UNIVERSITÄT KOBLENZ · LANDAU
Institute for Web Science and Technologies

EyeVisionSearch

Nutzung von Blickerfassungsgeräten zur Verbesserung der Bedienung von Bildersuchmaschinen

Diplomarbeit

zur Erlangung des Grades eines
Diplom-Informatikers
im Studiengang Informatik

vorgelegt von

Leon Kastler

Erstgutachter: Prof. Dr. Steffen Staab
Institute for Web Science and Technologies
Zweitgutachter: Juniorprof. Dr. Ansgar Scherp
Institute for Web Science and Technologies

Koblenz, im November 2011

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

.....
(Ort, Datum) (Unterschrift)

Danksagung

Ich möchte mich hiermit bei Herrn Prof. Dr. Staab, Herrn Juniorprof. Dr. Scherp und Frau Dipl.-Inf. Walber für die Betreuung während meiner Diplomarbeit bedanken.
Des Weiteren möchte ich mich bei meinen Eltern, Stephanie Schneider, Jürgen Starek für ihre Hilfe sowie allen Teilnehmern der Evaluation bedanken.

Inhaltsverzeichnis

1 Einleitung **13**

1.1 Motivation 13

1.2 Aufbau der Arbeit 14

2 Szenarien **16**

2.1 Suche nach einem guten Portrait einer berühmten Persönlichkeit 17

2.2 Suche nach einem Bildschirmhintergrund 17

2.3 Suche nach Bildern für eine Komposition 18

3 Verwandte Arbeiten **19**

3.1 Begriffserklärungen 19

3.2 Verwendung von Eyetracking in der Informatik 21

3.2.1 Eyetracking im textbasiertem Information Retrieval 22

3.2.2 Eyetracking im bildbasierten Information Retrieval 23

3.2.3 Eyetracking als implizites Relevanz-Feedback im Information Retrieval 24

4 Anforderungen an *EyeVisionSearch* **27**

4.1 Anwendungsfall 27

4.2 Die Google Bildersuche 29

4.2.1 Übersicht 29

4.2.2 Aufbau 31

4.2.3 Einschränkungen in Bezug auf den Versuchsaufbau 32

4.3 Das Clipboard 33

4.4 Anforderungen 34

4.4.1 Clipboard 34

4.4.2 Basissystem 35

4.4.3 Eyetracking 35

5 Konzeptioneller Entwurf *EyeVisionSearch* 36

5.1 Architektur von *EyeVisionSearch* 36
5.2 Komponenten 38
5.2.1 Main 38
5.2.2 BL-Schnittstelle 38
5.2.3 Evaluationskomponente 39
5.2.4 Clipboard 39
5.3 Dokumente 39
5.4 Auswertung 40

6 Prototyp 41

6.1 Mozilla Add-on SDK 41
6.1.1 Kommandozeile 42
6.1.2 Ordnerstruktur 43
6.1.3 CommonJS - Modularität in JavaScript 43
6.1.4 Allgemeine Eigenschaften der APIs 44
6.1.5 Page-Mod API 45
6.1.6 Panel - Ein einfacher Dialog 47
6.1.7 Widget API 48
6.1.8 Notifications - Erkennung von Browser-Events 49
6.2 Text 2.0 50
6.2.1 Komponenten von Text 2.0 50
6.2.2 Kommunikation zwischen Eyetracker und Webseiten 53
6.3 Implementierung der Komponenten 54
6.3.1 Main 54
6.3.2 Initialisierung 54
6.3.3 Speicherung der Kommunikation 57
6.3.4 Hilfsfunktionen 57
6.3.5 Kommunikation vom Eyetracker zum Clipboard 60
6.3.6 Der Evaluationswizard 62
6.3.7 Clipboard Darstellung und Kommunikation 64
6.4 Auswertungstool 65

6.4.1 JSON in Java 65
6.4.2 JOpenDocument 66
6.4.3 Das Programm *tool-evaluation* 68

7 Evaluation **72**
7.1 Entwurf 72
7.1.1 Teilnehmer 73
7.1.2 Aufgaben 74
7.1.3 Erhobene Daten 75
7.1.4 Auswertung der erhobenen Daten 76
7.2 Durchführung 77
7.3 Ergebnisse 78
7.3.1 Effektivität 78
7.3.2 Effizienz 78
7.3.3 Benutzerzufriedenheit 79
7.4 Diskussion der Ergebnisse 79
7.4.1 Effektivität 79
7.4.2 Effizienz 81
7.4.3 Benutzerzufriedenheit 84
7.4.4 Kommentare der Probanden 87
7.4.5 Interpretation 88

8 Fazit **90**
8.1 Problembehandlung 90
8.2 Ausblick 91

Abbildungsverzeichnis

4.1 Use-Case Diagramm für *EyeVisionSearch* 28
4.2 Darstellung der Suchergebnisse für die Anfrage „Michael Jackson“ [Stand: Mitte 2011] 30
4.3 Weiterführende Darstellung der Google Bildersuche 32
4.4 Clipboard von *EyeVisionSearch* 33

5.1 Darstellung der Komponenten und ihrer Beziehungen 36
5.2 Informationsfluss des Evaluations-Datenstroms 37
5.3 Evaluationsobjekt-Datenfluss zwischen den Komponenten 38

6.1 Schematische Darstellung der Kommunikation über Page Worker (Darstellung von addons.mozilla.org) 47
6.2 Screenshot des Text 2.0 Servers 51
6.3 Screenshot des Text 2.0 Diagnose Programms 52
6.4 Slider, oben: Initialzustand, keine Darstellung des Handler; unten: Handler Darstellung nach Interaktion mit Benutzer 64
6.5 Klassendiagramm des Auswertungstools mit verwendeten Bibliotheken (dunkelgrau) und zugehörigen Dokumenten (hellgrau) . . 68

7.1 Allgemeine Informationen der Probanden nach Gruppen 74
7.2 Aufgaben in der Evaluation 75
7.3 Aufgabenbezogene Fragebögen, aufgeteilt in Gruppe A (Baseline) und Gruppe B/C (Clipboard-Varianten) 76
7.4 Finale Fragebögen, aufgeteilt in Gruppe A (Baseline) und Gruppe B/C (Clipboard-Varianten) 77
7.5 Aufbereitung der Dauer und der Ereignisse nach Aufgaben und Gruppen 80

