingeschränkt geeignet, eine Benutzerschnittstelle zu simulieren. Wie wir später noch aufzeigen werden, war der Bezug dieser Frage auf die Form und nicht auf den Inhalt des Prototyps für die Testteilnehmer sehr missverständlich und nur schwer zu trennen (siehe erster Abschnitt des Kapitels 4.2).

Die letzten drei Fragen dienten einer Bewertung der Gütekriterien einer Prototypen-Methode. Ein Prototyp erfüllt dann vollumfänglich seinen Zweck, wenn er:

a) genauso aussieht wie das spätere fertige System

b) sich genauso verhält wie das spätere fertige System

c) genauso schnell auf Benutzereingaben reagiert wie das spätere fertige System

Mit den Fragen drei bis fünf des Fragebogens konnten Testteilnehmer diese Aspekte des Prototyps bewerten.

Stichprobenumfang

Der subjektive Fragebogen kam in beiden, der objektive Fragebogen nur im zweiten Untersuchungszeitraum zum Einsatz. Die Stichprobengrösse beträgt demnach für den subjektiven Fragebogen 15 Teilnehmer, für den objektiven 10 Teilnehmer. Der Grund, weshalb der objektive Fragebogen erst im zweiten Untersuchungszeitraum eingesetzt wurde, liegt in den Erfahrungen des ersten Untersuchungszeitraumes. Wir konnten wie beschrieben bereits bei den ersten fünf Testdurchgängen beobachten, wie die Teilnehmer Mühe hatten, die Bewertung des Prototyps von der Bewertung dessen Inhalte zu trennen. Deswegen wollten wir beginnend mit dem zweiten Untersuchungszeitraum eine zweite quantitative, von der subjektiven Bewertung der Teilnehmer unabhängige Datenreihe erheben.

4 Ergebnisse der Studie

4.1 Auswertung der Fragebogen

Nachfolgend wird die Auswertung der quantitativen Messwerte der objektiven und subjektiven Fragebogen beschrieben, die im Rahmen der Benutzertests zum Einsatz kamen (siehe Beschreibung der Fragebogen auf Seite 56 und Hypothesen auf Seite 33).

4.1.1 Auswertung der Vorkenntnisse der Testteilnehmer

Zwei Fragen möchten wir noch vor der Prüfung der Hypothesen beleuchten:

- Wie gut waren die Testteilnehmer mit den zehn RIA Patterns vertraut?
- Hatten Vorkenntnisse der Testteilnehmer über die getesteten RIA Patterns Auswirkungen auf die Bewertung der jeweiligen Prototypen?

Zur Erinnerung: Die erste Frage des subjektiven Fragebogens lautete: *„Ich kann mich daran erinnern, diese Art der Bedienung bereits früher schon einmal kennen gelernt zu haben.“* Die nachfolgende Auflistung gibt an, wie gut die Testteilnehmer Ihre Vorkenntnisse eingestuft haben. Der maximale Wert 5 steht dabei für *„stimme voll und ganz zu“*, der minimale Wert 1 steht für *„stimme gar nicht zu“*.

RIA	Name des RIA Patterns	Mittelwert der Vorkenntnisse	Anteil der Testteilnehmer mit hohen Vorkenntnissen (Wert 4 oder 5)
1.	Schieberegler (Slider)	2.67	42%
2.	Editierbare Tabelle	3.58	67%
3.	Rich Text Editor	4.00	75%
4.	Eingabevorschläge	5.00	100%
5.	Pulldown Menüs	4.92	100%
6.	Kontext Menüs	4.00	75%
7.	Kartenausschnitte verschieben	4.75	100%
8.	Drag und Drop	3.08	42%
9.	Sofortige Filterung von Suchergebnissen	4.17	83%
10.	Validierung von Eingaben	4.58	92%
	Insgesamt	4.08	78%

Tabelle 10: Vorkenntnisse der Testteilnehmer

Mit einem Mittelwert von 4.08 über alle RIA Patterns hinweg geben die Testteilnehmer im Mittel an, bereits früher einmal mit der jeweiligen RIA Pattern in Kontakt gekommen zu sein. Als Anteil ausgedrückt liegt der mittlere Anteil der Testteilnehmer, die mit der jeweiligen RIA vertraut waren, bei 78%.

Aus diesem Muster brechen nur wenige RIA Patterns aus. So waren die direkten grafischen Manipulationen auf dem Bildschirm via Schieberegler (Slider) und „Drag und Drop“ im Internetkontext bei unseren Testteilnehmern noch mehrheitlich unbekannt (je 42% Bekanntheitsgrad). Von der editierbaren Tabelle mit 67% Bekanntheitsgrad abgesehen waren mindestens drei von vier Testteilnehmern mit alle weiteren RIA Patterns vertraut.

Um die zweite Frage nach der potenziellen Auswirkung des Bekanntheitsgrades auf den Umgang mit der entsprechenden RIA zu beantworten, haben wir eine Korrelationsberechnung nach Spearman durchgeführt.

- Spearman-Rho Korrelation *Bedienung bereits kennen gelernt – Irgend etwas hat gestört* (subjektiv): $r_s = .01$, $p = .479$, $N = 120$
- Spearman-Rho Korrelation *Bedienung bereits kennen gelernt – Kam insgesamt gut zurecht* (objektiv): $r_s = .08$, $p = .237$, $N = 80$

Der Korrelationskoeffizient r_s gibt Auskunft darüber, wie eng der Zusammenhang ist zwischen dem Bekanntheitsgrad eines RIA Patterns und dem Grad, wie gut Testteilnehmer später (subjektiv oder auch objektiv) mit seiner Simulation in einem Prototyp zurecht kommen. Die Korrelationswerte können dabei von $r_s = .00$ (überhaupt kein Zusammenhang) bis $r_s = 1.00$ (perfekter Zusammenhang) reichen. Die ermittelten Werte sind in unserem Fall so tief, dass sich statistisch betrachtet Vorkenntnisse über RIA Patterns nicht darauf auswirken, wie gut Testteilnehmer mit dem entsprechenden Prototyp zurecht kommen.

4.1.2 Deskriptive Auswertung der Fragebogen

Mit den subjektiven und objektiven Fragebogen erhoben wir, inwieweit sich die vier eingesetzten Prototyp-Versionen auf die vorab definierten Messgrössen auswirkten. Die nachfolgende Grafik zeigt die Ergebnisse der objektiven Fragebogen, die von den Beobachtern während der Prototyping-Tests ausgefüllt wurden.

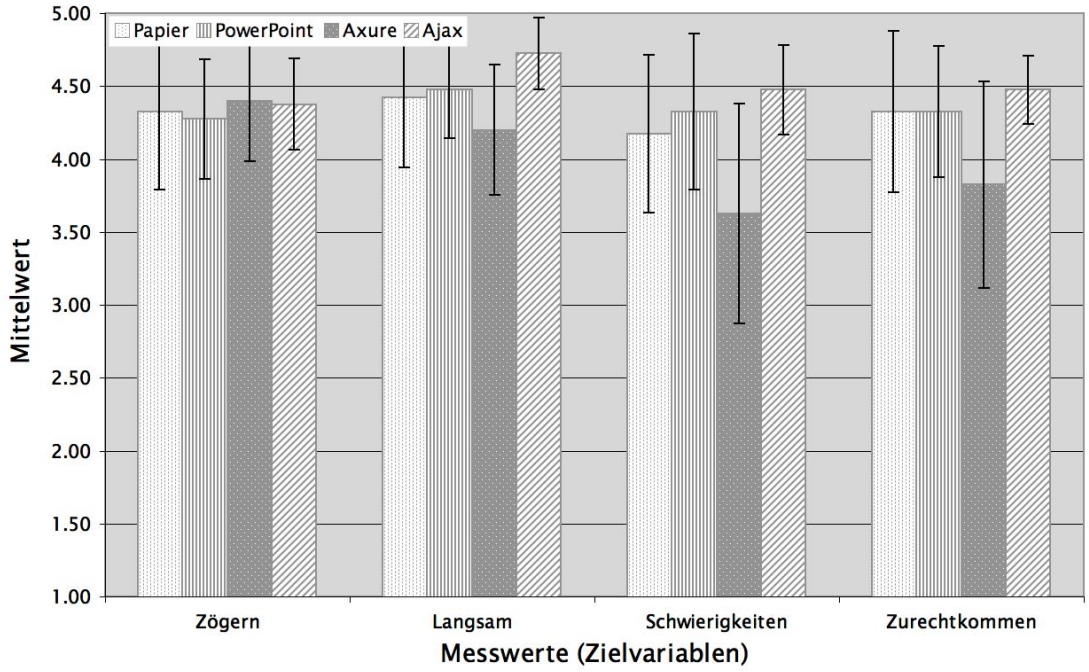

Abbildung 31: Mittelwerte der vier Zielvariablen des „objektiven“ Fragebogens (n=10) einschliesslich Fehlerbalken (95% Vertrauensintervall). Die Werte sind so gespiegelt, dass hohe Werte für positive Bewertungen des jeweiligen RIA Pattern Prototyps stehen.

Per Augenschein zögerten die Testteilnehmer bei der Bearbeitung der Testaufgaben über die vier Prototyp-Versionen gleich häufig. Ebenso ist die Geschwindigkeit, mit der sie die Aufgaben bearbeiteten, vergleichbar. Die Testteilnehmer schienen bei der Bearbeitung von

Axure RP Pro-basierten RIA Pattern Prototypen etwas mehr Schwierigkeiten gehabt zu haben als mit den anderen drei Prototyp-Versionen. Das gleiche Muster ist bei der Einschätzung der Beobachter zu erkennen, wie gut die Testteilnehmer insgesamt mit den Testaufgaben zurecht kamen. Auch hier war die Bewertung von Axure RP Pro-basierten RIA Pattern Prototypen scheinbar etwas weniger positiv als bei den anderen drei Prototyp-Versionen.

Trotz der hervorgehobenen Unterschiede fällt auf, dass die gemessenen Unterschiede zwischen den objektiven Messgrössen über die vier Prototyp-Versionen hinweg eher marginal sind. Hinzu kommt, dass die Streuung der Messwerte (erkennbar an den Fehlerbalken in der Grafik) erheblich ist. Damit wird die Bedeutung von Unterschieden im Mittelwert zusätzlich relativiert.

Im Vergleich dazu ergibt die deskriptive Auswertung der Messwerte ein differenzierteres Bild:

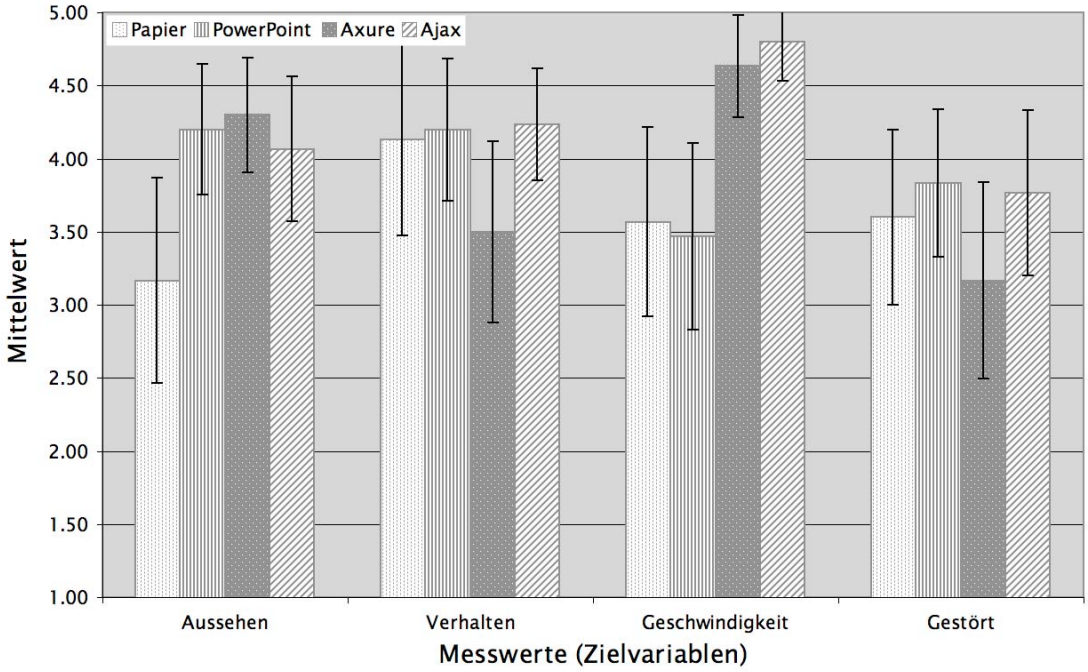

Abbildung 32: Mittelwerte der vier Zielvariablen des „subjektiven“ Fragebogens (N=15) einschliesslich Fehlerbalken (95% Vertrauensintervall). Die Werte sind so gespiegelt, dass hohe Werte für positive Bewertungen des jeweiligen RIA Pattern Prototyps stehen.

Das Aussehen des handgezeichneten Papier-Prototypen schien in den Augen der Testteilnehmer deutlich weiter von dem eines realisierten Angebotes entfernt zu sein als das der anderen drei Prototyp-Versionen. Bezüglich des Verhaltens blieb die Bewertung der Axure RP Pro-basierten RIA Pattern Prototypen hinter den anderen Prototyp-Versionen zurück. Die Geschwindigkeit der beiden papierbasierten Prototyp-Versionen wurde als langsamer wahrgenommen als bei den beiden elektronischen. Schliesslich fühlten sich die Testteilnehmer durch den Axure RP Pro-basierten RIA Pattern Prototypen etwas mehr gestört als bei den anderen drei Prototyp-Versionen.

Die Unterschiede in den Messgrössen des subjektiven Fragebogens fallen etwas grösser aus als in denen des objektiven Fragebogens. Dennoch ist auch bei den subjektiven Bewertungen eine sehr grosse Streuung der Messwerte festzustellen. Somit herrscht selbst

in den Fällen, in denen per Augenschein eine bedeutsame Differenz der Bewertungen auszumachen ist, wenig Einigkeit bezüglich dieser Unterschiede. Offensichtlich reagierten die Testteilnehmer stark unterschiedlich auf die beleuchteten Merkmale der vier Prototyp-Versionen.

4.1.3 Inferenzstatistische Prüfung der Hypothesen

Die Inferenzstatistik (siehe Glossar) dient dazu, Unterschiede bzw. Zusammenhänge zwischen verschiedenen Messgrössen gegen den Zufall abzugrenzen. Sie hilft bei der Bewertung, ob Unterschiede bzw. Zusammenhänge in den Daten statistisch auffällig (signifikant) oder durch zufällige Varianz (siehe Glossar) in den Messwerten zu erklären sind.

Die Prüfung der acht Hypothesen erfolgte in jeweils zwei Schritten. Im ersten Schritt wurden deskriptive Statistiken wie der Mittelwert und die mittleren Ränge berechnet. In einem weiteren Schritt wurde inferenzstatistisch untersucht, ob zwischen der unabhängigen Variable (der Prototyp-Version) und der jeweiligen abhängigen Variablen ein statistisch bedeutsamer Zusammenhang besteht.

Mehrere Voraussetzungen für die Berechnung von einfaktoriellen Varianzanalysen waren verletzt. So waren die Aussagen zu den verschiedenen Prototyp-Versionen in unserem Versuchsdesign voneinander abhängig, sämtliche Verteilungen der acht geprüften abhängigen Variablen waren nach einem *Kolmogorov-Smirnov Anpassungstest* (siehe Glossar) nicht normalverteilt, und in sechs von acht Fällen waren nach einem *Levene-Test* (siehe Glossar) die Varianzen über die Gruppen der unabhängigen Variablen hinweg nicht homogen. Deswegen haben wir uns entschieden, für die inferenzstatistische Prüfung der Unterschiedshypothesen verteilungsfreie Berechnungsverfahren anzuwenden, insbesondere den *Friedman-Test* (siehe Glossar).

Der Friedman-Test prüft *„die Nullhypothese identischer Populationen für abhängige Daten. Er entspricht damit der einfaktoriellen Varianzanalyse mit wiederholter Messung, macht aber wie der Kruskal-Wallis Test* [siehe Glossar] *keine Voraussetzungen bezüglich der Verteilung der Populationen und verwendet nur ordinale Informationen in den Daten.“* (Kruskal-Wallis Test, 2008, S.3). Der Friedman-Test beruht auf einem Vergleich zwischen der relativen Bewertung der Prototyp-Versionen bezüglich der jeweiligen Zielvariablen und einem zufallsbasierten Erwartungswert. Beträgt die Wahrscheinlichkeit p des empirischen (beobachteten) Bewertungsmusters maximal 5% ($p <= .05$), so spricht man von einem Ergebnis, das signifikant vom Zufall abweicht.

Hypothese 1 (Nullhypothese)

Bei der Aufgabenbearbeitung zögern Testteilnehmer bei den vier Prototypen-Versionen gleich stark.

Die Hypothese wurde bestätigt. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede:
$\chi^2(3, n = 10) = .33, p = .96$.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	10	4.33	2.65
PowerPoint	10	4.28	2.55
Axure	10	4.40	2.45
Ajax	10	4.38	2.35

[a]Stichprobenumfang [b]Hohe Werte stehen für wenig Zögern

Tabelle 11: Mittelwerte und Mittlere Rangreihen bei „Zögern bei Aufgabenbearbeitung“ (objektiv)

Hypothese 2 (Nullhypothese)

Testteilnehmer bearbeiten die Testaufgaben mit den vier Prototypen-Versionen gleich schnell.

Die Hypothese wurde bestätigt. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede:
$\chi^2(3, n = 10) = 2.01, p = .57$.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	10	4.43	2.70
PowerPoint	10	4.48	2.45
Axure	10	4.20	2.10
Ajax	10	4.73	2.75

[a]Stichprobenumfang [b]Hohe Werte stehen für schnelle Aufgabenbearbeitung

Tabelle 12: Mittelwerte und Mittlere Rangreihen bei „Langsam bei Aufgabenbearbeitung“ (objektiv)

Hypothese 3 (Nullhypothese)

Testteilnehmer haben bei der Bearbeitung der Testaufgaben mit den vier Prototypen-Versionen gleich viele Schwierigkeiten.

Die Hypothese wurde bestätigt. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede: $\chi^2(3, n = 10) = 3.21, p = .36$. Der per Augenschein ausgemachte Unterschied bei den Messwerten lässt sich somit nicht mit ausreichender Sicherheit gegen den Zufall abgrenzen.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	10	4.18	2.70
PowerPoint	10	4.33	2.50
Axure	10	3.63	1.95
Ajax	10	4.48	2.85

[a]Stichprobenumfang [b]Hohe Werte stehen für wenig Schwierigkeiten bei der Aufgabenbearbeitung

Tabelle 13: Mittelwerte und Mittlere Rangreihen bei „Schwierigkeiten bei Aufgabenbearbeitung“ (objektiv)

Hypothese 4 (Nullhypothese)

Testteilnehmer kommen bei der Bearbeitung der Testaufgaben mit den vier Prototypen-Versionen gleich gut zurecht.

Die Hypothese wurde bestätigt. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede: $\chi^2(3, n = 10) = 2.16, p = .54$). Auch hier kann die Inferenzstatistik den ersten Eindruck der deskriptiven Auswertung nicht bestätigen.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	10	4.33	2.95
PowerPoint	10	4.33	2.40
Axure	10	3.83	2.15
Ajax	10	4.48	2.50

[a]Stichprobenumfang [b]Hohe Werte stehen für gutes Zurechkommen bei der Aufgabenbearbeitung

Tabelle 14: Mittelwerte und Mittlere Rangreihen bei „Zurechtkommen mit Aufgabenbearbeitung“ (objektiv)

Hypothese 5 (Nullhypothese)

Testteilnehmer fühlen sich bei der Bearbeitung der Testaufgaben durch die vier Prototypen-Versionen gleich stark gestört.

Die Hypothese wurde bestätigt. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede: $\chi^2(3, n = 15) = 2.48, p = .48$). Die Unterschiede sind auch hier nicht ausreichend gross, als dass der Ersteindruck inferenzstatistisch bestätigt werden kann.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	15	3.60	2.50
PowerPoint	15	3.83	2.73
Axure	15	3.17	2.10
Ajax	15	3.77	2.67

[a]Stichprobenumfang [b]Hohe Werte stehen für wenig Störung durch den Prototypen bei der Aufgabenbearbeitung

Tabelle 15: Mittelwerte und Mittlere Rangreihen bei „Störung durch Prototypen“ (subjektiv)

Hypothese 6 (Alternativhypothese)

Testteilnehmer empfinden, dass der per Hand erstellte Papierprotoyp weniger wie ein fertiges System aussieht als die anderen drei Prototypen-Versionen.

Die Hypothese wurde verworfen. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede: $\chi^2(3, n = 15) = 6.56, p = .09$). Wenn auch die Unterschiede der Messwerte in die theoretisch begründete Richtung bewegen sind die Unterschiede aufgrund ihrer grossen Schwankungsbreite nicht eindeutig genug.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	15	3.17	1.83
PowerPoint	15	4.20	2.80
Axure	15	4.30	2.83
Ajax	15	4.07	2.53

[a]Stichprobenumfang [b]Hohe Werte stehen für Prototypen, deren Aussehen sich vom späteren System wenig unterscheidet

Tabelle 16: Mittelwerte und Mittlere Rangreihen bei „Aussehen des Prototyps“ (subjektiv)

Hypothese 7 (Nullhypothese)

Testteilnehmer empfinden das Verhalten der vier Prototypen-Versionen als gleich nahe zu dem des späteren Systems.

Die Hypothese wurde bestätigt. Zwischen den mittleren Rängen der Prototyp-Versionen bestehen gemäss eines *Friedman*-Tests keine bedeutsamen Unterschiede: $\chi^2(3, n = 15) = 2.70, p = .44$). Das Muster setzt sich fort: Das Verhalten von Axure RP Pro wird nicht ausreichend konsistent und stark als unterschiedlich wahrgenommen, als dass man aus statistischer Sicht von einem bedeutsamen Unterschied sprechen könnte.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	15	4.13	2.60
PowerPoint	15	4.20	2.70
Axure	15	3.50	2.10
Ajax	15	4.23	2.60

[a]Stichprobenumfang [b]Hohe Werte stehen für Prototypen, deren Verhalten sich vom späteren System wenig unterscheidet

Tabelle 17: Mittelwerte und Mittlere Rangreihen bei „Verhalten des Prototyps“ (subjektiv)

Hypothese 8 (Alternativhypothese)

Testteilnehmer empfinden die beiden Papier-Prototypen langsamer als die beiden elektronischen Prototypen.

Die Hypothese wurde bestätigt. Die mittleren Ränge der vier Prototyp-Versionen unterscheiden sich gemäss eines *Friedman*-Tests bedeutsam voneinander: $\chi^2(3, n = 15) = 15.97, p < .01$). Paarweise *Wilcoxon*-Vergleichstests zwischen den vier Prototyp Versionen ergaben, dass der per Hand erstellte Papierprotyp als deutlich langsamer empfunden wurde als die mit Axure bzw. Ajax erstellen Protypen ($Z = -2.00, p = 0.05$ bzw. $Z = -2.84, p < .01$). Das gleiche gilt für den Vergleich des PowerPoint Prototypen mit Axure bzw. Ajax ($Z = -2.38, p = .02$ bzw. $Z = -3.01, p < .01$). Diese achte Hypothese ist somit die einzige, die den Eindruck eines Unterschiedes der Messwerte per Augenschein bestätigt.

Prototyp-Version	N[a]	Mittelwert[b]	Mittlerer Rang[b]
Papier	15	3.57	2.03
PowerPoint	15	3.47	1.83
Axure	15	4.63	2.87
Ajax	15	4.80	3.27

[a]Stichprobenumfang [b]Hohe Werte stehen für Prototypen, deren Reaktionsgeschwindigkeit sich vom späteren System wenig unterscheidet

Tabelle 18: Mittelwerte und Mittlere Rangreihen bei „Geschwindigkeit des Prototyps“ (subjektiv)

Hypothese 9 (Zusammenhangshypothese)

Es besteht ein Zusammenhang zwischen dem Ausmass, in dem sich Testteilnehmer an einer Prototyp-Version stören und ihrem Aussehen, Verhalten und Geschwindigkeit.

Die Hypothese wurde bestätigt. Nach einer *Spearman-Rho* Korrelationsberechnung gibt es signifikante Zusammenhänge zwischen dem Ausmass, in dem sich Testteilnehmer an einer Prototyp-Version stören und deren Aussehen ($r_s = .25$, $p < .01$), Verhalten ($r_s = .63$, $p < .01$) und Geschwindigkeit ($r_s = .20$, $p = .01$).

Über alle Hypothesen hinweg wird sehr deutlich, dass der erste Eindruck der Auswertung der deskriptiven Messwerte nur in Ausnahmefällen Entsprechungen in der inferenzstatistischen Prüfung findet. So liegt der einzige statistisch abgesicherte Unterschied zwischen den vier Prototyp-Versionen in der geringeren Reaktions-geschwindigkeit der papierbasierten Prototypen im Vergleich zu den elektronischen Prototypen. Erstaunlicherweise wurde hingegen von den Testteilnehmern das Aussehen der handgezeichneten Prototypen als nicht bedeutsam „unfertiger“ als das der elektronisch hergestellten Prototypen empfunden.

Nach der Auswertung der quantitativen Fragebogendaten werden im nachfolgenden Unterkapitel die Erkenntnisse aus den qualitativen Beobachtungsnotizen vorgestellt. Sie beschreiben anschaulich die Grundlage der Bewertungen der Testteilnehmer und der Beobachter.

4.2 Auswertung der Beobachtungsnotizen

Bereits während der Usability Tests konnten wir beobachten, dass zwei Arten von Problemen auftraten:

- solche, die sich auf inhärente Eigenschaften der jeweiligen Prototyping-Methode zurückführen liessen (z.B. „unübersehbare Systemreaktionen“ bei den papierbasierten Prototypen),
- solche, die sich auf Missverständnisse in der Aufgabenstellung oder Gestaltungsmerkmale der Prototypen bezogen (so vermissten beispielsweise manche Testteilnehmer die Bezeichnung „Franken“ bei der Skala des Schiebereglers, um seine Funktion schneller einordnen zu können)

Wir haben deshalb die beobachteten Probleme in diese zwei Bereiche geteilt.

Da wir als Ziel unserer Arbeit herausfinden wollten, ob primär die Grenzen der Prototyping Methode die Ergebnisse bei RIA Patterns beeinflussen, haben wir uns in der Auswertung auf den ersten Bereich konzentriert. Die zweite Gruppe von Problemen werteten wir nicht weiter aus.

Die direkt durch die Prototypen ausgelösten Erkenntnisse konnten wir folgendermassen gruppieren:

- Reaktionsgeschwindigkeit des Systems
- Visuelle Wiedergabetreue: Erkennbarkeit und Aufforderungscharakter
- Verhalten des Systems
- eigentliche "Bugs", also Programmier- bzw. Herstellungsfehler

4.2.1 Reaktionsgeschwindigkeit des Systems

Einigermassen vorhersagbar war der Umstand, dass manche Prototypen zu langsam sein könnten. Etwas unerwartet war die Beobachtung, dass ein System auch zu schnell reagieren kann. Dies im Sinn, dass Änderungen auf dem Bildschirm so schnell passierten, dass sie von den Testteilnehmern nicht wahrgenommen wurden. Eine zusätzliche Feedbackmeldung hätte in diesen Fällen geholfen. Bei der Erstellung der Prototypen waren wir aber davon ausgegangen, dass bereits die Änderung der Anzeige an sich die Rückmeldung über die Änderung des Systemzustandes sein würde.

Reaktionsgeschwindigkeit zu langsam

RIA	Pattern Name	Prototyp-Art
1.	Schieberegler (Slider)	▪ Papier
3.	Rich Text Editor	▪ PowerPoint
4.	Eingabevorschläge (Type & Suggest)	▪ Papier ▪ PowerPoint ▪ Ajax (beim 1. Durchgang durchgehend, beim 2. Durchgang punktuell)
5.	Pulldown Menüs	▪ PowerPoint
8.	Drag und Drop	▪ Ajax (beim 1. Durchgang)
10.	Validierung von Eingaben	▪ PowerPoint

Tabelle 19: RIA Patterns, die durch zu langsame Systemreaktionen beeinträchtigt waren

Wie die Auflistung klar zeigt, waren die papierbasierten Prototypen nicht durchgehend zu langsam und auch nicht im gleichen Mass beim Papier- wie auch beim PowerPoint Prototypen.

Beim "Schieberegler" Pattern lag der Unterschied an der unterschiedlichen Bauweise des Prototyps: Beim handgezeichneten Prototypen wurde jeder einzelne Kartenpunkt einzeln geklebt, während bei der PowerPoint Version die ganze Karte mit den vorgedruckten Kartenpunkten ausgetauscht wurde.

Bei den Patterns "Rich Text Editor" und "Pull Down Menüs" lässt sich der Unterschied nur durch unterschiedliche Erwartungen der Testteilnehmer erklären, da die Funktionsweise der beiden Papierbasierten Prototypen identisch war.

Bei diesen drei Patterns bedeutete "langsam", dass viele Hantierungen erfolgten, welche zeitraubend waren und dem Testteilnehmer eine Wartezeit aufzwangen. In diesen Fällen wurden diese Wartezeiten von den Testteilnehmern negativ kommentiert und bewertet.

Eine andere Art von "zu langsam" kam von der verzögerten Reaktion des Systems. Sowohl der menschliche Operator als auch das ausprogrammierte System zeigte bei den Patterns "Eingabevorschläge" und "Drag und Drop" die Reaktion entweder zu spät oder gar nicht. D.h. die Reaktion nützte dem Testteilnehmer nichts mehr, weil er den ganzen Eintrag

bereits selber getippt hatte, oder die fehlende Reaktion erweckte beim Testteilnehmer den Eindruck, dass er soeben eine falsche Handlung ausgeführt hatte, weil sie nichts auslöste.

Reaktionsgeschwindigkeit zu schnell

RIA	Pattern Name	Prototyp-Art
9.	Sofortige Filterung von Suchergebnissen	▪ Axure ▪ Ajax

Tabelle 20: Patterns, die durch zu schnelle Systemreaktionen beeinträchtigt waren

Beim Pattern "Sofortige Filterung von Suchergebnissen", änderte sich die Anzeige der Suchresultateliste. Es wurden also je nach Filterkriterium mehr oder weniger Zeilen angezeigt. Diese Änderung erfolgte so schnell, dass bei kleineren Änderungen (nur eine Zeile mehr oder weniger), die Testteilnehmer diese gar nicht wahrnahmen. Da es keinerlei andere Rückmeldungen gab, führte dies zu etlicher Verwirrung bei den Testteilnehmern. Einige änderten das Filterkriterium oder klickten auf "Back", um zu sehen, ob sie eventuell eine Änderung übersehen hatten. Andere Testteilnehmer kommentierten zwar ihre Unsicherheit "ist da jetzt etwas passiert?", zögerten kurz, fuhren dann aber in der Aufgabenbearbeitung weiter.

Bei den Papier-Prototypen kam dieses Problem nicht vor, da der Operator nie unbemerkt eine Manipulation vornehmen konnte.

4.2.2 Visuelle Wiedergabetreue

Die Rede ist hier nicht von Design-Entscheidungen, sondern von Prototyp-inhärenten Einflüssen auf das Erscheinungsbild

Erkennbarkeit, Lesbarkeit

RIA	Pattern Name	Prototyp-Art
2.	Editierbare Tabelle	▪ Axure
3.	Rich Text Editor	▪ Papier ▪ Ajax
4.	Eingabevorschläge (Type & Suggest)	▪ Papier
8.	Drag und Drop	▪ Papier

Tabelle 21: Patterns, die durch nicht erkennbare Bildschirmelemente beeinträchtigt waren

Bei der "Editierbaren Tabelle" verdeckte das Kalender-Popup wichtige Informationen.

Beim "Rich Text Editor" waren die Icons des Papier-Prototyps schlecht erkennbar. Dies hatte zwar nur bei einem Testteilnehmer einen Einfluss auf die Lösung der Aufgabe (Zögern bei der Auswahl), wurde aber von weiteren Testteilnehmern in der Nachbesprechung kommentiert. Die Icons wurden von den anderen eher aufgrund der Erfahrung mit dem Gesamtlayout erkannt, als anhand der einzelnen Zeichnung.

Bei den weiteren Nennungen (Patterns 4 und 8) erschien den Testteilnehmern die handgeschriebenen Texte als "unprofessionell". Da diese Erkenntnisse auf je einer Nennung beruhen, geht es dabei aber vermutlich weniger um das konkrete Pattern, als um handgeschriebenen Text an sich.

Aufforderungscharakter

RIA	Pattern Name	Prototyp-Art
2.	Editierbare Tabelle	▪ PowerPoint
6.	Kontext-Menüs	▪ PowerPoint ▪ Ajax
7.	Kartenausschnitte verschieben	▪ PowerPoint ▪ Axure
8.	Drag und Drop	▪ PowerPoint ▪ Ajax
9.	Sofortige Filterung von Suchergebnissen	▪ Papier

Tabelle 22: RIA Patterns, die von schlechtem Aufforderungscharakter beeinträchtigt waren

Diese Erkenntnisse hängen stark damit zusammen, dass die entsprechende Funktion auf einer Webseite nicht erwartet wurde, oder aber in einer anderen Art als sie im Prototypen umgesetzt wurde.

Während aber weder editierbare Tabellen, Rechtsklick, Drag und Drop noch die sofortige Filterung (ohne Dialogabfrage) auf dem Web erwartet werden, haben generell eher die elektronischen Prototypen die Testteilnehmer dazu angeregt, einfach auf gut Glück etwas zu versuchen. Dort, wo dennoch bei Ajax oder Axure ein Zögern entstand, lag es daran, dass der Cursor seine Form nicht zu einer Hand gewechselt hat. Dies erweckte bei den Testteilnehmern den Eindruck, dass eine Interaktion nicht möglich war.

Beim Papier - wo solche visuellen Hinweise generell nicht vorkamen - wurde öfters zuerst beim Versuchsleiter nachgefragt, was der Testteilnehmer denn jetzt tun solle oder ob eine bestimmte Handlung erwartet werde, bzw. zum Erfolg führe ("Kann ich hier draufklicken?").

Das Problem bei "Drag und Drop" mit Ajax war ein Spezialfall. Zur Lösung der Aufgabe war ein Löschen aus dem Warenkorb eigentlich nicht nötig. Der Prototyp gefiel aber mehreren Testteilnehmern so gut, dass sie begannen, damit herumzuspielen und verschiedene Möglichkeiten auszutesten. Dabei wurde die "Dropzone" nicht erkannt, also die definierte Fläche, wohin man Elemente aus dem Warenkorb ziehen musste, um sie zu löschen.

4.2.3 Verhalten

In diese Gruppe fallen eigentlich verschiedene Erkenntnisse:

- unerwartetes Verhalten aufgrund anderer Erwartungen der Testteilnehmer
- unerwartetes Verhalten, weil der Prototyp das gewünschte Verhalten nicht unterstützt
- unklare Aufteilung der Rollen von System und Testtteilnehmer

Andere Erwartungen der Testteilnehmer

RIA	Pattern Name	Prototyp-Art
1.	Schieberegler (Slider)	▪ Papier ▪ PowerPoint
2.	Editierbare Tabellen	▪ Ajax
4.	Eingabevorschläge (Type & Suggest)	▪ Axure
5.	Pulldown Menüs	▪ PowerPoint
7.	Kartenausschnitte verschieben	▪ PowerPoint ▪ Axure
8.	Drag und Drop	▪ Papier

Tabelle 23: Patterns, deren Verhalten für die Testteilnehmer unerwartet war

Beim Pattern „Schieberegler (Slider)“ deckte sich die Erwartung des Testteilnehmers, ob auf einen Klick nur einer oder beide Teile des Schiebereglers an den neuen Ort springen sollen, nicht mit der Implementation durch das System.

Die editierbare Tabelle verhielt sich nicht wie eine Excel Tabelle (Springen von Feld zu Feld per Tabulator war nicht unterstützt). Dies wurde vom Testteilnehmer bemängelt.

Beim Pattern „Eingabevorschläge“ konnte die Dropdown-Auswahl nicht per Cursortaste angewählt werden. Dies widersprach der Erwartung des Testteilnehmers.

Das Pattern „Pulldown Menü“ ahmte ein echtes Rollover zu vollständig nach und zeigte in verschiedenen Zwischenschritten alle Zwischenstadien, die bei einer elektronischen Version üblicherweise in einem Schritt ausgeführt werden.

Beim Pattern „Kartenausschnitte verschieben“ erwartete der Testteilnehmer wegen Google Maps, dass auch Zoomen (z.B. durch Aufziehen eines Rechtecks) oder Klicken (verschieben der Karte, so dass der geklickte Punkt zentriert wird) möglich sei.

Bei "Drag und Drop" erwartete der Testteilnehmer nicht, dass der Gegenstand, der physisch in den Warenkorb gezogen worden war, nun vom System entfernt und durch einen Listeneintrag ersetzt wurde.

All diese unterschiedlichen Erwartungen der Testteilnehmer hinderten diese zwar nicht daran, die Aufgabe zu lösen. Sie führte aber zu schlechterer Bewertung und verbalen oder mimischen Ausdrücken von Irritation seitens der Testteilnehmer.

Prototyp unterstützt Verhalten nicht

RIA	Pattern Name	Prototyp-Art
1.	Schieberegler (Slider)	▪ Axure
2.	Editierbare Tabelle	▪ Axure
3.	Rich Text Editor	▪ Axure
7.	Kartenausschnitte verschieben	▪ Axure
8.	Drag und Drop	▪ Axure
10.	Validierung von Eingaben	▪ Axure

Tabelle 24: Unerwartetes Verhalten: Prototyp unterstützt Verhalten nicht

Axure RP Pro unterstützt in keiner Weise ein Ziehen oder Ziehen und Loslassen von Bildschirmelementen. Bei allen Patterns, wo ein solches Verhalten nötig war, musste es durch ein anderes Verhalten vorgetäuscht werden. Zusätzlich kann es nicht erkennen, aus welcher Richtung der Mauszeiger kommt. Dadurch kam es bei den aufgelisteten Prototypen teilweise zu völlig unkontrollierbarem Verhalten.

Beim Schieberegler beispielsweise "sprang" das Eingabeelement weg vom Cursor, statt dass man es in eine Richtung ziehen konnte. Und da das Bildschirmelement nicht erkennen konnte, aus welcher Richtung der Cursor kam, sprang es vorprogrammiert immer nur in eine Richtung. An diesem Prototypen konnte die Aufgabe - wenn überhaupt - nur mit viel Geduld und nach mehreren Anläufen gelöst werden.

Auch Drag und Drop wurde dadurch zu einer extrem schwierig zu lösenden Aufgabe. Die zu schiebenden Elemente bewegten sich bei Annährung durch den Cursor nach unten. Das resultierte einerseits in Elementen, die unten wieder aus dem Warenkorb herauskamen, andererseits in Elementen, die sich von selbst bewegten, wenn der Cursor zufällig in die Nähe kam.

Bei der Editierbaren Tabelle trat ein Problem nur auf, wenn sich der Testteilnehmer nicht an eine vorausgesagte und entsprechend gescriptete Reihenfolge hielt. Nach der Vorhersage sollte der Testteilnehmer zuerst den ganzen Text schreiben und anschliessend Elemente auswählen und durch Klick auf ein Icon formatieren. Wenn der Testteilnehmer entsprechend vorging, verhielt sich der Prototyp wie erwartet.