7.6 Average-High-Low Diagramm der Dauer, aufgeschlüsselt nach Gruppen 81
7.7 Auswertung der aufgabenbezogenen Fragebögen nach Aufgaben und Fragen/Gruppen (1 = keine Zustimmung, 5 = Volle Zustimmung) 82
7.8 Aufbereitung der abschließenden Fragebögen nach Aufgaben und Fragen/Gruppen (1 = keine Zustimmung, 5 = Volle Zustimmung) 83
7.9 Aufgabenbezogener Fragebogen: Mann-Whitney U-Test Ergebnisse über Fragen und Gruppen (Signifikanzniveau $\alpha = 0.05$) 83
7.10 Abschließender Fragebogen: Mann-Whitney U-Test Ergebnisse über Fragen und Gruppen (Signifikanzniveau $\alpha = 0.05$) 84
7.11 Detailansicht Mann-Whitney U-Test Ergebnis: Aufgabe T3, Frage Q6 85
7.12 Detailansicht Mann-Whitney U-Test Ergebnis: Aufgabe T5, Frage Q6 85
7.13 Detailansicht Mann-Whitney U-Test Ergebnis: Aufgabe T5, Frage Q9 86

1 Einleitung

1.1 Motivation

Suchmaschinen sind in der heutigen Zeit ein wichtiger Bestandteil der täglichen Nutzung des Internets. Sie belegen vier Plätze der Top Ten Seiten des Internets [1]. Eine Form von Suchmaschinen-Nutzung stellt die Bildersuche dar, die zunehmend an Bedeutung gewinnt, da die Menge der Bilder stetig steigt. Bildersuche ist im Vergleich zur Textsuche ein junger Forschungsbereich.

Die meisten großen, kommerziellen Bildersuchen, die momentan im Internet zur Verfügung stehen, wie Google Images[1], Yahoo Images[2] oder Picsearch[3], arbeiten mit textbasierten Anfragen auf Metadaten. Diese Metadaten werden durch das Parsen der textbasierten Dokumente erstellt, in denen die Bilder eingebettet sind. Zudem bieten die meisten Suchmaschinen die Möglichkeit, die Anfrage um einige Restriktionen, wie beispielsweise die Größe des Bildes oder das Format, zu ergänzen.

Ein anderes Verfahren stellen Bilderhosting-Dienste, wie Flickr[4], bereit. Hier werden hochgeladene Bilder durch den Urheber mit Tags versehen, mit denen die Anfragen verglichen werden. Textbasierte Bildersuche hat den Vorteil, dass sie den meisten Benutzern bekannt ist, da sie sich ähnlich verhalten können wie bei der textorientierten Websuche allgemein. Allerdings ist es oft schwer, eine exakte Anfrage zu formulieren. Die Ergebnisse sind oft ungenau [2].

Dadurch ergeben sich mehrere Probleme: es ist für manche Suchen notwendig, unterschiedliche Suchanfragen zu stellen. Hierbei ist es leicht möglich, die Übersicht über die verschiedenen Ergebnismengen zu verlieren. Ein anderes Problem stellt sich bei Suchanfragen, die nicht exakt formuliert werden

[1] http://www.google.com/images
[2] http://images.search.yahoo.com/
[3] http://www.picsearch.com
[4] http://www.flickr.com

können. Der Suchbegriff *Wallpaper* [3] ist ein gutes Beispiel, da er einerseits sehr häufig verwendet wird und andererseits eine sehr große Ergebnismenge bietet. Somit muss der Benutzer die Ergebnismenge über mehrere Seiten durchsuchen, um ein geeignetes Bild zu finden. Hierdurch leidet ebenfalls die Übersicht.

In der Diplomarbeit soll die Verwendung und Möglichkeit zur Einbindung eines Eyetrackers in der Bildersuche untersucht werden. Eyetracker sind Geräte zur Blickerfassung. Sie werden häufig in Design- und Usabilitystudien verwendet, um Informationen über den Umgang der Benutzer mit dem Produkt zu untersuchen. Seit einiger Zeit werden Augenbewegungen auch zur Erkennung von benutzerrelevanten Informationen und Bereichen verwendet, wie zum Beispiel bei dem Projekt Text 2.0[5] [4]. Hierbei werden Blickrichtung und -fixierung benutzt, um eine Interaktion mit dem Leser eines Textes auf eine möglichst einfache, dabei aber subtile Weise zu ermöglichen.

Diese Eigenschaften des Eyetracking sollen im Rahmen dieser Arbeit für die Bildersuche genutzt werden. Dem Benutzer wird ein *Clipboard* angeboten, ähnlich einer *„kürzlich angesehene Artikel“*-Seitenleiste, wie sie von Amazon und anderen Webseiten verwendet wird. In diesem Clipboard werden die Bilder der Ergebnismenge einer Bildersuchanfrage vermerkt, die für den Benutzer interessant sind. Diese werden erkannt durch die Interpretation der Daten des Eyetrackers. Das Clipboard agiert unabhängig von der zugrundeliegenden Bildersuchmaschine, Es werden nur die angebotenen Bilder verwendet. Die Informationen des Clipboards werden ausschließlich lokal genutzt. Die Verwendung eines Clipboards ist eine gute Möglichkeit, um die Übersicht über die interessanten Kandidaten zu behalten und diese über mehrere Anfragen hinweg zu vermerken. Es gilt zu untersuchen, inwieweit diese Ergänzungen die Bedienung von Bildersuchen verbessern.

1.2 Aufbau der Arbeit

Die Arbeit ist in acht Kapitel gegliedert. Das erste Kapitel gibt eine allgemeine Einführung in das Themengebiet und erläutert die Struktur der Arbeit. Das

[5] http://www.text20.net

zweite Kapitel zählt mehrere Szenarien auf, die häufige Arbeitsabläufe im Zusammenhang mit webbasierten Bildersuchen modellieren. An diesen gilt es zu untersuchen, ob die Verwendung eines Eyetrackers den Benutzer bei der Bearbeitung der Aufgabe unterstützen kann. Die Szenarien bilden zudem die Grundlage für die Aufgabenstellungen in der durchgeführten Evaluation. Im 3. Kapitel, werden verwandte Arbeiten zum Information Retrieval, im Rahmen der Nutzung von Blickerfassungsgeräten, dargestellt und mit der Ausrichtung der Diplomarbeit verglichen.