Wenn der Testeilnehmer jedoch zuerst auf das Formatier-Icon klickte und dann erst den Text schreiben wollte, erschien der ganze, korrekt formattierte Text wie von Zauberhand. Dies wurde teilweise als Prototyp-Fehler erkannt, teilweise aber auch irrtümlich als Auto-Complete Funktion interpretiert und kritisiert.

Validierung von Eingaben fiel in ein ähnliches Gebiet. Wenn Testteilnehmer begannen, längere Texte in die Textbox zu schreiben - was nicht vorgesehen war - brach der Text nicht um sondern lief rechts zur Box hinaus.

Kartenausschnitte verschieben stellte keine grösseren Probleme. Es galt nur die Einschränkung, dass durch Scrollbalken navigiert werden musste, nicht durch Ziehen der Karte. Manchen Testteilnehmern fiel dies nicht einmal auf.

Unklare Rollenverteilung

RIA	Pattern Name	Prototyp-Art
8.	Drag und Drop	▪ Papier
10.	Validierung von Eingaben	▪ PowerPoint

Tabelle 25: Unerwartetes Verhalten: unklare Rollenverteilung

Nachdem der Testteilnehmer verstanden hatte, welche Handgriffe das System vornehmen würde, begann er, diese Handgriffe selber auszuführen. Beispielsweise legte er selber Fehlermeldungen hin oder nahm sie wieder weg. Er kam damit dem Operator zuvor, dessen Reaktionsgeschwindigkeit ungenügend war, dies zu verhindern.

Bei den Testteilnehmern führte dies weder zu einer schlechteren Bewertung noch hinderte es bei der Lösung der Aufgabe. Es ist aber ein Phänomen, das nur bei Papier-Prototypen überhaupt auftreten kann.

4.2.4 Fehlverhalten

RIA	Pattern Name	Prototyp-Art
2.	Editierbare Tabelle	▪ Papier ▪ Ajax
3.	Rich Text Editor	▪ Axure
7.	Kartenausschnitte verschieben	▪ Papier ▪ PowerPoint

Tabelle 26: Unerwartetes Verhalten aufgrund von Fehlverhalten

Es gab auch echte Programmier- bzw. Erstellungsfehler. Bei der Editierbaren Tabelle zeigte eine neue Zeile beim Papier-Prototyp nicht alle Zusatzelemente (z.B. Kalender). Bei Ajax hingegen zeigte die neue Zeile Werte wie "null" oder " ". Ausserdem akzeptierte ein Textfeld nicht genügend Zeichen, um den ganzen verlangten Text hineinzuschreiben.

Axure zeigte beim Prototypen zu Pattern "Rich Text Editor" überlappende Bilder.

Im ersten Durchgang funktionierte der Papier-Prototyp für Kartenausschnitte verschieben überhaupt nicht. Der Kartenausschnitt liess sich nicht verschieben bzw. bei PowerPoint nur sehr schlecht.

4.3 Zusammenfassung der Testergebnisse

In diesem Abschnitt möchten wir die zentralen Testergebnisse beleuchten – erst aus der Sichtweise der quantitativen Daten, dann aus der der qualitativen Daten.

4.3.1 Quantitative Datenauswertung

Die quantitative Auswertung der Fragebogendaten hatte zum Ziel, systematische Auswirkungen der Prototyp-Versionen auf definierte Zielvariablen aufzudecken. Dabei ist zu berücksichtigen, dass die Wirkung der zehn RIA Patterns, die der Messung zugrunde lagen, ausgemittelt wurde. Die quantitative Auswertung kann somit keine zuverlässigen Aussagen darüber treffen, wie sich bestimmte RIA Patterns in bestimmten Prototyp-Versionen auswirken. Dazu ist die qualitative Beobachtung und Befragung besser in der Lage.

Die Auswertung des objektiven Fragebogens konnte keine systematischen Auswirkungen der Prototyp-Versionen auf die Zielvariablen feststellen. Testteilnehmer arbeiteten mit allen vier Versionen von sich aus gleich flüssig und zögerten bei der Aufgabenbearbeitung vergleichbar häufig. Ebenso hatten die Testteilnehmer bei der Bearbeitung Testaufgaben gleich viele Schwierigkeiten und kamen ähnlich gut zurecht, egal ob sie die Testaufgaben mit den Papier-, PowerPoint-, Axure oder Ajax basierten Prototypen bearbeitet haben.

Der durch die Teilnehmer selbst ausgefüllte, subjektive Fragebogen ergab ein ähnliches Bild. So wurden das Verhalten und erstaunlicherweise sogar das Aussehen der vier Prototyp-Versionen als gleichartig empfunden. Die verschiedenen Prototyp-Versionen hatten zudem vergleichbar viele Eigenschaften, welche die Testteilnehmer als störend empfanden. Hingegen waren die langsameren Reaktionen der beiden papierbasierten Prototypen im Vergleich zu den beiden elektronischen statistisch bedeutsam.

Im Vergleich zu einem fertigen System nahmen die Teilnehmer viele Prototypen als unfertig hinsichtlich Aussehens, Verhalten und Geschwindigkeit war. Am stärksten wurde das Nutzenerlebnis beeinträchtigt, wenn sich die Prototyp-Versionen anders als erwartet

verhielten. Nicht ganz so stark, aber immer noch spürbar wirken sich ein unfertiges Aussehen und eine langsame Reaktionszeit der Prototypen-Version aus.

Zusammenfassend zeigte die Auswertung der quantitativen Messwerte so marginale Unterschiede zwischen den vier Prototypen-Versionen auf, dass keine von ihnen bei der Simulation von RIA Patterns eine systematische Überlegenheit aufweisen konnte.

Nun zu den zentralen Ergebnissen der qualitativen Beobachtungs- und Befragungsdaten.

4.3.2 Qualitative Datenauswertung

Reakionsgeschwindigkeit des Systems, Feedback

Wie auch die quantitative Auswertung zeigt, wurde die Geschwindigkeit bei den Papier-basierten Prototypen generell als langsamer beurteilt. Meist drückten die Testteilnehmer aber in den Kommentaren aus, dass sie dies nicht weiter gestört hätte.

Die Testteilnehmer haben von sich aus bereitwillig Kunstpausen für den Operator eingelegt, wenn sie erkannten, dass seine Reaktionen zu langsam waren. Im Unterschied dazu erkannten die Testteilnehmer beim ausprogrammierten Prototypen oft nicht, dass überhaupt etwas hätte passieren sollen, wenn er zu langsam war.

Analog dazu nahmen die Testteilnehmer kein Feedback wahr, als beim ausprogrammierten Prototypen die sofortige Filterung von Suchergebnissen zu schnell erfolgte. Wenn ein Operator involviert ist, kann ein Feedback gar nicht übersehen werden, auch wenn er verspätet reagiert.

Visuelle Wiedergabetreue: Erkennbarkeit und Aufforderungscharakter

In einigen wenigen Fällen wurde handschriftlicher Text als „unprofessionell“ moniert, nicht aber handgezeichnete Eingabeelemente wie Buttons oder Tabellen. Generell wurden auch die Papier-Prototypen spontan als „sehen aus wie ein echtes System“ beurteilt, teilweise mit einer späteren Einschränkung, dass dies nicht wörtlich zu verstehen sei. Bei den elektronischen, nie aber bei den papierbasierten Prototypen wurde manchmal bemängelt, dass das Design etwas „kahl“ sei, Farben fehlen und generell das Drumherum sehr spartanisch sei.

Ganz selten wurde eine Schwierigkeit beim Lösen der Aufgabe auf schlechte Lesbarkeit der Handschrift oder fehlende Farbigkeit bei den Papier-Prototypen zurückgeführt.

Mängel beim Aufforderungscharakter gabe bei allen Prototypen. Während es aber bei den elektronischen möglich wäre, Interaktivität durch dynamische visuelle Hinweise anzuzeigen, ist diese Möglichkeit bei den papierbasierten Prototypen nicht vorhanden.

Bei Unsicherheiten bezüglich des Aufforderungscharakters war die Experimentierfreudigkeit bei den elektronischen Prototypen generell einiges grösser.

Verhalten

Jedes unerwartete Verhalten führte zu Irritationen. Wenn das Verhalten aber so weit abwich, dass es der Computererfahrung, bzw. einem echten System widersprach, führte dies in einzelnen Fällen dazu, dass eine Aufgabe gar nicht gelöst werden konnte. Einige Testteilnehmer versuchten, das Verhalten des Prototypen zu verstehen, andere gaben auf. Alle aber reagierten heftig - mit Verärgerung oder Belustigung.

Fehlverhalten aufgrund von Bugs müssten in einer nächsten Iteration korrigiert werden. Sie führten aber zu weit weniger Ärger als systemimmanente Probleme, die durch Alternativlösungen wegen Beschränkungen des Werkzeugs verursacht wurden.

5 Diskussion der Ergebnisse

In diesem Kapitel besprechen wir die Testergebnisse des vorangegangenen Kapitels. Zu Beginn erfolgt eine Bewertung der Prototyping Methoden zur Simulation von RIA Patterns (5.1). Anschliessend geben wir Empfehlungen zu den Prototyping Methoden und deren Einsatz (5.2). Im letzten Abschnitt geben wir Hinweise, wie das Einsatzspektrum der Prototyping-Methoden möglichst weit reichend ausgenutzt werden kann (5.3).

5.1 Bewertung der Prototyping-Methoden zur Simulation von RIA

Um eine qualitative Bewertung der vier Prototyping Methoden vornehmen zu können, müssen zuerst die einzelnen Bewertungsaspekte und -kriterien definiert werden. Bewertet werden sollen alle relevanten und inhaltlich überschneidungsfreien Aspekte, die für die Erstellung und Verwendung von RIA Prototypen gelten.

Entwicklungsaufwand	Wir betrachten alle Ressourcen, die für die Erstellung der Prototypen notwendig sind. Dazu zählen Kriterien wie der Schulungs- und Einarbeitungsaufwand für die Anwendung der Methode, der Aufwand für die Erstellung des Prototyps und die Kosten für die eingesetzten Hilfsmittel (Büromaterial, Software etc.).
Wartungsaufwand	Wir bewerten, wie einfach ein Prototyp bei neuen oder veränderten Anforderungen angepasst oder erweitert werden kann. Beispiele von Kriterien sind die Änderbarkeit von einzelnen Interaktionselementen oder globalen Elementen.
Ästhetik	Wir bewerten die Prototyping Methoden im Hinblick auf Möglichkeiten zur visuellen Ausarbeitung des Prototyps. Kriterien sind beispielsweise die Ähnlichkeit mit dem finalen System, die Konsistenz der Bildschirmelemente oder der wahrgenommene Aufforderungscharakter der Interaktionselemente.
Interaktivität	Die Kriterien in dieser Kategorie bewerten die Prototyping Methoden in Bezug auf die Möglichkeiten, die dynamischen und interaktiven Aspekte von RIA zu simulieren. Dazu zählen die Reaktionsfreudigkeit bei Eingaben des Benutzers, die Simulation von Maus- und Tastatureingaben, die Möglichkeiten, Teilbereiche des Bildschirms zu aktualisieren oder den Systemstatus zu visualisieren.

5.1.1 Per Hand erstellte Papier-Prototypen

Entwicklungsaufwand

Die Papier-Prototypen waren handgezeichnete Skizzen, die bewusst einfach gehalten waren. Die Entwicklung konnte entsprechend schnell (ca. 9 Stunden) und ohne grosse Vorbereitung umgesetzt werden. Damit die Prototypen mit Benutzern evaluiert werden konnten, mussten wir uns im Vorfeld sorgfältig überlegen, welche Fälle bei einem Test eintreffen können und diese entsprechend vorbereiten. Ebenfalls war es ratsam, die Prototypen so gut als möglich zu modularisieren, um den Erstellungsaufwand zu minimieren. Statt für jeden möglichen Zustand eine eigene Seite zu zeichnen, wurden die

Informations- und Eingabeelemente grösstenteils als eigene Papierfragmente auf ein Blatt mit dem Seitenlayout gelegt. Dies entsprach auch der die Dynamik von RIA Patterns in dem Sinne, dass einzelne Bildschirmelemente ersetzt werden können ohne die gesamte Seite neu zu laden. Grundlegende methodische Kenntnisse sind für die Erstellung der Papier-Prototypen erforderlich, diese können aber innert weniger Stunden angeeignet werden. Bei einigen der Patterns war der Ablauf der Interaktion gut vorhersehbar. Die Prototypen für die Pulldown- und Context-Menus oder jener für das Verschieben des Kartenausschnitts mussten z.B. nur eine einfache Interaktion des Benutzers unterstützen und erforderten auch nicht die Darstellung von unterschiedlichen Zuständen. Bei anderen Prototypen, wie beispielsweise „Schieberegler“ oder „Validierung von Eingaben“, hatte der Benutzer mehr Freiräume wie die Aufgabe gelöst werden konnte. Entsprechend mussten mehrere mögliche Zustände berücksichtigt werden. Generell kann gesagt werden, dass der Aufwand für die Vorbereitung und Erstellung eines handgezeichneten Papier-Prototypen mit der Komplexität der Interaktion und der Anzahl der möglichen Zustände zunimmt. Da für die Entwicklung der Papier-Prototypen gängige Büroutensilien wie Papier, Schere, Stifte, Post-It Zettel und Klebeband verwendet wurden, sind die Kosten für die Umsetzung nahezu vernachlässigbar.

Wartungsaufwand

Trotz der Modularisierung haben einige der Änderungen zwischen der ersten und der zweiten Iteration ein komplettes Neuzeichnen von einigen Prototypen erfordert. Dies speziell dann, wenn der Basisbildschirm geändert hat, beispielsweise aufgrund von Textänderungen. Ebenfalls mussten einige Prototypen neu gezeichnet werden, weil Testteilnehmer im ersten Durchgang mit Kugelschreiber hineingeschrieben hatten. Mit 6 Stunden lag der Aufwand für die Korrekturen bei zwei Drittel des initialen Erstellungsaufwandes.

Ästhetik

Trotz dem stark aufs Wesentliche reduzierte Detaillierungsgrad gab es wenige Probleme, die auf die visuelle Ausarbeitung zurück zu führen waren. Kleinere Schwierigkeiten gab es einzig bei den Prototypen mit grafischen Elementen. Einige der Testpersonen hatten Mühe, die Icons des Rich Text Editors zu erkennen oder die Produktbilder bei der Drag und Drop Aufgabe zu unterscheiden. In diesem Zusammenhang wurden von wenigen Testpersonen die fehlenden Tooltips auf den Icons bemängelt. Der Aufforderungscharakter der anderen Interaktionselemente war nicht spürbar schlechter als bei den Prototypen, die mit anderen Techniken erstellt wurden. Der handgeschriebene Text wurde zwar in wenigen Fällen als unprofessionell beanstandet, stellte aber ansonsten kein Problem dar.

Interaktivität

Die Dynamik und Interaktivität von RIA liess sich nur mit der Hilfe des Operators simulieren. Das automatische Auffrischen von einzelnen Bereichen des Bildschirms, bedingt durch eine Aktion des Benutzers, verliert durch den Eingriff des Operators an Authentizität und wirkt sich teilweise störend auf den Interaktionsfluss aus. Die Frage, ob eine Veränderung des User Interfaces vom Benutzer wahrgenommen wird, oder ob ein explizites Feedback notwendig ist, konnte mit einem Papier-Prototypen kaum beantwortet werden. Auch die Veränderung des Mauszeigers bei bestimmten Operationen (z.B. die Visualisierung von möglichen Zielen bei einer Drag und Drop Operation) liess sich nicht simulieren.

Da dem Benutzer für die Interaktion lediglich ein Stift und seine Finger zur Verfügung standen, konnten nicht alle Maus- und Tastaturereignisse simuliert werden. Operationen wie ein Rechtsklick zum Öffnen des Kontextmenus oder das Zoomen von Karten durch den Einsatz des Scrollrades waren damit nicht möglich, bzw. nur mit begleitendem verbalem Kommentar des Testteilnehmers. Bei der Simulation von Drag und Drop

Operationen wurde erfreulicherweise von allen Testteilnehmern der Zeigefinger intuitiv verwendet, um die Papierschnitzel auf dem Papier zu verschieben. Eine weitere Limitation der Papier-Prototypen war die Simulation von realistischen Antwortzeiten. Die Zeit, die vom Operator benötigt wurde, um bei einer Aktion des Benutzers eine Aktualisierung des Prototyps vorzunehmen, war in der Regel länger als die effektive Antwortzeit des fertigen Systems.

5.1.2 Mit PowerPoint erstellte Papier-Prototypen

Entwicklungsaufwand

Der Aufwand für die Erstellung der PowerPoint-Prototypen war mit 13 Stunden etwa ein Drittel höher als bei den Papier-Prototypen. Dies lag unter anderem aber auch daran, dass die PowerPoint Prototypen vor den handgezeichneten erstellt wurden und einige Male geändert werden mussten. Der Mehraufwand, der für das Zeichnen der einzelnen Bildschirmelemente benötigt wurde, konnte zum grossen Teil durch die Möglichkeit, Objekte zu kopieren, wieder wettgemacht werden. Statt jeden Bildschirm so zu modularisieren, dass beliebige Zustände durch die Anordnung von kleinsten Einzelteilen zusammengestellt werden konnte, war es mit PowerPoint einfach, mehrere Ausprägungen von möglichen Zuständen vorzubereiten. Die Aufgabe für den Schieberegler sah beispielsweise vor, dass für drei vordefinierte Preisbereiche jeweils eine Anzahl von Hotels auf der Karte und in der Liste angezeigt wird. Mit drei komplett vorbereiteten Karten und Listen konnten also bereits die wahrscheinlichsten Zustände abgedeckt werden. In diesen Fällen konnte der Operator die gesamten Bereiche austauschen, was gegenüber einer manuellen Anordnung der Marker auf der Karte und der Einträge in der Liste weit weniger fehleranfällig und zeitaufwändig war. Die notwendigen PowerPoint Kenntnisse waren in unserem Fall gegeben. Durch die Einfachheit von PowerPoint können aber auch Novizen mit geringem Einarbeitungsaufwand mit dem Tool umgehen. Als Bestandteil der Microsoft Office Suite ist PowerPoint ebenfalls sehr verbreitet.

Wartungsaufwand

Durch die digitale Speicherung konnten sehr einfach mehrere Versionen erstellt und ausgedruckt werden. Ein Neuzeichnen eines ganzen Prototyps war in keinem Fall notwendig. Die Änderungen zwischen den zwei Durchgängen konnten in etwa 6 Stunden umgesetzt werden. Dies entsprach in etwa der Hälfte des Initialaufwands bzw. der gleichen Zeit, die für die Änderungen an den handgezeichneten Papier-Prototypen benötigt wurde.

Ästhetik

Die mit PowerPoint erstellten Wireframes unterschieden sich im Detaillierungsgrad bewusst kaum von den handgezeichneten Skizzen. Der Vorteil bestand darin, dass graphische Elemente wie Schriften, Karten oder Icons in besserer Qualität dargestellt werden konnten. Dies förderte bei den Prototypen die Lesbarkeit und Wahrnehmung.

Interaktivität

Da die Prototypen ebenfalls auf Papier getestet wurden, waren sie in gleichem Masse interaktiv wie die handgezeichneten Skizzen. Durch den Umstand, dass einige der Zustände als ganze Elemente vorbereitet werden konnten, erhöhte sich bei gewissen Prototypen die Antwortzeit bei einer Benutzerinteraktion signifikant, was sich positiv auf den Testfluss auswirkte.

5.1.3 Mit Axure RP Pro erstellte digitale Prototypen

Entwicklungsaufwand

Um den ganzen Funktionsumfang von Axure auszureizen, war eine gewisse Einarbeitungszeit notwendig. Während die Erstellung von statischen Wireframes sehr intuitiv und unkompliziert ist, erfordert die Umsetzung von Dynamik und Interaktivität etwas mehr Übung. Da Axure aber keine speziellen Programmierkenntnisse voraussetzt, kann die gesamte Logik über visuelle Editoren bearbeitet werden. Der Programmcode für die Regeln wird in einer Pseudo-Code-ähnlichen Sprache generiert, welche mit zunehmender Erfahrung auch direkt editiert werden kann. War die Lernkurve einmal genommen, konnten die meisten Prototypen relativ schnell umgesetzt werden. Bei der Umsetzung der Prototypen "Drag und Drop" und "Navigation in Karten" gab es allerdings unüberwindbare technische Probleme und es musste relativ viel Zeit aufgewendet werden, um eine alternative funktionstüchtige Lösung zu implementieren. Um den Entwicklungsaufwand tief zu halten wurden auch nur die Elemente, die für das Szenario wichtig waren, interaktiv gestaltet. Für die Einarbeitungs- und Umsetzungszeit wurden knapp 40 Stunden benötigt. Eine Einzellizenz von Axure RP Pro kostet USD 589 bzw. USD 539 ab 5 Lizenzen. Dieser Preis erscheint für den gebotenen Funktionsumfang mehr als gerechtfertigt.

Wartungsaufwand

Durch die Verwendung von Master-Objekten können globale Vorlagen von Informations- und Eingabeelementen oder Layouts erstellt werden. Diese Master Objekte können zentral verwaltet und geändert werden und erleichterten so die Wartung von mehrfach verwendeten Elementen. Zusätzlich wird eine globale “Search and Replace“ Funktion angeboten, um Texte und Labels auf einer Seite oder im ganzen Dokument zu ersetzen. Die Erweiterung eines Prototyps um zusätzliche Seiten ist ebenfalls sehr einfach möglich. Der Aufwand für die Änderungen belief sich auf etwa 10 Stunden, wobei die Optimierung der „Spezialfälle“ den grössten Teil der Zeit in Anspruch nahm.

Ästhetik

Axure RP Pro eignet sich vor allem für die Erstellung von interaktiven Wireframes. Eine pixelgenaue Darstellung ist aufgrund der limitierten Zeichnungsfunktionen nur annähernd möglich. Mit der Einbindung von Bitmaps können zwar auch Prototypen mit höherem visuellen Detaillierungsgrad erstellt werden, dafür muss aber ein zusätzliches Tool wie beispielsweise Photoshop eingesetzt und erheblich mehr Zeit aufgewendet werden. Tabellen werden grundsätzlich mit einem einfachen schwarzen Raster gezeichnet und die Formularelemente werden als normale Betriebssystem Widgets ausgegeben. Leider bietet Axure keine Unterstützung für eigene CSS Stylesheets, um das Aussehen der Elemente im generierten HTML Output zu verändern. Da in unserem Fall die mit Axure erstellten Prototypen in etwa die gleiche Detaillierung wie die Papier-Prototypen aufweisen sollten, stellte diese Einschränkung für uns kein Problem dar. Für einige Elemente konnten wir Bitmaps aus den Ajax Prototypen übernehmen, z.B. der Toolbar des Rich Text Editors oder die Produktebilder des Drag und Drop Prototyps.

Interaktivität

Axure bietet einige Möglichkeiten um die Prototypen mit Dynamik und Interaktivität zu zu ergänzen. Die Widgets unterstützen einige Maus- und Tastaturereignisse, um bei einer Interaktion des Benutzers eine Aktion auszuführen. So kann z.B. auf einem Texteingabefeld ein linker Mausklick (OnClick), ein Tastaturanschlag (OnKeyUp) oder das Erhalten und Verlassen des Eingabefokus (OnFocus, OnLostFocus) abgefangen werden. Eine Region einer Imagemap unterstützt die Ereignisse linker Mausklick (OnClick), Maus fährt in die Region (OnMouseEnter) und Maus verlässt die Region (onMouseOut). Diese

Ereignisse können jeweils mit einer oder mehreren Aktionen verknüpft werden, so beispielsweise das Aktivieren/Deaktivieren des Widgets, das Springen auf eine andere Seite oder das Anzeigen/Verbergen eines so genannten dynamischen Panels. Dynamische Panels sind Bereiche des Bildschirms die für verschiedene Zustände gestaltet werden können. Mit den Funktionen, die Axure bietet, konnten 8 der 10 Prototypen problemlos umgesetzt werden. Unlösbare Probleme gab es einzig beim Schieberegler und bei Drag und Drop, weil die notwendigen Ereignisse für das Gedrückthalten und Ziehen (OnMouseMove, OnMouseUp) von Objekten mit der Maus nicht unterstützt werden. Die umgesetzten Alternativlösungen entsprachen in keiner Weise dem erwarteten Verhalten und erwiesen sich bei den Benutzertests als unbrauchbar. Bei zwei weiteren Prototypen musste die Interaktion leicht anders gelöst werden. Das Kontextmenu konnte aufgrund der fehlenden Rechtsklick-Unterstützung nur mit einem Linksklick aktiviert werden und das Navigieren in Karten musste mit horizontalen und vertikalen Scrollbalken gelöst werden.

5.1.4 Mit Ajax Technologien ausprogrammierte digitale Prototypen

Entwicklungsaufwand

Für die Entwicklung von Ajax Prototypen werden Kenntnisse in verschiedenen Web-Technologien vorausgesetzt. Dazu zählen vor allem HTML, Cascading Style Sheets (CSS), JavaScript und das Document Object Model (DOM). Des Weiteren ist die Erfahrung in einem Ajax-Framework sinnvoll, damit die asynchrone Server-Kommunikation und die User Interface Widgets nicht selbst programmiert werden müssen. Je nach Anforderung an die Datenqualität des Prototyps ist auch die Programmierung von Server-seitigen Skripten z.B. in PHP und die Ablage der Daten in einer Datenbank wie mySQL notwendig. Von den meisten der benutzten Technologien war bereits ein gutes bis sehr gutes Know-how vorhanden. Neu war lediglich das mehrheitlich verwendete Ajax-Framework „Prototype“ und die Google Maps Programmierschnittstelle. Die für die Umsetzung der Prototypen notwendigen Kenntnisse konnten in einigen Stunden erarbeitet werden. Für einige der Patterns fanden wir auch Beispielprogramme im Internet, die für unsere Zwecke angepasst oder konfiguriert werden konnten. Der Aufwand für die Umsetzung der 10 Prototypen belief sich auf etwa 30 Stunden. Ohne das vorhandene Basiswissen hätte die Erstellung allerdings ein Mehrfaches der Zeit in Anspruch genommen. Da alle verwendeten Tools und Libraries als Open Source Software verfügbar waren, gab es keinerlei finanziellen Aufwand.

Wartungsaufwand

Der Wartungsaufwand von Ajax Prototypen hängt sehr von der Art der Veränderungen ab. Während Anpassungen im Layout oder Design in der Regel rasch umzusetzen sind, können Änderungen in der Logik schnell viel Aufwand verursachen. Für die Überarbeitung der Prototypen nach der ersten Iteration mussten etwa 7 Stunden aufgewendet werden. Hilfreich sind ergänzende Tools wie z.B. eine Versionsverwaltung, ein Editor bzw. Entwicklungsumgebung mit globalen Refaktorisierungs-Möglichkeiten und ein JavaScript Debugger.

Ästhetik

Beim Detaillierungsgrad gibt es bei den Ajax Prototypen keinerlei Einschränkungen. Auch wenn ein ausprogrammierter Prototyp meistens in allen Belangen High-Fidelity ist, können je nach Anforderung und verfügbarer Zeit auch Abstriche in einer oder mehreren Dimensionen gemacht werden. So waren auch die von uns erstellten Ajax Prototypen in der visuellen Ausarbeitung auf das Wesentliche reduziert, um den Vergleich mit den anderen Methoden zu ermöglichen.

Interaktivität

Auch in Bezug auf Interaktivität ist grundsätzlich alles machbar, was auch im finalen System umgesetzt werden kann. Damit der Aufwand für die Erstellung des Prototyps aber kleiner ist, müssen Kompromisse in einer oder mehreren Dimensionen eingegangen werden. So ist es denkbar, dass nur Elemente entlang dem Test-Szenario interaktiv sind und andere z.B. als Bilder eingebunden werden. Ebenfalls ist die Simulation von Server-seitiger Programmlogik und Datenspeicherung auf dem Client möglich.

5.2 Empfehlungen zum Einsatz der Prototyping-Methoden

Die Erfahrungen, die wir im Rahmen unserer Untersuchung gemacht haben zeigen, dass das Prototyping von Rich Internet Anwendungen gegenüber traditionellen Web Applikationen höhere Anforderungen an die einzelnen Methoden stellt. Bei einigen Prototypen sind wir bei der Umsetzung an Grenzen gestossen und es konnten nicht immer befriedigende Lösungen erarbeitet werden. Wie wir bereits im vorhergehenden Kapitel feststellen konnten, haben alle Methoden Vor- und Nachteile.

Die Frage, welche Methode am besten für das Testen von Rich Internet Anwendungen geeignet ist, kann auch nicht pauschal beantwortet werden. Stattdessen hängt die Wahl von verschiedenen Faktoren ab:

Zeit und Kosten	Viele Projekte stehen unter einem grossen Zeit- und Kostendruck. Für das Erstellen von Prototypen bleibt oft nur wenig Zeit und auch die finanziellen Mittel sind beschränkt.
Projektphase	Da in der frühen Phase des Projekts noch nicht alle Anforderungen und technischen Details des Systems bekannt sind, sollte ein Prototyp möglichst schnell und billig erstellt werden, da die Wahrscheinlichkeit von Änderungen noch gross ist.
Ziel und Zielgruppe	Mit einem Prototyp wird ein bestimmtes Ziel verfolgt und eine spezielle Zielgruppe angesprochen. Ob beispielsweise erste Benutzertests durchgeführt werden sollen oder ein Produkt der Marketingabteilung vorgestellt werden muss, beeinflusst ganz wesentlich die Wahl der Prototyping Methode.
Vorhandene Kentnisse	Die Erstellung und Evaluation von Prototypen erfordert gewisse Fähigkeiten, die im Projektteam vorhanden sein müssen. Während die Anfertigung von Papier-Prototypen keine bis wenig technische Kenntnisse erfordert, müssen für einen voll interaktiven Prototypen in der Regel Programmierer einbezogen werden.
Lebensdauer	Je nach Methode haben die Prototypen eine unterschiedliche Lebensdauer. Während handgezeichnete Skizzen bewusst als Wegwerf-Prototypen entwickelt werden, können digital erstellte Prototypen über mehrere Iterationen hinweg angepasst und eventuell sogar in das finale System überführt werden.
Eignung für Benutzertests	Ein wichtiges Ziel bei der Erstellung von Prototypen ist die anschliessende Validierung mit Benutzern. Je nach Test-Methodik und Anspruch an die Qualität der Testergebnisse gibt es Aspekte die es zu berücksichtigen gilt.

5.2.1 Zeit und Kosten

Von Hand gezeichnete Prototypen sind bezüglich Zeitaufwand und Kosten unschlagbar. Sie können ohne Vorbereitung oder technische Infrastruktur erstellt werden, eignen sich aufgrund ihrer Einfachheit aber eher nur für erste Entwürfe von Teilen einer Anwendung. Bei grösseren Prototypen ist der Einsatz eines Tools wie PowerPoint, Keynote, Visio oder OmniGraffle ratsam, da mehrfach verwendete Elemente oder ganze Versionen kopiert und leichter angepasst werden können. Mittlerweile gibt es auch einige preisgünstige Alternativen zur Erstellung von Wireframes (z.B. Balsamiq oder MockupScreens). Interessant sind auch Service Angebote wie Jiffy oder Protoshare, die für eine monatliche Gebühr von mehreren Benutzern über das Internet genutzt werden können. Die Anschaffung eines Prototyping-Tools wie Axure lohnt sich, wenn mehrfach Prototypen erstellt und diese über mehrere Projektphasen verfeinert und dokumentiert werden sollen. Der Einarbeitungs- und Zeitaufwand für die Erstellung von RIA Prototypen mit Axure hängt sehr stark davon ab, wie dynamisch und interaktiv sich der Prototyp verhalten soll. Die Umsetzung von ausprogrammierten Prototypen benötigt am meisten Zeit, bedingt aber oft keine Softwarekosten, da viele der Tools und Frameworks „Open Source“ sind und kostenlos eingesetzt werden können.

Der zeitliche und finanzielle Aufwand für die Erstellung eines Prototyps korreliert mit seinem Detaillierungsgrad. Je ähnlicher der Prototyp im Aussehen und Verhalten dem finalen System gleicht, desto grösser ist der Erstellungsaufwand. Die Prototyping Methode sollte darum so gewählt werden, dass sie dem Zeitplan und Projektbudget, sowie dem für die Projektphase notwendigen Grad an Interaktivität und Dynamik entspricht.

5.2.2 Projektphase

Die Papier-Prototypen haben aufgrund der tiefen Entwicklungskosten und der schnellen Realisierbarkeit den grossen Vorteil, dass Ideen sehr schnell visualisiert und in kurzer Zeit mehrere Design Alternativen evaluiert werden können (Tohidi, Buxton, Baecker, Sellen, 2006). Da die Erstellung eines ausprogrammierten Prototyps aufwändiger ist, wird oft nur eine Lösung entwickelt. Diese Variante wird dann als kostbarer angesehen und der Entwickler ist weniger bereit, aufgrund von kritischem Feedback Änderungen zu machen (Wong, 1992).

Die Einfachheit von handgezeichneten oder mit PowerPoint erstellten Papier-Prototypen hilft auch, dass sich die Benutzer auf das Interaktionsdesign und die Informationsarchitektur konzentrieren und sich nicht in Diskussionen über visuelle Details verlieren. Bei den Benutzertests hat sich herausgestellt, dass die handgezeichneten Prototypen in Bezug auf das Aussehen nicht signifikant schlechter bewertet wurden als die interaktiven Prototypen. Vor allem die interaktiven Ajax Prototypen wurden mit einem viel strengeren Massstab gemessen als die auf Papier getesteten Prototypen. So wurde einige Male bemängelt, dass das Design unvollendet wirkt, eine Aussage die bei den Papier-Prototypen kein einziges Mal zu hören war. Dies zeigt, dass die Erwartung des Benutzers dem verwendeten Medium angepasst wird.

Papier-Prototypen sind ebenfalls weniger limitierend in den technischen Möglichkeiten und fördern die Kreativität des Designs. RIA Entwicklungsumgebungen und Frameworks zwingen dem Entwickler gewisse Einschränkungen und Eigenheiten bei der Umsetzung des User Interfaces auf und schränken so die Kreativität und mögliche Design-Alternativen ein. Wenn beispielsweise ein Ajax Framework für das Zoomen von Karten einen Schieberegler vorsieht, ein anderes einen Linksklick und ein drittes das Scrollrad der Maus, wird die Lösung gezwungenermassen durch die Wahl des Frameworks diktiert. Berücksichtigt man im Vorfeld eine breitere Auswahl von Alternativen führt dies eher zu einem effektiven Design, auch wenn am Ende der Entwickler standardisierte Interface Widgets einsetzt (Beaudouin-Lafon & Mackay, 2003).

Die Eignung von papierbasierten Prototypen in frühen Projektphasen ist unbestritten (z.B. Rudd et al., 1996; Sefelin, Tscheligi, Giller, 2003; Liu, Khooshabeh, 2003; Snyder, 2003; Tohidi et al., 2006). Auch unsere Ergebnisse haben bewiesen, dass sich von Hand oder mit PowerPoint erstellte Prototypen eignen, um qualitative Ergebnisse über die Benutzbarkeit von RIA Entwürfen zu erhalten. Die Erstellung von ausprogrammierten Prototypen lohnt sich in der Regel erst in späteren Projektphasen, wenn die grundlegendsten Design Entscheide bereits gefällt wurden. Axure kann über mehrere Projektphasen eingesetzt werden und grösstenteils Papier & Bleistift oder PowerPoint für die Erstellung von Papier-Prototypen ersetzen, sowie ansatzweise auch als Alternative zu einem ausprogrammierten Prototyp eingesetzt werden.

Die folgende Graphik illustriert am Beispiel des Rational Unified Process (RUP), in welchen Projektphasen die Prototyping Methoden am effektivsten eingesetzt werden können.

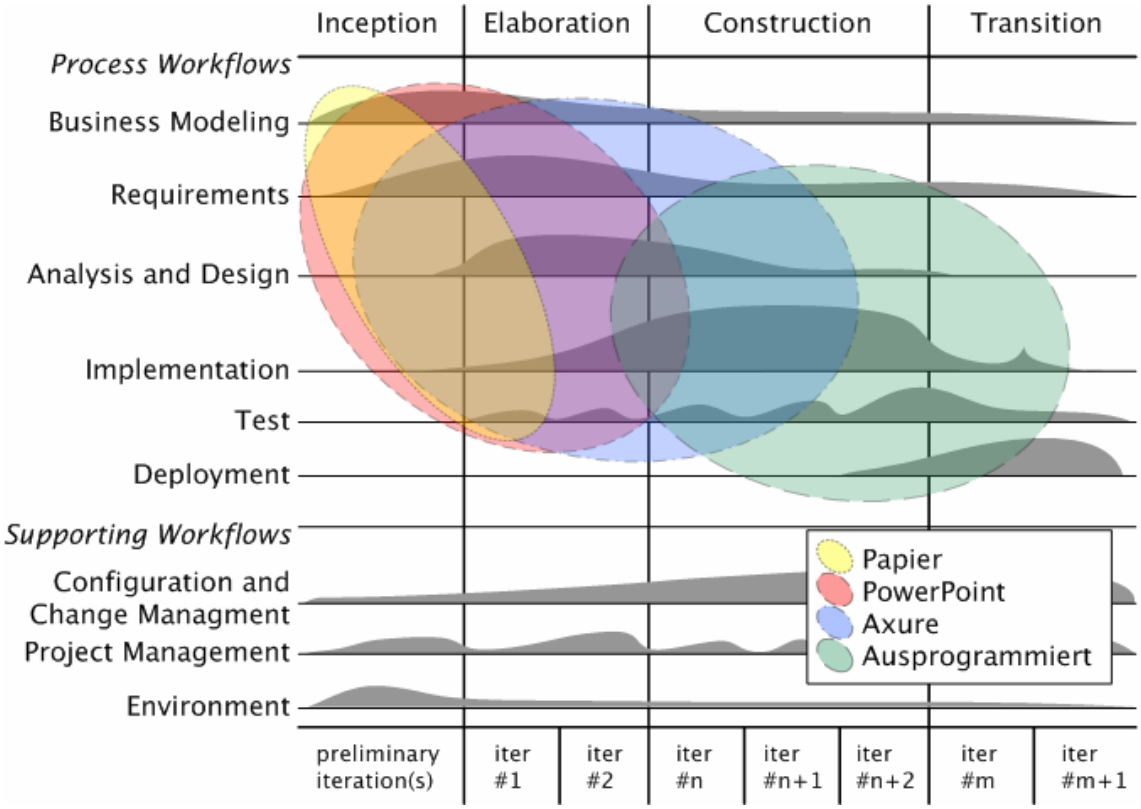

Abbildung 33: Einsatz der Prototyping Methoden im Projektverlauf

Da es sich bei RUP um ein iteratives Vorgehensmodell handelt, ist es durchaus möglich, dass innerhalb einer Iteration mehrere Methoden zum Einsatz kommen.

5.2.3 Ziel und Zielgruppe

Das Ziel, das mit dem Prototyp verfolgt wird, bzw. die Zielgruppe, die angesprochen werden soll, beeinflusst die Wahl der Prototyping Methode bedeutend. Von Hand oder mit PowerPoint erstellte Papier-Prototypen eignen sich vor allem für die Diskussion von Design Entwürfen innerhalb des Projektteams und qualitativen Erhebungen mittels begleitenden Walkthroughs. Für die Dokumentation der Anforderungen einer RIA sind Papier-basierte Prototypen nur bedingt geeignet. Ein Problem ist, dass die dynamischen Aspekte einer RIA mit einem statischen Papier-Prototyp nur schlecht dokumentiert werden können. Mögliche Ansätze für die Beschreibung sind Storyboards (Scott, 2005; Zapata, 2006) oder animierte Filme. Für die Dokumentation von Zustandsänderungen von RIA gibt es Vorschläge für eine spezielle graphische Notation (Cecil, 2007). Park et al. (2007)

beschreiben einen Vorschlag, wie der gesamte Design Prozess für RIA optimiert werden könnte. All diese Konzepte erfordern allerdings ein grosses Mass an schriftlicher Erläuterungen, damit der Prototyp selbsterklärend ist.