In den nächsten drei Kapiteln werden die Anforderungen, das Konzept und die Implementierung des Prototypsystems - in dieser Arbeit *EyeVisionSearch* genannt - wie folgt besprochen. Im 4. Kapitel „Anforderungen an *EyeVisionSearch*“ wird ein Anwendungsbeispiel des Systems dargestellt und die einzelnen Prozesse genannt, die zu seinem Funktionieren notwendig sind. Darauf aufbauend werden die Anforderungen an das System vorgestellt. Im 5. Kapitel „Konzeptioneller Entwurf *EyeVisionSearch*“ wird das grundlegende Konzept des Systems beschrieben. Es enthält eine detaillierte Beschreibung der einzelnen Komponenten sowie der Datenflüsse und aller Dokumente, die vom System erstellt oder benötigt werden. Im 6. Kapitel „Prototyp“ werden der eigentliche Prototyp und seine Implementierung beschrieben. Hierzu gehören auch die einzelnen APIs, welche zu seiner Erstellung benötigt werden. Nach der Besprechung der Aspekte der Softwareentwicklung werden der Aufbau, der Verlauf und die Ergebnisse der Evaluation im darauffolgenden 7. Kapitel „Evaluation“ besprochen. Am Ende steht als letztes Kapitel das Fazit, in dem der Verlauf der Arbeit reflektiert und ein Ausblick auf mögliche Verbesserungen gegeben wird.

2 Szenarien

Das Ziel der Diplomarbeit ist zu untersuchen, inwieweit Blickerfassungsgeräte eingesetzt werden können, um die Bedienbarkeit webbasierter Bildersuchen zu erweitern oder gar zu verbessern. Hierzu wird eine Evaluation durchgeführt. Innerhalb dieser werden den Probanden die nachfolgenden Szenarien vorgegeben, die sie zu bearbeiten haben. Die Szenarien basieren auf Anwendungfällen, die einen häufig bei der Bildersuche auftretenen Vorgang in der alltäglichen Verwendung von Bildersuchen beschreiben. Zur Bewältigung der Aufgaben sollen alle Probanden ausschließlich die Google Bildersuche, als Beispiel einer webbasierten Bildersuche, verwenden. Innerhalb einzelner Aufgaben darf der Benutzer beliebige Anfragen stellen, um die gegebene Aufgabe zu lösen. Zudem ist es erlaubt, alle verfügbaren Filter der Bildersuche, wie die Größe oder den Typ des Bildes, zu nutzen.

Die Probanden werden in drei Gruppen eingeteilt. Eine Gruppe verwendet für die Bearbeitung der Aufgaben die herkömmliche Google Bildersuche. Die anderen beiden Gruppen verwenden jeweils eine erweiterte Bildersuche, die durch ein Clipboard ergänzt wird. Bei der zweiten Gruppe wird der Inhalt des Clipboards durch eine Relevanzfunktion bestimmt, die vollständig auf den von Google präsentierten Informationen basiert. Bei der dritten Gruppe wird der Inhalt des Clipboards ausschließlich durch die Auswertung der Informationen des Blickerfassungsgeräts bestimmt. Durch den Vergleich dieser Gruppen, im Besonderen in Bezug auf die Faktoren Effektivität, Effizienz und Usability, soll festgestellt werden, ob die Erweiterung eine Verbesserung für die Bedienung der Bildersuche darstellt.

Die Szenarien wurden mit Hilfe von *Google Insights for Search*[1] erstellt. Google Insights for Search ist ein Service von Google. Er ermöglicht die Analyse einer beliebigen Suchanfrage auf geographische Herkunft und zeitliche Ent-

[1] `http://www.google.com/insights`

wicklung. Durch die Verwendung dieses Services zur Erstellung der Szenarien soll ihre Realitätsnähe gewährleistet werden.

2.1 Suche nach einem guten Portrait einer berühmten Persönlichkeit

Als erstes Szenario wird den Testpersonen die Aufgabe gestellt, ein gutes Portrait einer berühmten Persönlichkeit zu finden. Bilder von bekannten Persönlichkeiten gehören zu den häufigsten Anfragen an die Bildersuche. Unter den Top Ten der Rising Searches, den Suchbegriffen mit steigender Nachfrage, befinden sich sechs personenbezogene Suchbegriffe [5]. Das Problem hierbei ist, dass die Bilder nicht immer den geforderten Eigenschaften eines guten Portraits entsprechen. Zwar besitzt Google Bildersuche einen Portrait Filter, allerdings lassen sich damit nicht alle irrelevanten Bilder entfernen. So werden zum Beispiel Werbeaufnahmen oder Bilder von kleinen Gruppen nicht durch diesen Filter erkannt. Das Clipboard könnte hier genutzt werden, um alle guten Bilder in einer Vorauswahl zu sammeln und danach das beste zu benutzen. Dabei könnte die Möglichkeit, sich alle Kandidaten nebeneinander anzusehen, vorteilhaft sein.

Beispiele für mögliche Evaluationsaufgaben sind:

- Portraitbild von '„Michael Jackson“
- Portraitbild von „Lady Gaga“

2.2 Suche nach einem Bildschirmhintergrund

Im zweiten Szenario soll die Testperson einen Bildschirmhintergrund, beispielsweise für einen Austellungs-Computer, finden. Der Computer soll auf einer lokalen Messe aufgestellt werden. Daher wäre es eine gute Idee, wenn der Bildschirmhintergund die Silhouette der jeweiligen Stadt wäre.

Der englische Suchbegriff *Wallpaper* wird sehr häufig in Bildersuchen gesucht [3]. Die Anfrage hierzu ist zumeist schwer zu formulieren. Man kann zwar grobe Beschreibungen wie *Natur* oder *Schwarz Weiß* hinzufügen. Allerdings kann man nur schwer die Ästhetik eines Bildes in exakte Begriffe fassen,

da der Term *schön* subjektiv ist. Dieser Aspekt ist besonders wichtig, da man das ausgewählte Bild häufig vor Augen hat und es einem gefallen sollte. Somit enthält die Ergebnismenge viele Bilder, die nicht den Ansprüchen genügen.

Ein Clipboard könnte hier vorteilhaft sein, da die möglichen Kandidaten über mehrere Seiten des Anfrageergebnisses oder auch auf mehrere Suchanfragen verteilt liegen können. Die Betrachtungsdauer eines Bildes kann hierbei unter Umständen genutzt werden, um einen guten Kandidaten zu erkennen und es im Clipboard zu vermerken. Beispiele für mögliche Evaluationsaufgaben sind:

- Bildschirmhintergrund mit dem Thema: „Romantischer Rhein“
- Bildschirmhintergrund mit dem Thema: „Ampelmännchen“

2.3 Suche nach Bildern für eine Komposition

Ein weiteres Szenario ist die Erstellung einer Komposition von Bildern. Die Aufgabe besteht darin, fünf Bilder von verschiedenen Sehenswürdigkeiten einer Stadt zu sammeln. Diese Bilder sollen als Komposition später zur positiven Darstellung der Stadt, beispielsweise im Rahmen einer Bundesgartenschau oder einer ähnlichen Veranstaltung, genutzt werden. Die Herausforderung hierbei ist, dass die Bilder der Ergebnismenge stilistisch sehr unterschiedlich sind. Auch hier erleichtert das Clipboard dem Benutzer unter Umständen die Entscheidungsfindung. Beispiele für mögliche Evaluationsaufgaben sind:

- Bilderkomposition mit dem Thema: „Loriot zu Weihnachten“
- Bilderkomposition mit dem Thema: „Sehenswürdigkeiten in Berlin“

3 Verwandte Arbeiten

Wie bereits erwähnt, ist das Ziel dieser Arbeit die Erweiterung und Verbesserung der Bedienbarkeit von webbasierten Bildersuchen. Dies soll erreicht werden durch den Einsatz von Blickerfassungsgeräten, auch Eyetracker genannt. Die Daten des *Blickerfassungsgeräts* sollen hierbei genutzt werden, um Informationen über die *Relevanz* des betrachteten Objekts für den Benutzer zu einer gegebenen Aufgabenstellung herzuleiten. Nachfolgend werden kurze Erläuterungen von für diese Thematik wichtigen Begriffen vorgestellt. Danach folgt eine Übersicht über Arbeiten im Bereich des Information Retrieval, die sich ebenfalls mit der Integration von Eyetrackerdaten beschäftigen.

3.1 Begriffserklärungen

Wie im oberen Absatz aufgezeigt, beschäftigt sich die Arbeit mit dem Begriff der Relevanz, der Funktion eines Blickerfassungsgeräts und damit, wie diese beiden in Zusammenhang gebracht werden können.

Begriffserklärung (Relevanz)**.** Nach Hjørland und Sejer Christensen [6] ist etwas relevant für eine gegebene Aufgabe, wenn es die Wahrscheinlichkeit erhöht, die Aufgabe zu lösen.

Nach der Definition der Wikipedia [7] beschreibt die Relevanz, wie gut ein erhaltenes Dokument aus einer Menge von Dokumenten das *Informationsbedürfnis (information need)* des Benutzers befriedigt.

Es ist schwer zu bestimmen, was eine relevante Information für die jeweilige Person ausmacht, da das Konzept der Relevanz ein multidimensionales, kognitives Konzept darstellt. [8]. Die Möglichkeit, einen Rückschluss auf die gesuchten Informationen zu finden, ist die Verwendung von *Relevanz-Feedback*.

Begriffserklärung (Relevanz-Feedback)**.** Relevanz-Feedback beschreibt die iterative Verbesserung der Antwort auf die Anfrage durch Interpretationen der zurückgegebenen Informationen des Benutzers.

Information Retrieval Systeme versuchen, durch die iterative Annäherung möglichst nahe an die Menge der relevanten Dokumente zu kommen. Man unterscheidet zwei Gruppen von Informationen des Nutzers, die als Relevanz-Feedback genutzt werden können:

- explizites Relevanz-Feedback
- implizites Relevanz-Feedback

Explizites Relevanz-Feedback bedeutet, dass der Benutzer direkt angibt, welche Ergebnisse für ihn relevant sind. Ein Beispiel für explizites Feedback ist eine „I like“ Schaltfläche unter den jeweiligen Dokumenten. Diese wird willkürlich durch den Benutzer betätigt, um anzuzeigen, dass das korrespondierende Dokument für ihn relevant ist. Das Verfahren ist sehr exakt, um relevante Dokumente für den Benutzer zu ermitteln. Allerdings hat es auch den Nachteil, dass der Benutzer zusätzliche Zeit aufbringen muss, um zu entscheiden, ob das Dokument für ihn relevant ist. Implizite oder auch indirekte Formen des Relevanz-Feedbacks stellen beispielsweise die Menge von Klicks dar, die ein Ergebnis zu einem bestimmten Suchbegriff erhält. Der Benutzer gibt nicht explizit an, dass das Ergebnis relevant ist. Da er ihm aber Aufmerksamkeit widmet, indem er es anklickt und die Quelle betrachtet, kann hieraus indirekt die Relevanz ermittelt werden. Diese Art des Feedbacks ist weniger exakt als die des expliziten Feedbacks, hat aber den Vorteil, dass sie parallel zur normalen Benutzung der Suche angewandt werden kann, ohne dass der Benutzer ihr Aufmerksamkeit zukommen lassen muss.

Im Folgenden wird erläutert, was Blickerfassungsgeräte sind und warum die durch sie ermittelten Informationen als implizites Maß für die Relevanz verwendet werden können.

Begriffserklärung (Blickerfassungsgerät)**.** Blickerfassungsgeräte, auch Eyetracker genannt, sind Geräte zur Aufzeichnung und Analyse von Blickbewegungen. [9]

Begriffserklärung (Blickbewegungen)**.** Unter Blickbewegungen versteht man die Bewegung der Augen, die den Blick willkürlich oder unwillkürlich auf ein Objekt ausrichten. [9]

Vor allem zwei spezielle Blickbewegungen werden beim Eyetracking betrachtet, nämlich Sakkaden und Fixationen.

Begriffserklärung (Sakkaden)**.** Sakkaden beschreiben Blicksprünge, also schnelle, plötzliche Veränderungen der Augenstellung. [9]

Begriffserklärung (Fixation)**.** Fixation beschreibt die Ausrichtung des Auges auf ein Objekt und die daraus resultierenden Minimalbewegungen oder Mikro-Sakkaden des Auges. [9]

Der Grund, warum besonders die Fixation eine wichtige Komponente in der Blickerkennung darstellt, wird von Lindsay und Norman [9] wie folgt beschrieben: Der Mensch besitzt nur in einem kleinen Bereich des Auges, der so genannten Sehgrube (Fovea centralis) innerhalb des Gelben Flecks (Macula lutea), eine sehr scharfe Bildwahrnehmung. In diesem Bereich befinden sich besonders viele Stäbchen, spezialisierte Sinneszellen der Netzhaut, die vor allem für die Farbwahrnehmung und Bildschärfe verantwortlich sind. Da der Bereich der Sehgrube sehr beschränkt ist (ca. 1.5mm Durchmesser), kann man aus Sakkaden und Fixationen herauslesen, welchen Bereichen der Benutzer seine Aufmerksamkeit zuwendet. Wie bereits erwähnt, werden Sakkaden und Fixationen von Eyetrackern aufgezeichnet. Diese Eigenschaft kann genutzt werden, um Informationen vom Benutzer über die Relevanz eines Ergebnisses einer Suchanfrage zu erhalten. Der Zusammenhang wurde beispielsweise auch von Kaski et al. [10] untersucht. Zu diesem Thema existieren bereits einige Studien, die sich mit dem Eyetracker befasst haben. Sie werden im Folgenden besprochen.

3.2 Verwendung von Eyetracking in der Informatik

Eyetracker finden in der Informatik in verschiedenen Bereichen Verwendung. Seit langem werden sie in Design- und Usability-Studien eingesetzt. Mit ihnen lassen sich die Blickrichtungen und Augenbewegungen der Testpersonen aufzeichnen, um so die Anordnung von Benutzeroberflächen zu verbessern.

Ein Beispiel hierfür ist die Usability-Studie von Google [11]. In dieser Studie untersuchte man bei Google, welche Bereiche eines Suchergebnisses von den Benutzern betrachtet werden. Hierbei wurde festgestellt, dass das vor allem die ersten zwei Suchergebnisse einer Textsuche sind. Zudem stellte man fest, dass das Universal Search Interface, also die Integration von Bildersuche durch Thrumbnails in die Textsuche, das Durchsuchen der Ergebnisse wenig beeinflusst. Allerdings sind die Thrumbnails hilfreich beim Finden von gewünschten Ergebnissen.

Auch im Bereich der Mensch-Maschine-Kommunikation wird Eyetracking verwendet. Hier hat es die Eingabefunktion, beispielsweise als Mausersatz für Personen mit körperlicher Einschränkung. Wie bereits beschrieben, können Eyetracker als zusätzliche Informationsquelle zusammen mit den bestehenden Eingabegeräten verwendet werden.

3.2.1 Eyetracking im textbasiertem Information Retrieval

Hardoon et al. versuchten 2007, einen Ansatz im Zusammenhang von Eyetracking und implizitem Relevanz-Feedback zu erarbeiten [12]. Mit Hilfe der Augenbewegungen sollten Informationen zur Herleitung von Anfragen gefunden werden. Ihr Ziel war es, aus den Wörtern und Begriffen, die sich die Benutzer betrachten, die Anfragen herzuleiten, für die sich die Benutzer interessieren. Hierzu führten sie eine Untersuchung durch, bei der sie den Testpersonen Abschnitte von Wikipedia-Artikeln zu einem bestimmten Thema zeigten und aufzeichneten, welche Begriffe betrachtet wurden. Auf Basis der gewonnenen Daten entwickelten sie ein System, das die Anfragen anhand der Augenbewegungen herleiten kann. Die Evaluation des Systems ergab, dass dieses Verfahren eine höhere Präzision im Vergleich zu einer rein textuellen Auswertung der Dokumente hatte. Zudem stellte sich auch heraus, dass explizites Relevanz-Feedback durch den Benutzer eine weitere Erhöhung der Präzision zur Folge hat. Jedoch ist das mit einem deutlichen Mehraufwand verbunden. Trotz des Bezugs auf textbasierte Systeme stellt diese Arbeit einen Ansatzpunkt für andere Systeme im Rahmen des Eyetracker-unterstützen Information Retrieval dar.

Ein weiteres Beispiel für den Einsatz von Eyetrackern zum Gewinnen von

implizitem Relevanz-Feedback in einem textbasierten Umfeld ist Text 2.0 von Buscher et al. [4] Das System ermöglicht die Interaktion zwischen einem gelesenen Text und dem Leser. Hierzu werden die Augenbewegungen des Lesers aufgezeichnet und somit die Textabschnitte erkannt, die gerade gelesen werden. Fixiert der Leser ein spezielles Wort oder liest er einen speziellen Abschnitt, können ihm weitere Informationen zu diesem Bereich angezeigt werden. Gleichzeitig können die Daten auch als implizites Relevanz-Feedback genutzt werden.

3.2.2 Eyetracking im bildbasierten Information Retrieval

In der Studie von Cosato et al. [13] wurden Eyetracker im Zusammenhang mit der Bildersuche verwendet. Sie untersuchten deren Einsatz in so genannten *Rapid Serial Visualisation Presentation Methods*, kurz RSVP. Diese werden genutzt, wenn die Datenbank eine große Menge von Bildern enthält, aber nur eine kleine Menge davon relevant ist. Sie verglichen hierzu verschiedene Präsentationstechniken des RSVP mit einer üblichen Grid-basierten Darstellung der Bilder. Im RSVP werden relevante Bilder mit der Maus durch den Benutzer markiert. Die Auswahl von Kandidaten bei den RSVP Darstellungen wurde in dieser Studie durch den Benutzer vorgenommen, indem er eine Taste auf der Tastatur betätigte und das System das momentan vom Benutzer fixierte Bild als Kandidaten vermerkte. Es handelt sich hierbei also um explizites Feedback, welches der Benutzer zu leisten hat. Die Varianten wurden nun in Bezug auf ihre Effektivität und die Ermüdung der Testperson verglichen. Hierbei stellten sie fest, dass die RSVP Techniken einen höheren Grad an Effektivität besitzen, gleichzeitig aber ermüdender sind.

Klami et al. untersuchten in ihrer Arbeit von 2007 [10], ob es möglich ist, die Relevanz eines Bildes durch Beobachtung der Augenbewegung herzuleiten. Sie stellten dabei fest, dass es sehr wohl möglich ist, die Relevanz von Bildern durch eine relativ geringe Anzahl von Features zu ermitteln. Allerdings wurde die Untersuchung nur mit sehr allgemeinen Suchbegriffen, wie *sports*, durchgeführt.

3.2.3 Eyetracking als implizites Relevanz-Feedback im Information Retrieval

Neben der Erfassung des Blickes auf das Bild an sich beschäftigt die Forschung vor allem die Betrachtung von bestimmten Bereichen der Bilder und damit, welche Informationen daraus gewonnen werden können.

Jeff et al. [14] benutzten Eyetracker, um zu untersuchen, wie Personen Bilder mit semantisch unterschiedlichen Kategorien betrachten. Den Benutzern wurden Bilder in einer zufälligen Reihenfolge gezeigt, die zu einer von fünf vorher bestimmten, semantischen Kategorien gehörten. Ihnen wurde gesagt, dass sie sich die Bilder ansehen sollten. Darüber hinaus wurden aber keine weitere Aufgaben gestellt. Die Daten wurden nun anhand verschiedener Gesichtspunkte untersucht. Als erstes wurde untersucht, inwieweit die Betrachtungsmuster verschiedener Betrachter für ein Bild ähnlich sind. Zweitens versuchte man festzustellen, wie sich die Betrachtungsmuster zwischen zwei Kategorien unterschieden. Für Punkt Eins stellte sich heraus, dass alle Betrachter ein Bild auf eine ähnliche Art betrachten. Allerdings gibt es hier Unterschiede in den jeweiligen semantischen Kategorien. So ist zum Beispiel die Ähnlichkeit der Muster für die Kategorie *handshake* oder *main object* wesentlich größer als für die Kategorie *landscape*, bei der nur eine geringere Konsistenz zwischen den jeweiligen Betrachtern existierte.