Demgegenüber visualisieren digitale Prototypen die Dynamik und Interaktivität von RIA durch die Anwendung selbst. Ausprogrammierte Prototypen von hohem Detaillierungsgrad eignen sich darum auch, in Ergänzung zu funktionalen Anforderungen, als Vorlage für die Entwickler von Software, Schulungsunterlagen oder Hilfetexten. Da der Prototyp meist schon Monate vor der Verfügbarkeit des finalen Systems existiert, kann die Funktionalität bereits der Öffentlichkeit (z.B. auf Messen) vorgestellt werden (Rudd et al., 1996).

Aufschlüsse über die technische Machbarkeit des User Interfaces liefern einzig die ausprogrammierten Prototypen, aber auch nur dann, wenn für die Entwicklung der Prototypen die gleichen technischen Rahmenbedingungen (Browser, RIA Plattform, Widget Library etc.) wie für das fertige System gelten. Wird der Protoyp beispielsweise mit Flash programmiert, das fertige System soll aber mit einem Ajax Framework implementiert werden, ist es schwierig, eine Aussage über die technische Realisierbarkeit und das Antwortzeitverhalten zu treffen.

5.2.4 Vorhandene Kenntnisse

Die Fähigkeiten die in einem Projektteam vorhanden sind, beeinflussen oft auch die Wahl der Prototyping Methode. Während Papier-Prototypen – ob von Hand gezeichnet oder mit einem einfachen Werkzeug wie PowerPoint erstellt – keine Programmierkenntnisse erfordern, sind für die Erstellung von ausprogrammierten RIA Prototypen technische Fertigkeiten unabdingbar. Sind diese Kenntnisse im Team nicht vorhanden und stehen keine externen Spezialisten zur Verfügung, ist es wenig effizient, sich das notwendige Wissen zuerst zu erarbeiten. Stattdessen sollte man eine Methode wählen, die problemlos gemeistert werden kann. Ein Prototyping-Tool wie Axure schliesst die Lücke zwischen Papier- und ausprogrammierten Prototypen und richtet sich speziell an Business Analysten, Interaktionsdesigner, Usability Experten, und Informationsarchitekten ohne Programmierkenntnisse. Ein Prototyping Tool wird allerdings immer hinter den technischen Möglichkeiten, welche die Browser und Browser Plugins bieten, zurückbleiben.

5.2.5 Lebensdauer

Handgezeichnete Papier-Prototypen haben gegenüber den digital erstellten Prototypen eine relativ kurze Lebensdauer und eignen sich nur bedingt für die Spezifikation von funktionalen Anforderungen. Veränderte Anforderungen haben in der Regel ein Neuzeichnen des ganzen oder von grösseren Teilen des Prototyps zur Folge. Von Hand erstellte Prototypen sind daher eher für kleinere Anwendungen oder Teilbereiche von grösseren Systemen geeignet. Papier-Prototypen die mit einem Tool wie PowerPoint erstellt worden sind haben den Vorteil, dass Änderungen und neue Versionen durch Kopier- und Ersetzen Funktionen schneller realisiert werden können und die Wireframes mit Kommentaren ergänzt in andere Dokumente eingefügt werden können.

Eine spezialisierte Software wie Axure unterstützt den Lebenszyklus eines Prototypen zusätzlich, indem für die einzelnen Screens und Widgets Anmerkungen und Kommentare eingefügt werden können und bei Bedarf ein komplettes Spezifikationsdokument inklusive verlinkten HTML Screens generiert werden kann. Digital erstellte Prototypen können immer auch elektronisch verbreitet werden, beispielsweise um Feedback von mehreren Stakeholdern einzuholen.

Ausprogrammierte Prototypen können ebenfalls über mehrere Iterationen hinweg verfeinert werden und am Ende der Prototyping Phase als Teil der Spezifikation der Entwicklung übergeben werden. Je nach Detaillierungsgrad des Prototyps können Vermassung, Farben, Labeling und das Verhalten teilweise bis vollständig dokumentiert

werden, so dass ergänzende Dokumente wie Wireframes oder ein Styleguide eventuell sogar überflüssig sind. Möglich ist auch ein evolutionäres Prototyping, wo der Prototyp schrittweise in das fertige System überführt wird. Dieser Ansatz wird vor allem auch von agilen Methoden wie Extreme Programming oder Scrum propagiert. Das Mehr an Zeit, dass für die Erstellung des Prototyps aufgewendet wurde, kann so durch ein kürzere Implementierungphase teilweise wieder wettgemacht werden. Axure generiert zwar auch HTML Code, dieser ist aber aufgrund seiner Verschleierung nicht für die Weiterverwendung geeignet.

5.2.6 Eignung für Benutzertests

Da bei einem Benutzertest die gesamte Interaktivität von einem Operator simuliert werden muss, eignen sich die Papier-Prototypen lediglich für begleitete Walkthroughs und Tests. Visuelle und technische Details von Rich Internet Anwendungen, die nicht simuliert werden können, müssen dem Benutzer erläutert werden. Die Methodik stellt dabei hohe Anforderungen an den Operator. Dieser muss die Funktionalität und das Verhalten des Systems genau kennen und alle Interaktionen des Benutzers genau interpretieren, um den Prototypen zu steuern. Aufgrund der hohen Beanspruchung des Operators ist es auch sinnvoll, einen zusätzlichen Moderator für den Benutzertest einzusetzen. Im Gegensatz zu seitenorientieren Anwendungen setzt sich die Seite bei Rich Internet Anwendungen dynamisch zusammen und kann je nach Typ der Seite beliebig viele Zustände annehmen. Dies erfordert bei komplexeren Interaktionen einige Eingriffe des Operators, was sich störend auf den Benutzer und den Interaktionsfluss auswirkt. Es gab auch vereinzelt Missverständnisse, z.B. wenn der Benutzer weitermachen wollte, bevor der Operator mit den Manipulationen fertig war oder unnötig wartete, obwohl er bereits wieder an der Reihe war. Der Operator verrät durch die Handgriffe auch, „dass etwas passiert“ und verfälscht so die effektive Wahrnehmung von Statusänderungen. Bei den Prototypen wo aufgrund einer Manipulation von Parametern eine Resultateliste gefiltert wurde, fiel einigen Benutzern bei den digitalen Prototypen die Zustandsänderung der Liste nicht auf, weil die Änderung so schnell und unmerklich ausgeführt wurde, dass die Statusmeldung nicht wahrgenommen wurde. Bei den Papier-Prototypen war die Veränderung für den Benutzer offensichtlich, weil durch den Eingriff des Operators beim Ersetzen der Vorlage ein deutliches Feedback erfolgte.

Für einen formalen Usability Test müssen die Prototypen zwingend digital sein. Papier-Prototypen eignen sich aufgrund ihrer fehlenden inhärenten Interaktivität nicht für eine selbstständige Anwendung durch den Benutzer. Ein mit Axure erstellter Prototyp generiert zwar einen klickbaren HTML Output, dieser kann aber nur mit Vorbehalten für einen Usability Test einer Rich Internet Anwendung verwendet werden. Wie bereits besprochen, mussten für einige Interaktionsmuster Alternativlösungen erarbeitet werden, welche nicht dem mentalen Modell des Benutzers entsprachen. Dies führte bei den Tests vereinzelt zu unüberwindbaren Problemen und erforderte teilweise die Hilfe des Moderators. Die Durchführung eines Usability Tests setzt also zwangsmässig auch ein realistisches Verhalten des Prototyps voraus. Abweichungen von der Ziellösung müssen vom Moderator angekündigt und das beabsichtigte Verhalten beschrieben werden.

Papier-Prototypen haben den Vorteil, dass sie portabel sind und ohne technische Infrastruktur präsentiert werden können. Digitale Prototypen setzen mindestens einen Computer mit einem Web Browser voraus, RIA, die mit einem Server kommunizieren um Daten auszutauschen, benötigen zusätzlich eine Internet-Verbindung. Auf der anderen Seite können Online Prototypen, die auf einem Web Server installiert sind, laufend aktualisiert werden und einer grossen Benutzergruppe (z.B. für Remote Usability Tests) zugänglich gemacht werden.

Digitale Prototypen haben auch den Vorteil, dass bei einem Test die Anwendung und der Benutzer mit einer Screen Capture Software synchron aufgezeichnet werden kann und so

eine nachträgliche einfache Auswertung möglich ist. Benutzertests von Papier-Prototypen können zwar mit einer Videokamera aufgenommen werden, doch dabei gehen gewisse Feinheiten aufgrund der schlechteren Bildauflösung und der ungünstigeren Kameraposition verloren.

Bei RIA spielt auch die “Rich User Experience” oder die “Joy of Use” eine wichtige Rolle. RIA machen mehr Freude in der Bedienung als ihre seitenbasierten Pendants. Mit einem Papier-Prototyp kann diese Gefühl nur sehr beschränkt vermittelt werden.

Der Frage, ob Papier-Prototypen gegenüber interaktiven Prototypen bessere Ergebnisse bei der Aufdeckung von Usability Problemen liefern, war nicht Bestandteil unserer Studie. Dieser Fragestellung wurde aber in einigen anderen Studien nachgegangen. Virzi, Sokolov und Karis (1996) untersuchten beispielsweise ob Low-Fidelity Prototypen auch in späteren Projektphasen genauso effektiv wie High-Fidelity Prototypen eingesetzt werden können. Es wurden zwei Anwendungen mit Hilfe von ausgedruckten Wireframes und interaktiven Prototypen getestet. Die Ergebnisse der Tests deckten bei beiden Techniken mehrheitlich die gleichen Usability Probleme auf. Damit wurde bewiesen, dass auch Low-Fidelity Prototypen in diesem Punkt genauso wirksam wie High-Fidelity Prototypen eingesetzt werden können. Die Studie räumt aber ein, dass vor allem für die Kommunikation mit Marketingfachleuten, Entwicklern und Personen die mit der Erarbeitung von Dokumentation und Schulungsunterlagen betraut sind, High-Fidelity Prototypen merkliche Vorteile haben.

In einer weiteren Studie von Walker et al. (2002) wurden zwei verschiedene Anwendungen in unterschiedlichem Detaillierungsgrad (Low- und High-Fidelity) auf zwei verschiedenen Medien (Papier und Computer) als Prototypen ausgearbeitet. Jeder Testteilnehmer testete jeweils zwei der möglichen 16 Kombinationen. Das Ergebnis der Studie zeigte keinen signifikante Unterschiede in Bezug auf die entdeckten Usability Probleme beim Vergleich zwischen Low- und High-Fidelity Prototypen. Die Anzahl der Kommentare war dagegen signifikant höher bei den am Computer getesteten Prototypen. Die Tatsache, dass die wenigen Kommentare zur Ästhetik nicht etwa bei den High-Fidelity Prototypen sondern bei den Low-Fidelity Prototypen geäussert wurden, leitet die Autoren zu der Hypothese, dass bei einem Task-basierten Usability Test der Fokus der Testteilnehmer auf Task-bezogene Informationen und Navigation gerichtet ist und weniger auf visuelle Details.

5.2.7 Übersicht der Bewertungskriterien

Die folgende Tabelle vergleicht die vier Prototyping Methoden anhand der weiter oben definierten Bewertungskriterien.

Legende:
- ++ sehr gut geeignet
- + geeignet
- - bedingt geeignet
- -- nicht geeignet
- n/a nicht anwendbar

	Papier	PowerPoint	Axure	Ajax
Entwicklungsaufwand				
Schulungs- und Einarbeitungsaufwand	++	+	-	--
Aufwand für die Erstellung des Prototyps	++	+	+	-
Verwendung von vorgefertigten Komponenten	-	+	+	++
Kosten für die eingesetzten Hilfsmittel	++	+	-	+
Wartungsaufwand				
Anpassung der Informationsarchitektur	-	+	+	-
Änderungen von einzelnen Interaktionselementen	+	+	++	+
Bearbeitung von globalen Elementen (Seitenlayout)	-	-	++	+
Erweiterung mit neuen Seiten	+	+	++	+
Änderung der Programmlogik	n/a[1]	n/a[1]	-	-
Ästhetik				
Ähnlichkeit mit dem finalen System	+	+	+	++
Konsistenz der einzelnen Bildschirmelemente	-	+	++	++
Wahrnehmung der Interaktionselemente (Aufforderungscharakter)	-	+	+	++
Entsprechen die Antwortzeiten des Prototyps dem fertigen System	-	-	+	++
Interaktivität				
Ähnlichkeit der Interaktion gegenüber dem finalen System	+	+	+	++
Sofortige Reaktion auf Benutzereingaben	-	-	+	++
Aktualisierung von Teilbereichen des Bildschirms	+	+	+	++

[1] Wird vom Operator („Computer“) während des Tests geändert

Direktmanipulation von Objekten	+	+	-	++
Simulation der linken Maustaste	+	+	+	++
Simulation der rechten Maustaste	-	-	--	++
Simulation von Drag und Drop	+	+	--	++
Simulation von Überlagerungen (Overlay)	++	++	+	++
Simulation von eingelegten Inhalten (Inlays)	-	-	-	++
Simulation von Tastatur-Kurzbefehlen	--	--	-	++
Navigation in Karten	+	+	-	++
Änderung des Mauszeigers zur Anzeige des Systemstatus	-	-	-	++
System-generierte Datenaktualisierung (Push)	-	-	--	++
Animation von Übergängen und Zustandsveränderungen	--	-	-	++
Verwendungszweck				
Visualisierung und Diskussion von Design-Ideen	++	+	+	-
Unsersuchen von mehreren Design-Alternativen	++	+	+	--
Dokumentation von Anforderungen	-	+	++	+
Spezifikation des User-Interfaces	-	+	+	++
Vorlage für Technical Writers und Ausbilder	-	+	+	++
Marketing-Showcase	--	-	-	++
Technischer Proof-of-Concept	--	--	-	++
Evolutionäres Prototyping	--	--	--	+
Eignung für Benutzertests				
Möglichkeit von ad-hoc Änderungen	++	+	-	-
Benötigte Infrastruktur	++	++	+	-
Aufzeichnungsmöglichkeiten	-	-	++	++
Usability Walkthrough	++	++	+	+
Usability Test	-	-	+	++
Remote Usability Test	--	--	+	++
Accessibility Test	--	--	-	++

Tabelle 27: Übersicht der Bewertungskriterien

5.3 Optimierung des Einsatzspektrums der Prototyping-Methoden

5.3.1 Papier-Prototyp

Ein Papier-Prototyp weist gewisse Einschränkungen auf, die nicht umgehbar sind: Es muss immer ein Operator anwesend sein, welcher die Interaktivität des Systems bereitstellt. Ein Prototyp kann daher auch immer nur mit einem Testteilnehmer gleichzeitig getestet werden.

Die Reaktionsgeschwindigkeit eines Menschen kann nie mit der eines Computers konkurrieren und auch die visuelle Wiedergabetreue ist extrem eingeschränkt. Ein schnelles und reibungsloses Funktionieren des Systems ist nur gegeben, wenn der Ersteller des Prototyps mögliche Benutzerhandlungen treffsicher vorhersagen konnte und der Operator das System gut kennt und alle Elemente gut organisiert bereitgelegt hat.

Dies bedeutet aber auch, dass Papier-Prototypen nicht an eine Infrastruktur, sondern nur an eine Person gebunden sind. Wo immer es die Möglichkeit gibt, Papier auszulegen, kann eine Evaluation durchgeführt werden.

Einschränkungen der Eingabe können sicher optimiert werden. Beispielsweise durch die Anweisung, einen Stift für einen Rechtsklick umzudrehen oder gezielt den Finger zu benutzen, wenn ein Element bewegt werden muss. Alternativ könnte auch nur der Finger als Mauszeiger zum Einsatz kommen und ein Stift nur für Tastatureingaben (inkl. Schreiben von Tastaturkürzeln). Ein Rechtsklick könnte dann mit verschiedenen Fingern (rechte Hand, linke Hand; Zeigefinger, Mittelfinger; etc.) ausgeführt werden.

Um versehentliches Beschreiben des Prototypen zu vermeiden, kann man einen Bleistift statt eines Kugelschreibers einsetzen oder den Prototypen mit Transparentfolie überdecken. Gesamthaft oder nur punktuell bei den Eingabefeldern. Für Eingabefelder kann man auch wiederablösbares Korrekturpapier verwenden.

Um die Abfolge von Benutzer- und Systemhandlungen zu klären, könnte der Operator dem Benutzer eine Sanduhr vorlegen - nicht nur bei längeren Wartezeiten, sondern auch, wenn der Benutzer beginnt, Systemhandlungen vorzunehmen.

Ungeschlagen ist sicher die Möglichkeit, auf der Stelle das Verhalten des Systems zu ändern oder Anpassungen am Interface zu machen, ohne dass die Änderung extrem ins Auge sticht. Beispielsweise mit der Schere Teile ad hoc wegschneiden, oder Teile zu überkleben, mit alternativen Texten zu versehen, etc.

Diese Möglichkeit, während oder zwischen Tests bereits Änderungen und Experimente am Interface vorzunehmen ist unter den getesteten Prototyping-Arten konkurrenzlos.

5.3.2 PowerPoint

Obwohl die genannte Möglichkeit der ad hoc Änderung auch für Papier-Prototypen zutrifft, die in PowerPoint erstellt wurde, kann hier die Änderung nicht "unauffällig" erfolgen. Handgeschriebene und handgezeichnete Änderungen werden zwischen einem Computerausdruck immer als "Flickwerk" auffallen.

Einfacher als beim handgezeichneten Papier-Prototyp können aber Varianten erstellt und vorbereitet werden. So kann mit relativ geringem Aufwand eine vergleichende Evaluation durchgeführt werden.

Mehr Potential sehen wir aber in digitalen Prototypen, die mit einer Präsentationssoftare erstellt werden. Hier könnten Präsentationsübergänge als dynamische Elemente verwendet werden bzw. zum Vortäuschen von Änderungen nur von Teilen des Bildschirms.

Dadurch könnte zuerst ein Papier-Prototyp erstellt werden, die Ergebnisse der Evaluation dann eingearbeitet und der gleiche Prototyp in einem späteren Stadium als digitaler Prototyp verwendet werden. Dieser könnte auch für Usability Tests ohne Anwesenheitn des Moderators verwendet werden, beispielsweise für Remote Tests.

Ebenso kann derselbe Prototyp mittels Screenshots oder als digitale Beilage zur Dokumentation von Spezifikationen verwendet werden. Die Lebensdauer und Verwendbarkeit übersteigt dadurch die des reinen handgezeichneten Prototypen.

5.3.3 Axure

Die grundlegende Überlegung bei einer Erstellung eines Prototypen mit Axure RP Pro ist, wie breit das Spektrum der abgedeckten Systemreaktionen auf Nutzerhandlungen sein soll. Ein schmaler Pfad würde ausschliesslich Nutzerhandlungen unterstützen, die auf dem Lösungspfad der geplanten Aufgaben liegen. Ein breiter Pfad würde auch Nutzerhandlungen unterstützen, die für die Lösung der Aufgaben nicht notwendig oder gar unplausibel sind. Eine gewisse Handlungsfreiheit für Testnutzer ist auch aus methodischer Sicht zu empfehlen: Ansonsten würde jede ausbleibende Systemreaktion auf eine Handlung zur Rückmeldung, dass die eben getätigte Handlung für die Lösung der Aufgabe nicht notwendig ist. Zudem ist oftmals die Fehlertoleranz eines Systems daran abzulesen, wie leicht es Nutzern gemacht wird, unerwünschte Systemzustände wieder zu verlassen. Das Anwendungsspektrum von Axure kann somit erweitert werden, indem man bewusst Systemreaktionen ausserhalb des Lösungspfades hinterlegt.