Bei den Unterschieden zwischen den Kategorien stellte man fest, dass sich hier auch die Kategorien *handshake* und *main object* in den Betrachtungsmustern stark unterschieden. Die Ergebnisse wurden genutzt, um damit eine automatische Klassifizierung auf Basis der Augenbewegungen durchzuführen. Hierzu wurde ein System entwickelt, das der Benutzer durch eine Auswahl an Beispielbildern trainieren kann. Dieser Klassifikator segmentiert die Bilder und ermöglicht dem Benutzer, ein hierarchisches Objektmodell zu erstellen. Obwohl die Arbeit als erste einen Zusammenhang zwischen der Beobachtung des Nutzers und dem Inhalt der Bilder im Bereich Information Retrieval zeigte, ist sie noch nicht weitreichend genug, da zur Erstellung des Klassifikators auf explizites Feedback des Benutzers gesetzt wurde.

Pasupa et al. [15] verglichen in ihrer Untersuchung die möglichen Kombinationen von Content Based Image Retrieval und von Informationen, die durch

den Eyetracker gewonnen wurden. Dieser Vergleich wurde in Bezug auf drei Fragestellungen untersucht. Die erste Fragestellung war, ob ein globales Modell existiert, mit dem sich anhand der Daten von mehreren Benutzern das Ranking eines Benutzers vorhersagen lässt. Die zweite Fragestellung ähnelte der ersten. Allerdings versuchten die Autoren hier nicht das Ranking eines Benutzers zu ermitteln. Stattdessen zielte ihr Ansatz darauf, das Ranking der Bilder einer gegebenen Seite anhand der Daten aller anderen Seiten zu bestimmen. Die dritte Frage, die sich die Forschungsgruppe stellte, war, ob sich das Ranking eines noch nicht gesichteten Bildes durch die bisherigen Daten des Benutzers bestimmen lässt.

Bei der ersten Fragestellung ergab sich nur ein marginaler Unterschied zwischen der Kombination beider Verfahren und dem ausschließlich inhaltsbasierten Ansatz. Für die zweite Fragestellung stellte sich heraus, dass die Kombination von Eyetrackinginformationen und inhaltsbasierten Information Retrieval Verfahren die besten Ergebnisse liefert. Generell ergab die Hinzunahme der Eyetrackinginformationen eine Verbesserung im Vergleich zu rein inhaltsbasierten Verfahren. Auch in der dritten Fragestellung erwies sich die Kombination beider Informationen als zuverlässiger als die anderen Methoden. Inhaltsbasierte Verfahren können hierbei zu Fehlinterpretationen führen, abhängig vom Thema der Bilder. Somit erwies sich in dieser Studie Eyetracking als nützlicher Zusatz zu den bisherigen inhaltsbasierten Verfahren, wenn sie auch mit einem höheren Aufwand verbunden sind.

Hardoon und Pasupa [16] führten eine weitere Studie basierend auf diesen Erkenntnissen durch. In dieser versuchten sie, die Probleme der letzten Studie zu korrigieren. So war eine Annahme der älteren Studie, dass für neue Bilder bereits Eyetracking-Daten vorhanden sind, unrealistisch. Sie gelangten jedoch, durch Modifikation der bestehenden Algorithmen, ebenfalls zu dem Ergebnis, dass Eyetracking in diesem Aufgabenfeld eine nützliche Erweiterung ist.

Ein weiteres Projekt, das im Zusammenhang mit Eyetracking und Image Retrieval erwähnt werden sollte, ist das Projekt GaZIR von Kozma und Klami [17]. In diesem Projekt wurde ein graphisches Interface für die inhaltsbasierte Bildersuche entwickelt. Es verwendet die Informationen des Eyetrackers, um Kandidaten unter den Bildern zu ermitteln, die für die nächste Anfrage verwendet werden.

Die genannten Ansätze von Eyetracking im Information Retrieval verfolgen das Ziel, die Daten der Augenbewegungen mit der inhaltlichen Analyse der Bilder zu kombinieren. Hierdurch soll die Suche nach Bildern verbessert werden, indem man nicht nur Eigenschaften von Bildern, wie Textur oder Farbe, extrahiert, sondern auch die Information des Benutzers, welche Bereiche und Inhalte für ihn besonders wichtig sind, einbindet. In diesen Ansätzen ist es allerdings notwendig, alle verwendeten Bilder vorher mit den Techniken des Content Based Image Retrievals zu analysieren. Diese Auswertung liegt bisher nur bei einem Bruchteil der im Internet vorhandenen Bilder vor. Zudem versuchen die meisten Ansätze, diese Informationen direkt zur Verbesserung der jeweiligen Suchverfahren zu nutzen. Der Ansatz dieser Diplomarbeit stellt dagegen eine Art Zwischenschritt dar, der vor allem der Verbesserung der Bedienbarkeit der existierenden Suchmaschinen dienen soll.

4 Anforderungen an *EyeVisionSearch*

Das System, welches für die Untersuchung der Aufgabenstellung entwickelt werden soll, trägt den Namen *EyeVisionSearch*. Die Aufgabe von *EyeVisionSearch* besteht aus zwei Teilen. Zum einen soll eine Verbindung zwischen Eyetrackern und einer bestehenden Bildersuche geschaffen werden. Die Informationen, die daraus gewonnen werden, sollen dann interpretiert werden, um ein Clipboard, einen separaten Bereich innerhalb der Bildersuche, mit den Bildern zu füllen, die für den Benutzer interessant sein können.

Die zweite Aufgabe, die *EyeVisionSearch* erfüllen soll, ist die Schaffung der Grundlagen für eine *summative Evaluation* des Systems selbst. Mit summativer Evaluation ist der Vergleich eines neuen Systems mit einer *Baseline*, einem bereits bestehenden Vergleichssystem, gemeint. In diesem Fall wäre das die Google Bildersuche ohne die Erweiterung durch das Clipboard. Das Design der Experimente wird allgemein als "*Between-Subjects*" bezeichnet. Bei diesem Design werden die unterschiedlichen Varianten des Systems jeweils einer Gruppe von Probanden zugewiesen, die diese in ihrem Experiment verwenden.

Im Folgenden werden die Anforderungen beschrieben, welche *EyeVisionSearch* zu erfüllen hat. Begonnen wird mit der Besprechung der Use-Case Analyse, die den grundlegenden Ablauf des Systems verdeutlichen soll. Daraus werden dann die detaillierten Anforderungen an die einzelnen Prozesse abgeleitet.