Diese Strategie bietet sich allerdings nur für Simulationen von RIA Patterns an, die von Axure RP Pro gut unterstützt werden. Für solche, bei denen dies nicht der Fall ist, muss ein anderes Vorgehen gewählt werden.

Auf der Homepage von Axure (2009) sind zahlreiche Bildschirmelemente verfügbar, die als Erweiterungen kostenlos für bestehende Axure RP Pro Installationen herunter geladen werden können. Manche der im Rahmen dieser Arbeit vermissten RIA Patterns (wie z.B. der Schieberegler) sind dort aufgelistet. Jedoch simulieren diese Bildschirmelemente rein das Aussehen der entsprechenden RIA Patterns, und nicht deren Verhalten. Für Tests, in denen Teilnehmer Aktionen mit diesen RIA Patterns ausführen, sind diese direkt von Axure angebotenen Hilfen somit nahezu wertlos.

Engagierte Nutzer von Axure RP Pro haben hingegen im Internet ganze Bibliotheken von interaktiven Bildschirmelementen zur freien Verfügung hinterlegt (Baxter, 2008). Diese sind lebendiger Ausdruck vom Gestaltungswillen mit begrenzten Mitteln, und bieten mitunter noch eindrückliche Beispiele, wie mit viel Kreativität und Fleiss behelfsmässige Lösungen entwickelt werden können. Doch selbst unter diesen günstigen Voraussetzungen stossen Vorreiter wie Baxter an technische Grenzen. Zum Einsatz seines Prototyps des RIA Patterns „Drag und Drop“ schreibt er: „The “Click and Drag” widgets are still a bit rough around the edges, so I’ve marked them ‚beta’ and recommend against using them for user testing.“ (Abschnitt „What’s new in this version?“).

Fortgeschrittene und experimentierfreudige Benutzer können auch auf „Hybride“ setzen, die den von Axure RP Pro generierten HTML Code mit Inhalten von anderen Quellen (z.B. mit Ajax- oder Flash-Komponenten) anreichern und so bestimmte RIA Pattern Simulationen ermöglichen (Wimmer, 2007). Axure RP Pro bietet zu diesem Zweck ein spezielles Integrations-Widget an.

Eine vollwertige Simulation von allen RIA Patterns mit Axure RP Pro wird allerdings wohl erst in Zukunft möglich werden. Dies, sobald der Softwareanbieter von Axure das Programm so weiterentwickelt, dass auch RIA Patterns standardmässig unterstützt werden, die auf dynamische nutzergesteuerte Bewegungen von Bildschirmelementen beruhen. Bis dahin müssen Nutzer von Axure RP Pro mit den programminhärenten Einschränken leben und sich mit Behelfslösungen trösten.

5.3.4 Ausprogrammierter Prototyp

Mit einem ausprogrammierten Prototyp können Aussehen und Verhalten der finalen Anwendung so realitätsnah demonstriert werden wie mit keiner anderen Prototyping Methode. Der Preis für diese Genauigkeit ist aber ein signifikant höherer Lern- und Entwicklungsaufwand. Es stellt sich daher weniger die Frage, wie die Methode in Bezug auf die Ähnlichkeit zum fertigen System verbessert werden kann, sondern wie der Aufwand und die Komplexität für die Erstellung des Prototyps optimiert werden kann.

Wie bereits erwähnt, wird der Prototyp idealerweise mit dem gleichen RIA-Framework wie das finale System implementiert. Dieses Ziel steht aber möglicherweise im Widerspruch mit dem Bestreben den Prototyp auf eine möglichst effiziente Art und Weise zu erstellen. Es kann darum gute Gründe geben, wieso die Wahl der technischen Infrastruktur von der Ziellösung abweicht:

- Die notwendigen Kenntnisse der Ziel-Architektur sind nicht vorhanden.
- Für die Umsetzung des Prototyps müssen zuerst grundlegende Voraussetzungen erarbeitet werden (z.B. Entwicklung von fehlenden Widgets).
- Die technische Infrastruktur ist zum Zeitpunkt der Prototyping Phase noch nicht bekannt.
- Die Ziel-Architektur erlaubt kein leichtgewichtiges Prototyping.
- Es soll bewusst ein Wegwerf-Prototyp implementiert werden.

Damit die Entwicklung der ausprogrammierten Prototypen einfacher und schneller vollzogen werden kann, sollten darum bei der Wahl der technischen Infrastruktur einige Punkte berücksichtigt werden.

Technologische Ausgereiftheit

Der Entwickler hat die Wahl zwischen mehreren RIA Plattformen, verschiedenen Programmiersprachen und unzähligen Frameworks. Der Markt für RIA Entwicklungs-Software ist hart umkämpft und fast täglich erscheinen neue Frameworks. Viele der Produkte sind aber technologisch noch wenig ausgereift, unvollständig oder schlecht dokumentiert. Entscheidend ist beispielsweise, ob die ausgewählten Komponenten alle Basis-Funktionen bieten, die man für den Prototyp benötigt. Fehlende Widgets, die zuerst selbst umgesetzt werden müssen, vergrössern den Entwicklungsaufwand unnötig. Eine grosse Entwickler-Gemeinschaft kann ebenfalls hilfreich sein, damit man auf Diskussionsgruppen oder Code-Beispiele Zugriff hat.

Programmiermodell

Für die Umsetzung der Dynamik und Interaktivität von RIA Patterns muss die Logik in einer bestimmten Form spezifiziert werden. Das Spektrum reicht von rein deklarativen Ansätzen über Meta- und Skriptsprachen bis zu kompilierenden Programmiersprachen. Die Wahl des Programmiermodells kann sowohl die Lernkurve wie auch die notwendige Anzahl Zeilen Code beeinflussen. Deklarative- oder Meta-Sprachen erleichtern vor allem den Nicht-Technikern den Einstieg, fordern aber oftmals Kompromisse bei der Umsetzung. Die Anwendung von nativen Technologien wie JavaScript, Java, HTML und CSS in Kombination mit einem Ajax-Framework bietet in der Regel mehr Flexibilität setzt aber gleichzeitig grössere technische Kenntnisse voraus. Mögliche Alternativen sind Programmiersprachen wie Ruby on Rails oder Grails die speziell den agilen Entwicklungsprozess unterstützen und immer mehr an Popularität gewinnen.

Tool-Support

Im Rahmen der RIA- und Ajax-Euphorie sind in letzter Zeit viele Entwicklungstools entstanden, die entweder auf bestehenden Standards und Technologien aufsetzen oder proprietär sind. Die Auswahl der Werkzeuge reicht von einfachen Source Editoren und

Generatoren über Ajax-Plugins für bestehende Entwicklungsumgebungen wie Dreamweaver oder Eclipse bis zu dedizierten visuellen RIA-Entwicklungsumgebungen. WYSIWYG Tools bieten beispielsweise einen Page Designer, bei dem UI Komponenten (Widgets) mit Drag und Drop auf dem User Interface arrangiert und mit Verhalten verknüpft werden. Der Code kann jederzeit generiert und ausgeführt werden. Viele der Tools sind unter einer Open Source Lizenz verfügbar und verursachen so keine Kosten. Je nach Wahl der Tools kann der Lern- und Entwicklungsaufwand gegenüber der traditionellen Programmierung signifikant gesenkt und der Lebenszyklus des Prototyps besser unterstützt werden.

Weitere Kriterien

Es gibt weitere Aspekte, welche die Wahl der technischen Infrastruktur beeinflussen können, für die Entwicklung von Prototypen aber eher von geringerer Bedeutung sind. Dazu zählen beispielsweise Cross-Browser Support, Security, Accessiblity und Abhängigkeiten zu einer Server-Infrastruktur.

Laut einer aktuellen Umfrage unter 200 Interaktionsdesignern und anderen mit dem Prototyping von User Interfaces betrauten Personen (Warfel, 2008) benutzt die Mehrheit der Befragten mindestens eine Technik bzw. Werkzeug für das Erstellen von ausprogrammierten Prototypen:

Client-seitige Programmierung mit HTML, CSS, JavaScript etc.	58%
→ Davon Einsatz von Adobe Dreamweaver	47%
→ Einsatz eines anderen HTML Editors	4%
Einsatz Adobe Flash/Flex, Microsoft Expression Blend oder ähnliches Tool	27%
→ Davon Einsatz von Adobe Flash	21%
Server-seitige Programmierung mit Java, Ruby, .NET, PHP, Xcode etc.	9%

Es zeigt sich, dass also vor allem Dreamweaver und Flash beliebte Tools für die Erstellung von digitalen Prototypen sind. Während Flash schon länger Möglichkeiten für den Bau von Rich Client Applikationen bot, mauserte sich auch Dreamweaver von einem HTML Editor langsam zu einer Ajax Entwicklungsumgebung. Seit der Version CS3 ist in Dreamweaver die Ajax Library Spry enthalten, die von Adobe mit Elementen des script.aculo.us Frameworks entwickelt wurde. Spry-basierte Ajax-Widgets lassen sich ohne zu programmieren in Web Seiten einbinden und optional mit Daten aus XML-Quellen oder Datenbanken befüllen. Es können ebenfalls Effekte integriert werden, um Seitenelemente bei einer Aktion des Benutzers zu animieren, beispielsweise um ein Bild bei einem Klick aus- oder einzublenden. In der Version CS4 kamen weitere unterstützende Funktionen für Ajax-Entwickler hinzu. Neu besteht auch die Möglichkeit, HTML-basierte Adobe AIR Anwendungen mit Dreamweaver zu entwickeln. Daneben existierten eine Menge von Dreamweaver Ajax Extensions von Drittanbietern, z.B. für jQuery oder Prototype.

Adobe Flash ist wie auch sein Vorgänger Macromedia Director ein beliebtes Tools bei Designern für die Erstellung von interaktiven Anwendungen. Flash hat den Begriff Rich Internet Applikationen geprägt (Mullet, 2003) und ist heute zusammen mit dem Adobe Flex Framework und dem Adobe AIR Client ein starker Mitbewerber bei der Auswahl einer RIA Plattform. Duane Bray beschreibt einen Trend unter dem Begriff „Live Prototyping“ (Moggridge, 2006), bei welchem Methoden und Tools von Interaktionsdesignern mit den Entwicklungswerkzeugen von Softwareentwicklern konvergieren. Als Beispiel nennt er Flash und Flex, das sich sowohl für das rasche Prototyping wie auch für die finale Entwicklung von interaktiven Anwendungen mit Anbindungen an reale Datenquellen eignet. Auch die Kultur, dass Code-Beispiele in Entwickler-Communities kommentiert und

ausgetauscht werden, reduziert die Lernkurve der Tools und beschleunigt die Umsetzung der Anwendungen.

Neben Dreamweaver und Flash gibt es heute eine Menge von neuen Programmierwerkzeugen, die sich für das Prototyping von RIA eignen. Eine Auswahl von Tools folgt an dieser Stelle.

Protoscript

Protoscript ist eine Meta-Skriptsprache die speziell für die Erstellung von Ajax-Prototypen entwickelt wurde. Die Syntax entspricht dem JSON Format (JavaScript Object Notation) und kann auch von Nicht-Technikern schnell erlernt werden. Elemente des User Interfaces können auf einfache Weise mit Maus- oder Tastaturereignissen und einem bestimmten Verhalten verknüpft werden.

Protoscript integriert sich mit der Yahoo! User Interface Library (YUI) und kann für andere Ajax Frameworks erweitert werden. Es werden im Moment etwas mehr als 30 Ereignisse und Verhalten unterstützt, darunter Doppelklick, Drag und Drop oder Spotlight. Zusammen mit den Widgets von YUI bietet Protoscript grosses Potential, um anspruchsvolle RIA Prototypen einfach zu erstellen. Allerdings werden Grundkenntnisse von HTML und JavaScript vorausgesetzt.

Ruby on Rails/Grails

Ruby on Rails ist ein Open-Source Framework für Web-basierte Anwendungen auf der Basis der Programmiersprache Ruby und der Model-View-Controller Architektur. Ruby ist eine objektorientierte und plattformübergreifende Interpreter-Sprache, die im Jahre 1995 veröffentlicht wurde und die Vorteile von Skriptsprachen wie Perl, List, Python und Smalltalk verbindet. Ein besonderes Merkmal der Sprache ist die extrem gute Lesbarkeit des Programm-Codes. Die Grundprinzipien von Ruby on Rails sind „Don't repeat yourself" (Wiederhole dich nicht) und „Convention over configuration" (Konvention über Konfiguration). Don't repeat yourself besagt, dass Redundanz zu vermeiden ist und jede Information nur einem Platz vorhanden ist. Convention over configuration bedeutet, dass der Entwickler nur unkonventionelle Aspekte der Anwendung definieren muss, alles andere erfolgt über Defaultwerte. Ruby on Rails bietet einige Tools, um die Entwicklung von wiederkehrenden Entwicklungsarbeiten zu vereinfachen. Für die Entwicklung RIA Patterns werden die Ajax Frameworks „Prototype" und „script.aculo.us" integriert. Ruby on Rails wird oft in Verbindung mit einer agilen Entwicklungsmethode eingesetzt und verspricht gegenüber anderen Technologien eine deutlich kürzere Entwicklungszeit.

Das Framework „Grails" lehnt sich an Ruby on Rails an, setzt allerdings auf der Programmiersprache Groovy auf.

Appcelerator

Appcelerator ist eine Open Source Plattform für die Entwicklung von Rich Internet Anwendungen. Die Programmierung des User Interfaces erfolgt bei Appcelerator mit der Web Expression Language, einer Erweiterung von HTML. Es kann damit auf eine deklarative Art die Client-Logik implementiert werden. Die Message-Orientierte Architektur von Appcelerator ermöglicht, dass jedes HTML Element Meldungen senden und empfangen kann, um das User-Interface dynamisch zu verändern oder Daten von einem Service anzufordern. Mit dem Widget Framework können eigene User-Interface Komponenten entwickelt werden oder bestehende Widget Libraries integriert werden. Appcelerator bietet selbst bereits etwa 40 Widgets die ebenfalls mit einfachem Markup eingebunden werden können. Appcelerator unterstützt sogenannte „Interactive Use Cases", eine agile Methode für das Rapid Application Development von RIA Patterns. Damit kann in schnellen Iterationen das User-Interface verfeinert werden und am Ende in das finale System überführt werden. Ein weiteres interessantes Feature von Appcelerator ist

die Möglichkeit, auf jedem User Interface Element einen Kommentar eingeben zu können bzw. die Kommentare von anderen Benutzern zu sehen.

Mit dem bevorstehenden Appcelerator „Titanium“ Release sollen Anwendungen aus dem Web installiert und ohne einen Browser auf dem Desktop ausgeführt werden können. Appcelerator tritt damit in Konkurrenz mit Adobe AIR und Microsoft Silverlight.

WaveMaker Visual Ajax Studio

WaveMaker Visual Ajax Studio ist eine Open Source WYSIWYG Entwicklungsumgebung für die Erstellung von Ajax Anwendungen mit Spring, Hibernate und Dojo. Der Editor läuft in einem Web Browser und erlaubt die einfache Zusammenstellung der Screens über ein Drag und Drop Interface. WaveMaker unterstützt die Widgets des Dojo Frameworks. Datenbanken und Web Services können problemlos eingebunden und mit den Komponenten verknüpft werden. Die erstellte Anwendung kann auf einen Klick für Testzwecke generiert und ausgeführt werden.

Gemäss der Aussage des Herstellers wird mit WaveMaker die Entwicklungszeit von Anwendungen um 67% reduziert und die Anzahl der Zeilen Code um 98%. WaveMaker richtet sich an Entwickler, die es gewohnt sind mit visuellen Tools zu arbeiten. All jene, die gerne selbst Code editieren, werden eher enttäuscht sein.

Microsoft Expression Blend

Expression Blend ist ein kommerzielles Design-Werkzeug von Microsoft für die Gestaltung von interaktiven graphischen Desktop- und Web-Benutzeroberflächen. Mit dem WYSIWYG Editor können XAML-basierte (Extensible Application Markup Language) Interfaces für die Windows Presentation Foundation (WPF) und Silverlight erstellt werden. Mehr als 30 Controls und Container können über ein Drag und Drop Interface verwendet werden. Handgezeichnete oder mit PowerPoint erstellte Wireframes können importiert werden. Eine Zeitachse ermöglicht die Animation von Objekten. Über Data-Binding können Controls mit Web-Services oder .NET Objekten verknüpft werden, ohne dass dafür Code geschrieben werden muss. Für erweiterte Logik kann in C# oder Visual Basic programmiert werden. Für die Ausführung von Web-Inhalten, die mit Expression Blend erstellt wurden, muss im Browser das Silverlight Plugin installiert sein.

Obwohl Microsoft Expression Blend ein Produktionstool ist, wird es von vielen Interaktionsdesignern auch für das Prototyping von RIA Patterns benutzt. Positiv ist, dass das Tool ein evolutionäres Prototyping vom statischen Wireframe bis zum fertigen Produkt unterstützt. Die für das Prototyping von User-Interfaces notendige Funktionalität kann in relativ kurzer Zeit erlernt werden. Da Expression Blend aber vorwiegend auf Microsoft-proprietäre Technologien setzt, ist ein Einsatz in einem nicht-Microsoft Umfeld eher fragwürdig.

Adobe Flash Catalyst

Flash Catalyst, lange Zeit bekannt unter dem Codenamen „Thermo“, ist ein neues Interaktions-Design Tool von Adobe für die schnelle und einfache Erstellung von interaktiven Inhalten und Rich Internet Applikationen. Das Tool richtet sich vor allem an Designer, welche Vorlagen, die mit Anwendungen aus der Adobe Creative Suite 4 (Photoshop, Illustrator oder Fireworks) erstellt wurden, in interaktive Flash oder AIR Anwendungen konvertieren können, ohne programmieren zu müssen. Es wird ein Roundtrip Workflow unterstützt, d.h. es kann jederzeit zwischen den Design Programmen und Flash Catalyst hin und her gewechselt werden, ohne dass dabei Arbeit verloren geht. Projektdateien können ebenfalls an Software-Entwickler für die Umsetzung von komplexerer Funktionalität (z.B. Anbindung an Backend-Systeme) mit dem Adobe Flex Builder weitergegeben werden. Das Tool unterstützt das Prototyping dahingehend, dass ein oder mehrere Design-Entwürfe über mehrere Iterationen hinweg von einfachen Wireframes bis zu voll funktionalen Anwendungen verfeinert werden können.

6 Fazit

Alle vier untersuchten Prototyping-Methoden[2] sind in der Lage, Prototypen von RIA Patterns zu simulieren. Jedoch hat jede Methode ihre spezifischen Schwächen:

- Per Hand und PowerPoint erstellte Papier-Prototypen reagieren spürbar langsamer auf Benutzerhandlungen als ein elektronischer Prototyp. Weiterhin sind Änderungen auf dem Bildschirm durch die notwendigen manuellen Manipulationen für die Testteilnehmer praktisch nicht zu übersehen. Damit wird die Erkennbarkeit von visuellem Feedback und systembedingten Bildschirmaktualisierungen deutlich überschätzt. Schliesslich animierten die Papier-Prototypen wesentlich weniger als die elektronischen, die Benutzerschnittstelle durch Ausprobieren frei zu explorieren.
- Axure RP Pro ist nicht in der Lage, dynamische, nutzergesteuerte Bewegungen von Bildschirmelementen zu simulieren. Damit sind RIA Patterns wie „Schieberegler", „Drag und Drop" sowie das „Verschieben von Kartenausschnitten" nur unbefriedigend zu simulieren. Zudem erkennt Axure RP Pro bei dem RIA Pattern „Kontextmenü" keinen Rechtsklick auf der Maus, so dass entsprechend angewählte Kontextinformationen nicht realitätsgetreu aufgerufen werden können.
- Die ausprogrammierten Ajax Prototypen waren am wenigsten von Einschränkungen bei der Simulation von RIA Patterns betroffen. Einzig eine zu langsame Reaktionszeit beim Pattern „Eingabevorschläge" beeinträchtigte die Darstellung in der ersten Iteration. Dieser Nachteil konnte durch eine optimierte technische Implementierung allerdings in der zweiten Iteration vollständig behoben werden. Ein weiteres Problem war die zu schnelle Reaktionszeit beim Pattern „Sofortige Filterung von Suchergebnissen". Die Systemreaktionen erfolgte so schnell, dass Testteilnehmern oftmals die aktualisierte Bildschirmanzeige entging, und sie vergeblich auf die vermeintlich noch folgende Systemreaktion warteten.