4.1 Anwendungsfall

Die zu erstellende Anwendung muss zwei unterschiedliche Anforderungsmengen erfüllen. Zum einen soll das System die Verbindung zwischen der Google Bildersuche und dem Eyetracker herstellen und die ermittelten Kandidaten in einem Clipboard darstellen. Die andere Anforderung ist die Umsetzung einer

Oberfläche für die Leitung eines Probanden durch eine summative Evaluation im Between-Groups Verfahren. Dies bedeutet, dass jeder Proband einer Variante von *EyeVisionSearch* zugeordnet wird, abhängig von seiner vorgegebenen Gruppe.

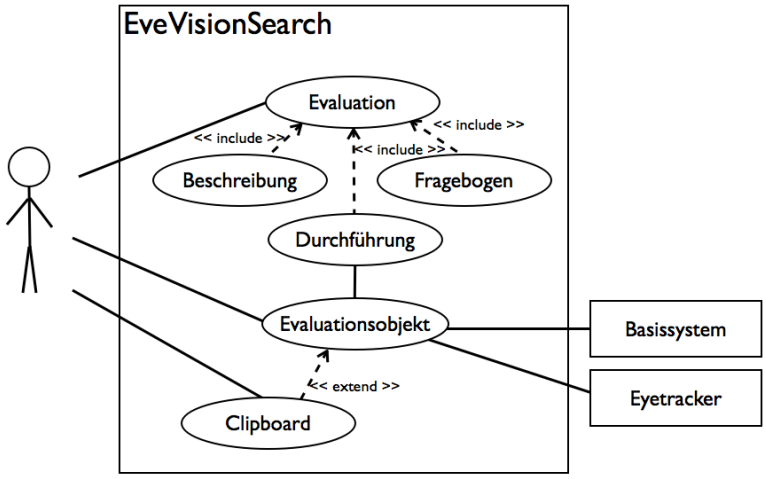

Abbildung 4.1: Use-Case Diagramm für *EyeVisionSearch*

Abbildung 4.1 stellt ein Use-Case Diagramm von *EyeVisionSearch* dar. Beginnt der Proband die Evaluation, befindet er sich im Evaluationsprozess. Dieser besteht aus drei Teilprozessen: *Beschreibungen*, *Durchführung* und *Fragebögen*. Der Prozess *Beschreibung* liefert dem Probanden Erklärungen über den Ablauf der Evaluation und Beschreibungen über das Evaluationsobjekt. Zudem erläutern sie dem Probanden die jeweiligen Aufgaben. *Durchführungen* enthalten die Bearbeitung der gestellten Aufgaben am Evaluationsobjekt. Das Evaluationsobjekt ist hierbei das Basissystem, die Google Bildersuche, mit eingebundenem Eyetracker sowie das Clipboard.

Während der Bearbeitung der Aufgabe zeichnet *EyeVisionSearch* alle Ereignisse des Basissystems und des Eyetrackers auf. Diese Daten werden, je nach Variante, interpretiert, wodurch die Reihenfolge der ausgewählten Bilder im

Clipboard beeinflusst wird. Zudem speichert *EyeVisionSearch* die aufgezeichneten Daten zur Auswertung der Evaluation ab. Diese Verzahnung von Evaluation und System ist ausdrücklich gewünscht, da hierdurch dem Probanden ein konsistentes Bild der Evaluation geboten wird und zudem die Mehrarbeit des Systems auf ein Minimum begrenzt werden kann.

Schließt der Proband die Durchführung ab, wird der letzte Teilprozess der Evaluation ausgeführt. Dieser ist der *Fragebogen*, der dem Probanden die zu beantwortenden Fragen stellt. Die Antworten werden nach der Bearbeitung durch den Probanden ebenfalls für die Auswertung abgespeichert.

Das System verwendet verschiedene Hilfsanwendungen. Als Grundlage wird das Basissystem verwendet, auf dem das Evaluationsobjekt aufbaut. Die Verwendung eines verbreiteten Basisystems vereinfacht die Zugänglichkeit des Gesamtsystems, wodurch die Durchführung der Evaluation einer größeren Gruppe von Probanden ermöglicht wird.

Sowohl die Ergebnisse der Fragebögen als auch die aufgezeichneten Daten der Aufgabe werden für die spätere Auswertung aufgezeichnet. Hierzu sollte ein gängiges Format verwendet werden, um die Hürde der Auswertbarkeit möglichst niedrig zu halten.

4.2 Die Google Bildersuche

Als Basissystem für die Implementierung des Systems wird die Google Bildersuche[1] in der deutschen Version verwendet.

4.2.1 Übersicht

Die Google Bildersuche ist eine der meistverwendeten Bildersuchen im Internet. Die Seite wurde im Juli 2001 das erste Mal öffentlich verfügbar. Innerhalb von zehn Jahren erhöhte sich die Anzahl der indizierten Bilder von 250 Millionen bis heute auf über Zehn Billiarden Einträge [18]. Täglich wird die Seite mehr als eine Milliarde Mal aufgerufen [19].

Um die Google Bildersuche zu verwenden, gibt der Benutzer eine textbasierte Anfrage auf der Webseite von Google ein. Ihm wird daraufhin eine Liste von

[1] images.google.de

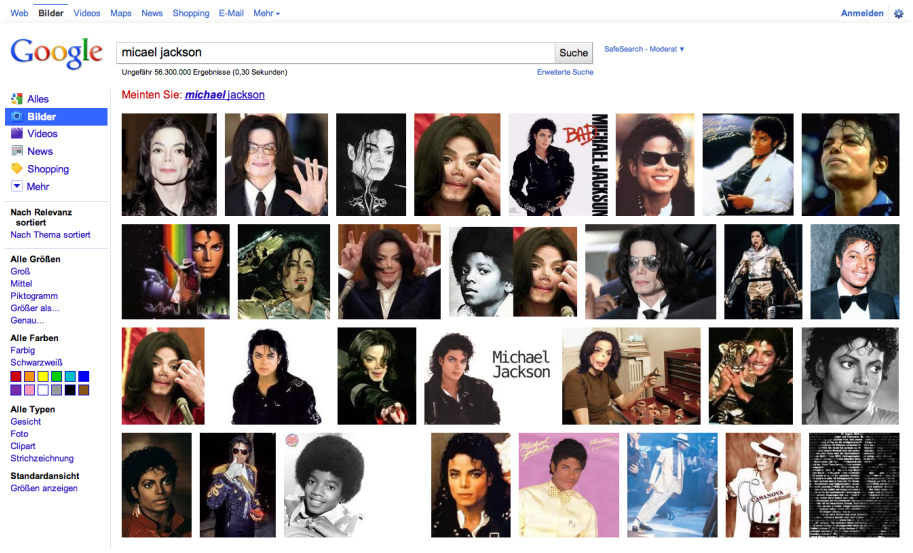

Abbildung 4.2: Darstellung der Suchergebnisse für die Anfrage „Michael Jackson“ [Stand: Mitte 2011]

passenden Bildern in Form von Thrumbnails, einer verkleinerten Darstellung der Bilder, präsentiert. Im Laufe der letzten Jahre hat die Seite einige Überarbeitungen der Oberfläche erfahren. Das letzte Update dieser Art wurde im Juli 2010 durchgeführt.