Das gewählte Versuchsdesign hat sich bewährt: Es war sehr aufschlussreich zu beobachten, wie Testteilnehmer mit den verschiedenen Prototyp-Versionen Aufgaben bearbeiteten. Viele Erkenntnisse wären uns ohne diesen empirischen Zugang verschlossen geblieben.

Trotz Gegenmassnahmen erscheint es uns allerdings während der Beobachtung als unvermeidlich, dass sich Testteilnehmer sehr oft auf die Inhalte der Prototypen beziehen, statt auf deren Form. Diese Abstraktionsleistung an die Testteilnehmer zu delegieren hat sich als Fehlschlag erwiesen. Somit müssen Schwierigkeiten, die sich auf inhärente Eigenschaften der Prototypen-Versionen beziehen, aus den zahlreichen Äusserungen und Beobachtungen der Testteilnehmer herausgefiltert werden. Um diese Filterung zu vereinfachen, wäre es vorteilhaft gewesen, mehr Aufwand in die Erarbeitung bzw. das vorgängige Testen der Testaufgaben zu stecken. Probleme, die aufgrund der Aufgabenformulierung bzw. des Aufgabendesigns entstanden, hätten dadurch sicher minimiert werden können. (vgl. 4.2)

Die inferenzstatistische Auswertung der quantitativen Messwerte ergab, dass sich die vier Prototyp-Versionen vergleichbar gut eignen, RIA Patterns zu simulieren. Dass die beobachteten Unterschiede zwischen den Zielvariablen keine statistische Signifikanz erreichten hatte drei Ursachen:

---

[2] a) per Hand erstellter Papier-Prototyp, b) mit PowerPoint erstellter Papier-Prototyp, c) mit Axure erstellter digitaler Prototyp und d) mit Ajax kodierter, programmierter Prototyp

a) Der Mittelwertsunterschied zwischen den gemessenen Zielvariablen war nur gering

b) Die Messwerte der Zielvariablen streuten stark

c) Der Stichprobenumfang war mit 10 bzw. 15 Testteilnehmern eher gering

Ein grösserer Stichprobenumfang hätte die Teststärke (siehe Glossar) erhöht und manche Mittelwertsunterschiede effektiver gegen den Zufall abgrenzen können. Bei quantitativen Auswertungen stellt sich jedoch nicht nur die Frage, ob die beobachteten Unterschiede auf systematische Effekte zurückzuführen (d.h. signifikant) sind, sondern auch, ob das Mass der Unterschiede praktisch relevant ist. Unsere Untersuchung gibt dieser Annahme wenig Nahrung - die vier ausgewählten Prototypen-Versionen sind demnach aus unserer Sicht für die Praxis allesamt gut geeignet, RIA Patterns zu simulieren. Statt pauschaler Bewertungen einer gesamten Prototyp-Methode sollten vielmehr RIA Pattern-spezifische Einschränkungen der einzelnen Prototyp-Versionen hervorgehoben werden, wie wir es eingangs getan haben.

Welche Empfehlungen lassen sich aus unseren Erkenntnissen für die Praxis der Gestaltung von Benutzerschnittstellen ableiten?

Papier-Prototypen spielen ihre Stärke bei der schnellen Erstellung von groben Entwürfen aus, um Lösungsvarianten miteinander zu vergleichen. Auch die Exploration von neuen Varianten noch während der Evaluation ist damit möglich Bei der iterativen Optimierung eines einzelnen Prototyps hingegen ist vor allem der per Hand erstellte Papier-Prototyp zu zeitaufwändig, da viele Papierelemente angepasst und für einen Testeinsatz zurechtgeschnitten werden müssen. Letzteres betrifft auch elektronisch erstellte Papier-Prototypen, auch wenn hier von elektronischen Vervielfältigungsmöglichkeiten profitiert werden kann. Bei iterativen Optimierungen haben elektronische Prototypen einen klaren Geschwindigkeitsvorteil durch die Anpassung von wiederkehrenden Vorlagen („Masters“) und durch die Möglichkeit, Elemente durch mehrfaches „Copy & Paste“ zu vielfältigen. Für eine nächste Untersuchung wäre es deshalb sicher interessant, PowerPoint sowohl für die Erstellung von Papier- wie auch elektronischen Prototypen zu verwenden. Mit demselben Werkzeug könnten dann Vorteile von zwei verschiedenen Prototypen-Arten genutzt werden.

Axure RP Pro erlaubt, einen digitalen Prototyp mit einer gewissen Interaktivität ohne jegliche Programmierkenntnisse zu erstellen. Der Bibliothek an Bildschirmelementen können neue hinzugefügt werden – damit können früher erstellte Interaktions- oder Anzeigebausteine leicht wieder verwendet werden. Diesem leichten Zugang zur Erstellung interaktiver HTML-Prototypen steht allerdings wie beschrieben eine mangelnde Fähigkeit gegenüber, RIA Patterns zu simulieren, die auf dynamisches Verschieben von Bildschirmelementen angewiesen sind. Dies ist eine erhebliche Einschränkung, da diese RIA Patterns bereits heute sehr zahlreich sind und in Zukunft mit Sicherheit noch weitere hinzukommen werden. Die Anbieter von Axure RP Pro müssen die Software um diese Fähigkeit in Kürze anreichern, wenn sie auch weiterhin als Prototyping Werkzeug erfolgreich sein möchte. Die Erfahrungen haben gezeigt, dass man mit Umgehungslösungen die nicht dem mentalen Modell des Benutzers entsprechen, mehr Schaden anrichtet als Probleme löst. Wenn Patterns wie Drag und Drop simuliert werden sollen, bleibt nur der Weg, diese mit einer echten RIA Technologie zu erstellen und in den Axure RP Pro Prototypen zu integrieren oder auf einen anderen, beispielsweise einen Papier-Prototypen, auszuweichen.

Ein ausprogrammierter Prototyp gleicht zweifelsohne am stärksten dem späteren produktiven System. Beim evolutionären Prototyping werden sogar Teile des Programmcodes in das finale System übernommen. Beschränkungen geht ein kodierter Prototyp eher aus Aufwands-/Nutzenüberlegungen ein, als aus technischen Gründen. Der Aufwand, einen kodierten Prototyp zu erstellen, hängt noch enger an den vorhandenen

Kenntnissen der Beteiligten. Insbesondere dürfte die Lernkurve wesentlich steiler ausfallen, sich fehlende Programmierfähigkeiten ad hoc für ein Projekt anzueignen. Somit ist eine Programmierung mit Ajax oder anderen RIA Technologien das mächtigste Prototyping Werkzeug, hat aber durch die erforderten Kenntnisse eine sehr hohe Einstiegshürde und ist je nach simuliertem RIA Pattern auch aufwändiger als die anderen besprochenen Prototyping-Methoden. Erfreulicherweise kommen immer mehr Tools auf den Markt, die versuchen die Komplexität bei der Erstellung von RIA zu reduzieren und auch Nicht-Technikern den einfachen Einstieg zu ermöglichen.

Gleich, welche dieser vier besprochenen Prototyping Methoden verwendet wird – alle erlauben in einem unterschiedlich hohen Masse, RIA Patterns in Prototypen zu simulieren. Eine auf den Einsatzzweck abgestimmte Auswahl, oder gar Kombination, von Prototyping Methoden ebnet den Weg, auch im Zeitalter von Web 2.0 interaktive Angebote durch Prototypen zu testen und für den Einsatzalltag zu optimieren.

7 Glossar

Abhängige Variable: Messgrösse eines wissenschaftlichen Experimentes. Veränderungen der abhängigen Variablen werden systematisch beobachtet, um Auswirkungen der gezielten experimentellen Manipulation der → unabhängigen Variablen zu messen und so Wirkungszusammenhänge aufzudecken.

Adobe AIR: (Adobe Integrated Runtime). Dabei handelt es sich um eine plattformübergreifende Laufzeitumgebung für die Entwicklung von → Rich Internet Applikationen, die auch ausserhalb des Browsers ausgeführt werden können.

Affordance: Siehe → Aufforderungscharakter

Aufforderungscharakter: Beschreibt, wie gut die sensorischen Eigenschaften eines Interaktionselements auf seine Funktionalität und Benutzung hinweisen. Ein Schalter soll z.B. je nach Aussehen gedrückt, gekippt oder verschoben werden.

Ajax: Ajax steht für **A**synchronous **J**avascript **a**nd **X**ML. Darunter ist die Anwendung von verschiedenen Technologien wie → JavaScript, → HTML, → Document Object Model (DOM) und der → XMLHttpRequest zu verstehen, mit welchen Desktop-ähnliche Web-Anwendungen realisiert werden können.

Blog: Kurzform für Weblog. Ein von einem "Blogger" geführtes öffentliches Web-Tagebuch oder Journal zu einem bestimmten Thema.

Document Object Model (DOM): Programmierschnittstelle, welche die Manipulation von → HTML und → XML Dokumenten ermöglicht, indem die Elemente des Dokuments als Datenstrukturen zugänglich gemacht werden.

Eclipse: Beliebte, → Open Source Software-Entwicklungsumgebung.

Fidelity: Beschreibt wie detailgetreu ein Prototyp in Bezug auf Aussehen und Verhalten gegenüber dem finalen System ist.

Folksonomy: Zusammensetzung der Worte *folk* "Volk" und *taxonomy* "Taxonomie". Eine durch → Tagging erstellte Sammlung von Tags.

Friedman Test: Statistisches Testverfahren zum Vergleich von drei oder mehr gepaarten Stichproben. Der Friedman Test macht keine Annahmen über die Wahrscheinlichkeitsverteilung der Stichproben. Im Gegensatz zum → Kruskal-Wallis-Test beruht er darauf, dass die gleichen Testteilnehmer mehrere Objekte bewerten, die untereinander verglichen werden sollen.

Heuristik: In der Psychologie sind Heuristiken einfache, effiziente Regeln, die sich durch evolutionäre Prozesse gefestigt haben oder erlernt wurden. Sie werden insbesondere genutzt, um die Lagebeurteilungen, Entscheidungsfindungen und Problemlösungen von Menschen in komplexen Situationen, in denen es häufig an Informationen mangelt, zu erklären (Wikipedia, 2009).

HTML: Abkürzung für **H**yper**t**ext **M**arkup **L**anguage. Standardisierte Seitenbeschreibungssprache für Dokumente im World Wide Web.

HTTP: **H**yper**t**ext **T**ransfer **P**rotocol. Ein Protokoll zur Übertragung von Daten (z.B. eine → HTML Seite) über das Internet.

Inferenzstatistik: Teilgebiet der Statistik, die Aussagen über Wahrscheinlichkeiten von bestimmten Zufallsverteilungen macht. Sie analysiert Daten mit Hilfe von mathematischen Modellen.

JavaScript: Von Netscape entwickelte objektbasierte Scriptsprache die überwiegend für die clientseitige

Programmierung von Web Seiten verwendet wird.

JSON: JSON – Abkürzung für **J**ava**S**cript **O**bject **N**otation. Kompaktes und einfach lesbares Format für den Austausch von Daten zwischen Anwendungen welches oft anstelle von XML bei → Ajax-Requests verwendet wird.

Kolmogorov-Smirnov Anpassungstest: Statistischer Test zur Prüfung, ob eine Stichprobe einer zuvor angenommenen Wahrscheinlichkeitsverteilung (z.B. einer Normalverteilung) folgt.

Kruskal-Wallis-Test: Statistisches Testverfahren zum Vergleich von drei oder mehr ungepaarten Stichproben. Wie der → Friedman Test macht der Kruskal-Wallis-Test keine Annahmen über die Wahrscheinlichkeitsverteilung der Stichproben. Im Gegensatz zum Friedman Test beruht er jedoch darauf, dass die gleichen Testteilnehmer jeweils nur eines der Objekte bewerten, die untereinander verglichen werden sollen.

Laufzeitumgebung: (von englisch: „runtime environment“) Softwareschicht, die von Programmen benötigte Basis-Funktionen zur Verfügung stellt. Lässt Anwendungsprogramme plattformunabhängig ablaufen, dh. unabhängig von der verwendeten Hardware und dem Betriebssystem. (Wikipedia, 2009).

Levene-Test: Statistischer Test, der prüft, ob die → Varianzen zweier Grppen homogen sind.

Mashup: Konzept bei dem Anwendungen und Inhalte aus verschiedenen Quellen zu neuen Angeboten kombiniert werden.

Model-View-Controller (MVC): Architekturmuster in der Software-Entwicklung zur Strukturierung des Programmcodes in die drei Einheiten Datenmodell (Model), Präsentation (View) und Programmsteuerung (Controller). Auf diese Weise wird die Wiederverwendbarkeit von Komponenten erhöht und Änderungen am User-Interface oder der Business-Logik erleichtert.

Open Source Software: Software, welche im Quellcode vorliegt und ohne Lizenzgebühren eingesetzt oder weiterentwickelt werden kann.

Papier-Prototyp: Prototyp, der aus gezeichneten oder auf Papier ausgedruckten Darstellungen von Bildschirminhalten besteht. Üblicherweise nur in einer groben →Wireframe Version.

PIA: **P**aged **I**nternet **A**pplications. Beschreibung von traditionellen Web-Anwendungen, bei welchem nach einer Interaktion des Benutzers eine komplette Seite an den Browser zurückgeschickt wurde.

Prädiktorvariable: Siehe → unabhängige Variable.

Problem Domain: Das Modell des Ausschnitts aus der Realwelt, dessen Probleme eine Software lösen soll. Definiert Konzepte und ihre Beziehungen. z.B. Kunde und Einkaufswagen in einem Online-Shop.

Rational Unified Process (RUP): Ein bei IBM entwickeltes Vorgehensmodell für die Durchführung von Softwareprojekten.

Reliabilität: Gütekriterium einer empirischen Untersuchung. Ergeben sich bei einer Wiederholung annähernd dieselben Ergebnisse? Siehe auch → Validität

REST: Abkürzung für **R**epresentational **S**tate **T**ransfer. Beschreibung von zustandslosen Web Services auf der Basis von HTTP.

RIA: Akronym für **R**ich **I**nternet **A**pplication. Der Begriff Rich Internet Application bezeichnet eine Klasse von Anwendungen, die die Eigenschaften von Desktop-Anwendungen besitzen, aber in einem Web-Browser ausgeführt werden und mit Internet Technologien realisiert wurden.

Runtime: Siehe → Laufzeitumgebung

SaaS: Abkürzung für **S**oftware **a**s **a** **S**ervice. Dienstleistung, bei der eine Software über das Internet angeboten und genutzt wird.

SOA: Abkürzung für **S**ervice **O**riented **A**rchitecture. Softwarearchitektur-Konzept, bei welchem Funktionen in Form von wiederverwendbaren und technisch unabhängigen Diensten (Services) implementiert werden.

SOAP: Akronym für **S**imple **O**bject **A**ccess **P**rotocol. → XML-basiertes Protokoll für den Austausch von Informationen zwischen Internetanwendungen. SOAP bildet die Basis für → Web Services.

Tagging: Assozierung von Web-Inhalten mit Schlagwörtern (Tags).

Teststärke: Wahrscheinlichkeit, mit der ein statistischer Test einen tatsächlich vorhandenen Unterschied bzw. Zusammenhang auch als statistisch signifikant erkennt.

Unabhängige Variable: Stellgrösse eines wissenschaftlichen Experimentes, die vom Versuchsleiter gezielt verändert wird. Die Auswirkungen der Manipulation der unabhängigen Variablen auf die Messgrösse (die → abhängige Variable) werden systematisch erhoben, um Wirkungszusammenhänge aufzudecken.

Validität: Gütekriterium einer empirischen Untersuchung. Wird gemessen, was gemessen werden soll? Siehe auch → Reliabilität

Varianz: „Mass, das beschreibt, wie stark eine [...] Zufallsgröße ‚streut'. Sie wird berechnet, indem man die Abstände der Messwerte vom Mittelwert quadriert, addiert und durch die Anzahl der Messwerte teilt" (Wikipedia, 2009).

Web 2.0: Bezeichnung einer zweiten Generation von Web-basierten Diensten und Communities wie soziale Netzwerke, Wikis, Blogs und Tauschbörsen, welche die Kreativität der Benutzer und den Austausch untereinander fördern.

Web Service: Verteilte Softwareanwendung im Internet, die über eine definierte → XML-basierte Schnittstelle aufgerufen werden kann.

Widget: Bezeichnet ein einzelnes Steuerelement, das in eine Anwendung integriert werden kann. Beispiele von Widgets sind ein Calendar Picker oder ein Tree View.

Wiki: Sammlung von Webseiten, die von den Benutzern gelesen und direkt und unkompliziert verändert werden können.

Wireframe: Visuelle Präsentation, welche Bildschirminhalte auf Linien, Flächen und generische oder grobe Darstellungen reduziert. Es geht hierbei primär um Struktur und Layout, also die Anordnung von Bildschirmelementen und deren Abfolge, nicht deren Aussehen.

XAML – Akronym für E**x**tensible **A**pplication **M**arkup **L**anguage. Deklarative, XML-basierte Markup-Sprache von Microsoft für die Beschreibung von graphischen Benutzeroberflächen für das .NET Framework.

XML: Akronym für E**x**tensible **M**arkup **L**anguage. Standardisierte Auszeichnungssprache für die strukturierte Beschreibung von Dokumenten. XML wird vor allem für den Austausch von Daten zwischen Computersystemen im Internet eingesetzt.

XmlHttpRequest: Ein Objekt im Web Browser, welches es ermöglicht mit → JavaScript asynchrone Anfragen an den Web Server zu stellen, um Daten auch ohne komplettes Neuladen der Seite auszutauschen.

Zielvariable: Siehe →abhängige Variable.

8 Literaturverzeichnis

Ajaxpatterns.org. Online verfügbar unter http://ajaxpatterns.org/, zuletzt aufgerufen am 29. Januar 2009.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language. Towns, Buildings, Construction*. New York: Oxford University Press.

Arnowitz, J., Arent, M., & Berger, N. (2007). *Effective prototyping for software makers*. San Francisco, CA: Morgan Kaufmann.

Axure Homepage. (2009). Online verfügbar unter http://www.axure.com/, zuletzt aufgerufen am 29. Januar 2009.

Baxter, L. (2008). *A clean design. Axure Design Pattern Library v2.0.* Online verfügbar unter http://www.acleandesign.com/2008/09/axure-design-pattern-library-v2/, zuletzt aufgerufen am 29. Januar 2009.

Beaudouin-Lafon, M., & Mackay, W. (2003). Prototyping Tools and Techniques. In *The Human-Computer Interaction Handbook. Fundamentals, Evolving Technologies and Emerging Applications* Second Edition (S. 1017–1039). New York: Lawrence Erlbaum Associates.

Beck, K. & Cunningham, W. (1987). *Using Pattern Languages for Object-Oriented Programs.* Online verfügbar unter http://c2.com/doc/oopsla87.html, zuletzt aufgerufen am 29. Januar 2009.

Bevan, N., Barnum, C., Cockton, G., Nielsen, J., Spool, J., & Wixon, D. (2003). The "magic number 5": is it enough for web testing. In ACM (Hrsg.), *CHI '03 extended abstracts on Human factors in computing system* (S. 698–699). New York: ACM (Conference on Human Factors in Computing Systems).

Cecil, R. F. (2007). *Documenting the Design of Rich Internet Applications: A Visual Language for State.* Online verfügbar unter http://www.uxmatters.com/MT/archives/000251.php, zuletzt aufgerufen am 29. Januar 2009.

Cooper, A. (2007). *About Face 3. The Essentials of Interaction Design.* New York: Wiley Publishing, Inc.

DIN EN ISO 9241 (1997). *Ergonomische Anforderungen für Bürotätigkeiten an Bildschirmgeräten. (ISO 9241:1997).* Berlin: Beuth Verlag.

DIN EN ISO 9241-110 (2006). *Grundsätze der Dialoggestaltung. (ISO 9241-110:2006).* Berlin: Beuth Verlag.

DIN EN ISO 13407 (2000). *Benutzer-orientierte Gestaltung interaktiver Systeme (ISO 13407:1999).* Berlin: ISO.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns. Elements of Reusable Object-Oriented Software*. Reading, MA: Addison Wesley.

Garrett, J. J. (2005). *Ajax: A New Approach to Web Applications.* Online verfügbar unter http://www.adaptivepath.com/ideas/essays/archives/000385.php, zuletzt aufgerufen am 29. Januar 2009.

Hammond, J. & Goulde, M. (2007). *Rich Internet Apps Move Beyond The Browser.* Cambridge, MA: Forrester Research.

Hein, G. (2005). *Rich Internet Applications: Eye Candy or Real Enterprise Value.* Midvale, UT: Burton Group.

Heinsen, S. & Vogt, P. (Hrsg.) (2003). *Usability praktisch umsetzen. Handbuch für Software, Web, Mobile Devices und andere interaktive Produkte.* [Website mit Linklisten und nützlichen Ergänzungen]. München: Hanser.

Floyd, I. R., Jones, C. M., Rathi, D., & Twidale, M. B. (2007). Web Mash-ups and Patchwork prototyping: User-driven technological innovation with Web 2.0 and Open Source Software. In HICSS '07: *Proceedings of the 40th Annual Hawaii International Conference on System Sciences* (S. 86), Washington, DC, USA. IEEE Computer Society.

Kruskal-Wallis Test. Online verfügbar unter visor.unibe.ch/WS02/statistik2/kruskal.pdf, zuletzt aufgerufen am 29. Januar 2009.

Liu, L. & Khooshabeh, P. (2003). Paper or Interactive? A Study of Prototyping Techniques for Ubiquitous Computing Environments. In ACM (Hrsg.), *CHI '03 extended abstracts on Human factors in computing systems* (S. 1030–1031). New York: ACM (Conference on Human Factors in Computing Systems).

Mahemoff, M. (2006). *Ajax Design Patterns.* (1. Auflage). Sebastopol, CA: O'Reilly Media.

Mahemoff, M. (2009). *AjaxPatterns.* Online verfügbar unter http://ajaxpatterns.org/, zuletzt aufgerufen am 29. Januar 2009.

Markoff, J. (2006). *Entrepreneurs See a Web Guided by Common Sense.* Herausgegeben von The New York Times. Online verfügbar unter http://www.nytimes.com/2006/11/12/business/12web.html, zuletzt aufgerufen am 29. Januar 2009.

Mäuselein, M. (2007). *Paper Prototypes vs. Computer-based Prototypes in a User-centered Design Process. Final Thesis.* Paderborn: Universität von Paderborn.

Moggridge, B. (2006). *Designing Interactions.* Cambridge, MA: MIT Press.

Molich, R., Damgaard T. A., Karyukina, B., Schmidt, L., Ede, M., van Oel, W., & Arcuri, M. (1999). Comparative evaluation of usability tests. In ACM (Hrsg.), *CHI '99 extended abstracts on Human factors in computing systems* (S. 83-84). New York: ACM (Conference on Human Factors in Computing Systems).

Monson-Haefel, R. (2007). *Rich Internet Applications: Creating an Effective Web Experience.* Midvale, UT: Burton Group.

Mullet, K. (2003). *The Essence of Effective Rich Internet Applications.* Herausgegeben von Macromedia. Online verfügbar unter http://download.macromedia.com/pub/solutions/downloads/business/essence_of_ria.pdf , zuletzt aufgerufen am 29. Januar 2009.

Nielsen, J. (1993). *Usability Engineering.* San Francisco, CA: Morgan Kaufmann.

Nielsen, J. (2005). *Why Ajax Sucks (Most of the Time).* Alertbox, December 2005. Online verfügbar unter http://www.usabilityviews.com/ajaxsucks.html, zuletzt aufgerufen am 29. Januar 2009.

Nielsen, J. & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In ACM (Hrsg.), *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems* (S. 206–213). New York: ACM (Conference on Human Factors in Computing Systems).

Nielsen, J. & Mack, R. L. (1994). *Usability Inspection Methods.* New York, NY: John wiley & Sons, Inc.

Olsen, H. (2002). *Results from a survey of web prototyping tools usage.* Online verfügbar unter http://www.guuui.com/issues/01_03_02.php, zuletzt aufgerufen am 29. Januar 2009.

O'Reilly, T. (2005). *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.* Online verfügbar unter http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, zuletzt aufgerufen am 29. Januar 2009.

Park, T., Lee, J. H., Kim, S., Kim, M., Shim, M.; Park, J. J.-K., & Noh, H. (2007). *Design Process Improvement for Effective Rich Interaction Design. The Hong Kong Polytechnic University School of Design.* Online verfügbar unter

http://www.sd.polyu.edu.hk/iasdr/proceeding/papers/Design%20Process%20Improvement%20for%20Effective%20Rich%20Interaction%20Design.pdf, zuletzt aufgerufen am 29. Januar 2009.

Rogowski, R. (2006). *Smackdown: Rich Internet Applications Versus HTML.* Cambridge, MA: Forrester Research.

Rudd, J., Stern, K., & Isensee, S. (1996). Low vs. high-fidelity prototyping debate. Interactions, 3 (1), S. 76–85.

Sarodnick, F. & Brau, H. (Hrsg.) (2006). *Methoden der Usability Evaluation. Wissenschaftliche Grundlagen und praktische Anwendung.* (1. Auflage). Bern: Huber.

Scott, B. (2005). *Storyboarding Rich Internet Applications with Visio.* Online verfügbar unter http://www.boxesandarrows.com/view/storyboarding_rich_internet_applications_with_visio, zuletzt aufgerufen am 29. Januar 2009.

Scott, Bill (2009): *Designing Web Interfaces. Principles and Patterns for Rich Interaction.* Online verfügbar unter http://designingwebinterfaces.com/, zuletzt aufgerufen am 29. Januar 2009.

Sefelin, R., Tscheligi, M., & Giller, V. (2003). Paper prototyping - what is it good for? A comparison of paper- and computer-based low-fidelity prototyping. In ACM (Hrsg.), *CHI '03 extended abstracts on Human factors in computing systems* (S. 778–779). New York: ACM (Conference on Human Factors in Computing Systems).

Snyder, C. (2003). *Paper Prototyping. The Fast and Easy Way to Design and Refine User Interfaces.* San Francisco, CA: Morgan Kaufmann.

Spool, J. & Schroeder, W. (2001). Testing web sites: five users is nowhere near enough. In ACM (Hrsg.), *CHI '01 extended abstracts on Human factors in computing systems* (S. 285–286). New York: ACM (Conference on Human Factors in Computing Systems).

Tibbetts, J. (2008). RIA Patterns: UIs, Plattforms, and Architecture. *Enterprise Architecture*, 11 (7), (S. 3). Arlington, MA: Cutter Consortium.

Tidwell, J. (1997). *Common Ground: A Pattern Language for Human-Computer Interface Design.* Online verfügbar unter http://www.mit.edu/~jtidwell/common_ground.html, zuletzt aufgerufen am 29. Januar 2009.

Tidwell, J. (2005). *Designing Interfaces. Patterns for Effective Interaction Design.* Online verfügbar unter http://designinginterfaces.com/, zuletzt aufgerufen am 29. Januar 2009.

Tidwell, J. (2005). *Designing Interfaces. Patterns for Effective Interaction Design.* Sebastopol, CA: O'Reilly.

Tohidi, M., Buxton, W., Baecker, R., & Sellen, A. (2006). Getting the right design and the design right. In ACM (Hrsg.), *Proceedings of the SIGCHI conference on Human factors in computing systems* (S. 1243–1252). New York: ACM (Conference on Human Factors in Computing Systems).

Toxboe, A. (2007). *UI Patterns. User Interface Design Pattern Library.* Online verfügbar unter http://ui-patterns.com/, zuletzt aufgerufen am 29. Januar 2009.

van Duyne, D. K., Landay, J. A., & Hong, J. I. (2006). *The Design of Sites: Patterns for Creating Winning Web Sites (2nd Edition).* Upper Saddle River, NJ: Prentice Hall PTR.

van Welie, M. (2001). *Interaction Design Patterns.* Online verfügbar unter http://www.welie.com/patterns/index.php, zuletzt aufgerufen am 29. Januar 2009.

van Welie, M. & van der Veer, G. C. (2003). *Pattern Languages in Interaction Design: Structure and Organization.* Online verfügbar unter http://www.welie.com/papers/Welie-Interact2003.pdf, zuletzt aufgerufen am 29. Januar 2009.

Virzi, R. A., Sokolov, J. L., & Karis, D. (1996). Usability problem identification using both low- and high-fidelity prototypes. In ACM (Hrsg.), *Proceedings of the SIGCHI conference on*

Human factors in computing systems (S. 236–243). New York: ACM (Conference on Human Factors in Computing Systems).

Walker, M., Takayama, L., & Landay, J. A. (2002). *High-fidelity or low-fidelity, paper or computer. Choosing attributes when testing web prototypes.* University of California, Berkeley, CA. Online verfügbar unter http://dub.washington.edu:2007/projects/fidelity/pubs/Walker_HFES_2002.pdf, zuletzt aufgerufen am 29. Januar 2009.

Warfel, T. (2009). *First Prototyping Survey Results*. Online verfügbar unter http://toddwarfel.com/archives/first-prototyping-survey-results/, zuletzt aufgerufen am 29. Januar 2009.

Wikipedia: PowerPoint. Online verfügbar unter http://de.wikipedia.org/wiki/PowerPoint, zuletzt aufgerufen am 29. Januar 2009.

Wikipedia: Heuristik. Online verfügbar unter http://de.wikipedia.org/wiki/Heuristik, zuletzt aufgerufen am 29. Januar 2009.

Wikipedia: Laufzeitumgebung. Online verfügbar unter http://de.wikipedia.org/wiki/Laufzeitumgebung, zuletzt aufgerufen am 29. Januar 2009.

Wikipedia: Varianz. Online verfügbar unter http://de.wikipedia.org/wiki/Varianz, zuletzt aufgerufen am 29. Januar 2009.

Wimmer, E. (2007). *Interaction Design: Axure and Scriptaculous*. Online verfügbar unter: http://www.interactiondesign.at/axure-und-scriptaculous. zuletzt aufgerufen am 29. Januar 2009.

Wong, Y. Y. (1992). Rough and ready prototypes: lessons from graphic design. In ACM (Hrsg.), *Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems* (S. 83–84). New York: ACM (Conference on Human Factors in Computing Systems).

Yahoo. (2006). *Design Pattern Library.* Online verfügbar unter http://developer.yahoo.com/ypatterns/, zuletzt aufgerufen am 29. Januar 2009.

Zapata, A. (2006). *The Guided Wireframe Narrative for Rich Internet Applications.* Online verfügbar unter http://www.boxesandarrows.com/view/the_guided_wire, zuletzt aufgerufen am 29. Januar 2009.

9 Anhang

9.1 Test Einführung

Begrüssung

Wir danken Ihnen sehr, dass Sie sich für unsere Usability Evaluation zur Verfügung gestellt haben. Wie Sie vielleicht bereits bei der Rekrutierung erfahren haben, geht es um unsere Masterarbeit in Human Computer Interaction Design. Wir versuchen die Benutzerfreundlichkeit der neuen Möglichkeiten von Web Applikationen zu testen. Dies anhand eines Reiseplaners. Wir haben rund um diesen Reiseplaner kleine Aufgaben vorbereitet, die Sie bitte lösen möchten. Wir werden Ihnen jede Aufgabe einzeln vorlegen und dabei das Vorgehen und die Aufgabenstellung erklären.

Die Aufgaben haben wir als sogenannte Prototypen in verschiedenen Techniken umgesetzt. Das bedeutet, dass eigentlich alles an Funktionalität ein "Fake", also nur vorgetäuscht ist. Aber wie gesagt, wir werden Ihnen bei jeder Aufgabe kurz erklären, wie wir dabei vorgehen und wie sie funktioniert.

Ich möchte noch einmal betonen, dass es beim Test um die Benutzerfreundlichkeit der WebApplikation geht. Auf keinen Fall geht es darum, Ihre Kenntnisse zB. der Arbeit mit dem Computer, zu testen. Sie können auf keinen Fall Fehler machen. Wenn etwas nicht so klappt, wie Sie es erwarten, dann liegt es daran, dass die Applikation bzw. der Prototyp noch verbessert werden muss. Zögern Sie also bitte auch nicht, Ihre Beobachtungen und Ihre Kritik zu äussern. Dazu haben wir Sie eingeladen.

Diese Testsitzung wird eine knappe Stunde dauern. Sie können die Testsitzung aber jederzeit abbrechen, wenn es Ihnen nicht mehr wohl ist, oder wir können auch einzelne Teilaufgaben abbrechen oder überspringen, wenn Sie dies wünschen.

9.2 Testaufgaben 2. Iteration

Aufgabe „Schieberegler (Slider)“

Sie sind auf der Suche nach einem Hotel in Zürich. Um einen Überblick über das Angebot zu erhalten, schauen Sie sich an, welche Hotels in Ihrer Preiskategorie zur Verfügung stehen.

Sie und Ihre Reisebegleitung sind sich noch nicht einig, welches Budget Sie dafür einsetzen möchten. Sie schauen sich deshalb nacheinander die folgenden Preiskategorien an:

a) Luxusklasse (Fr. 300.- bis Fr. 400.-)

b) mittleres Preissegment (Fr. 100.- bis Fr. 200.-)

c) Billigunterkünfte (unter Fr. 100.-)

d) nachdem Sie die Hotels in den verschiedenen Preisklassen angeschaut haben, entschliessen Sie sich schliesslich für das Hotel, das am nächsten beim See liegt.

Aufgabe „Editierbare Tabellen“

Um einen besseren Überblick über Ihre geplante Reise zu haben, haben Sie die einzelnen Stationen tabellarisch aufgelistet. Nun möchten Sie die Tabelle an den derzeitigen Stand der Planung anpassen:

- In Luzern bleiben Sie eine Nacht länger. Sie haben da ein Konzert gesehen, das Sie interessiert. Die darauffolgenden Daten ändern sich dadurch aber nicht.
- Zu Bern möchten Sie sich noch notieren, unbedingt den Zytgloggenturm zu besuchen
- Sie verlängern Ihre Ferien und verbringen zusätzlich als letzte Station noch 3 Tage (vom 3.11. - 6.11.) in Leukerbad zum Entspannen

Aufgabe „Rich Text Editor“

Sie haben ein Wochenende in Zürich verbracht. Leider hat Sie das Hotel krass enttäuscht. Um andere vor dem Hotel zu warnen, kommentieren Sie es umgehend auf der Hotelbewertungsseite:

„Das Hotel liegt an einer *sehr lauten* Strasse. Das Personal war zudem unfreundlich. **Nicht empfehlenswert**.“

(Bitte schreiben Sie den Text genau so, mit den *kursiven* und den **fetten** Stellen.)

Aufgabe „Eingabevorschläge (Type & Suggest)“

Sie möchten im Routenplaner die Strecke von Zürich nach Murten anzeigen lassen. Geben Sie Start- und Zielort ein.

Aufgabe „Pulldown Menus“

Sie befinden sich auf der Homepage des Reisebüro AMY. Neben dem eigentlichen Angebot des Reisebüros haben Sie auch die Möglichkeit, sich die Fahrpläne anzusehen. Gehen Sie auf den SBB Fahrplan.

(Bitte nehmen Sie dafür die Dienste der Reisebürowebsite in Anspruch. Geben Sie die URL der SBB nicht direkt ein.)

Aufgabe „Kontext Menu“

Sie haben im Reiseplaner eine Übersicht über Ihre Reiseroute erstellt. Der Reiseplaner bietet Ihnen aber noch zusätzlichen Komfort: Sie können sich von den Stationen Ihrer Reiseroute weitergehende Informationen anzeigen lassen.

Sie interessieren sich speziell für die Veranstaltungen in Bern.

Aufgabe „Navigation in Karten“

Sie haben auf der Karte bereits einige Stationen Ihrer Reise markiert. Sie sind noch unschlüssig, ob Sie Ihre Reise noch um einige Tage im nahen Ausland verlängern sollten. Sie hoffen, ein Blick auf die Karte helfe Ihnen bei der Entscheidungsfindung. Schauen Sie sich auf der Karte das angrenzende Gebiet von Deutschland, Oesterreich, Italien und Frankreich an. Sie entscheiden sich zuletzt für Mulhouse.

Aufgabe „Drag und Drop“

Zur Vorbereitung Ihrer Reise gehört auch noch der Erwerb von Reisezubehör. Kaufen Sie im Reiseshop einen Rucksack und eine Reisetasche.

Aufgabe „Filterung von Suchergebnissen“

Sie haben sich im SBB Fahrplan bereits die Züge von Zürich nach Murten ausgeben lassen.

Da Sie planen, auch Ihre Velos mit auf die Reise zu nehmen, wollen Sie sehen, in welchen Zügen dies möglich ist. Da sich möglicherweise noch einige Schweizer Freunde dazugesellen werden, sollte der Transport von Gruppen möglich sein.

Aufgabe „Validierung von Eingaben“

Ihre Erfahrung im Hotel „Goldiges Schäfli“ war ziemlich gemischt.

Das Hotel lag ganz ausgezeichnet und war auch sehr sauber. Leider mussten Sie aber so früh abreisen, dass der Frühstücksraum noch gar nicht geöffnet war. Das Personal war zwar sehr korrekt und auch freundlich. Bei allem Bedauern war es jedoch nicht in der Lage, für Sie speziell ein früheres Frühstück zu arrangieren. Für Sie ist es sehr wichtig, dies der Hotelführung zu melden.

Dazu benutzen Sie das Hotelbewertungsformular, welches das Hotel auf seiner Website anbietet.

Prototyping of Rich Internet Applications - Zweite Iteration

Testtag: ( ) Montag, 13. Okt.
( ) Dienstag, 14. Okt.
( ) Donnerstag, 16. Okt.

Nummer des/ der TestteilnehmerIn: .....

Name des/ der TestteilnehmerIn: .....

Dein Name: .....

	Aufgabe 1	2	3	4	5	6	7	8	
*"Er/ Sie **zögerte** bei der Aufgabenbearbeitung."*									a)
*ie hatte **Schwierigkeiten** bei der Aufgabenbearbeitung."*									a)
*"Er/ Sie war **langsam** bei der Aufgabenbearbeitung."*									a)
***"Insgesamt** kam er/ sie ... zurecht."*									b)

	a)	b)
1 =	gar nicht	sehr gut
2 =	kaum	gut
3 =	mittelmässig	mittelmässig
4 =	ziemlich	schlecht
5 =	ausserordentlich	sehr schlecht

9.4 „Objektiver“ Fragebogen

Fragebogen **Datum** ---------- **Testperson** ----------

RIA-Pattern ---------- **Test-Methode** ----------

	Stimme gar nicht zu				Stimme voll und ganz zu
Ich kann mich daran erinnern, diese Art der Bedienung bereits **früher schon einmal** kennen gelernt zu haben.	1	2	3	4	5
Irgend etwas an diesem Prototypen hat mich bei der Erledigung der Aufgabe **gestört**. (**Was genau** war dies?)	1	2	3	4	5
Die Bildschirmelemente des Prototypen **sehen genau so aus** wie bei einem fertigen System.	1	2	3	4	5
Die Bildschirmelemente **verhalten sich genau so** wie bei einem fertigen System.	1	2	3	4	5
Der Prototyp reagierte auf Befehle genau **so schnell** wie ein fertiges System.	1	2	3	4	5

Weitere Kommentare