Die Bilder werden anhand der Schlüsselworte der Anfrage ausgewählt. Google benutzt hierbei mehrere Kriterien, wie den Namen der Datei, die URL auf das Bild sowie den umgebenen Text, in den das Bild eingebettet ist. Seit Oktober 2009 kann sich der Benutzer auch ähnliche Bilder zu bereits gefundenen Bildern anzeigen lassen. Diese Funktion ist inhaltsbasiert und benutzt Metriken wie Farbe oder Struktur eines Bildes. Dieses Feature ging aus dem Similar Images Projekt von Google hervor. Seit Mitte 2011 bietet Google zudem die Möglichkeit, eine rein inhaltsbasierte Suche per Query by Example durchzuführen. Im weiteren werden die Ausführungen allerdings auf die textbasierte Suche beschränkt, da das neue Feature erst im Verlauf der Arbeit zur Google Bildersuche hinzugefügt wurde und nur der Vollständigkeit halber hier genannt wird.

4.2.2 Aufbau

Wie bereits erwähnt, werden textbasierte Anfragen durch eine Liste von Bildern beantwortet. Die Bilder werden hierbei nach den angegebenen Suchbegriffen ausgesucht, wobei die Namen der Bilder, ihre URL sowie der umgebene Text innerhalb eines Dokuments in Betracht gezogen werden. Bild 4.2 dient hierbei als Beispiel. Im Bild wird das Ergebnis der Anfrage "Michael Jackson" dargestellt.

Im oberen, zentralen Bereich befindet sich das Eingabefeld, mit dem man eine neue Anfrage formulieren kann. Neben dem Suchen-Button erscheint die SafeSearch-Anzeige. Unterhalb des Suchen-Buttons befindet sich die Erweiterte Suche, mit der man die Suche verfeinern kann.

Im Hauptbereich befinden sich die Suchergebnisse in einem eng zusammenstehenden Layout. Damit soll es möglich sein, sich einen schnellen Überblick über die Bilder zu verschaffen.

Zu früheren Zeiten verwendete Google eine seitenorientierte Darstellung, ähnlich wie in der allgemeinen Suche. Diese wurde in der Bildersuche aufgebrochen. Man kann sich durch Weiterscrollen die nachfolgenden Seiten in dem momentanen Fenster anzeigen lassen. Sie werden im Hintergrund per *XMLHTTPRequest (XHR)*, also einer JavaScript-gesteuerten Anfrage an den Web Server, nachgeladen. Das soll die Suche nach einem Bild weiter vereinfachen, da man die Übersicht über alle Bilder zu einer Anfrage behält.

Ein weiteres Feature, welches erst Mitte 2011 öffentlich eingesetzt wurde, ist die *hover pane*, eine erweiterte Darstellung des Thrumbnails mit zusätzlichen Informationen, wie beispielsweise dem Namen, der URL oder der Größe. Sie wird aktiviert, indem der Benutzer den Mauszeiger einige Zeit über dem Bild fixiert. Neben den genannten Informationen kann er sich zusätzlich ähnliche Bilder oder unterschiedliche Größen des Bildes anzeigen lassen.

Am linken Rand befinden sich die verschiedenen Filter, mit denen der Benutzer die Suche weiter verfeinern kann. Hier kann er verschiedene Größen, Typen und Farben auswählen. Bei den Typen werden allgemeine Strukturen der Bilder beschrieben, wie Fotografien oder Zeichnungen. Beim Farbfilter werden Bilder ausgewählt, die überwiegend die gesuchte Farbe enthalten. Weitere Filter sind solche über den Zeitpunkt der Aufnahmen in den Index und Sortie-

rungen nach Relevanz oder Thema.

Der Benutzer kann zudem das Bild anklicken und somit auf die im Kontext der Arbeit so genannte *weiterführende Ansicht* gelangen. Diese, dargestellt in Abbildung 4.3, zeigt eine zugehörige Darstellung der weiterführenden Ansicht für ein gegebenes Bild. Im oberen Rahmen werden das Bild sowie weitere Informationen angezeigt. Sie beinhalten die Größe, den Typ, den Namen und die URL des Bildes. Im Bereich unterhalb dieses Rahmens befindet sich die Darstellung der Webseite, in die das Bild eingebettet ist. Der Rahmen kann zudem geschlossen werden, um auf die normale Ansicht der Webseite zu gelangen.

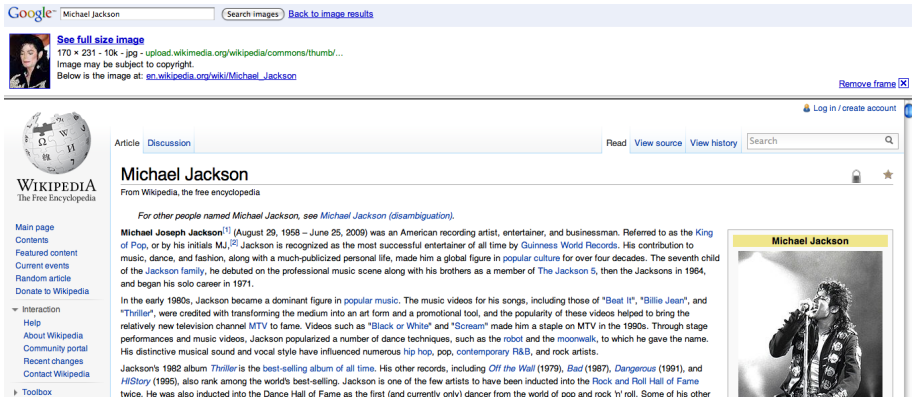

Abbildung 4.3: Weiterführende Darstellung der Google Bildersuche

4.2.3 Einschränkungen in Bezug auf den Versuchsaufbau

Ausgehend von diesen Eigenschaften musste die Google Bildersuche für den Versuchsaufbau angepasst werden. Ziel war dabei, dem Benutzer den Eindruck zu vermitteln, dass er sich weiterhin in der Google Bildersuche befindet, ihm gleichzeitig aber Optionen zu verwehren, die das Experiment behindern würden. Ein Beispiel hierfür sind die weiterführenden Ansichten der Bilder. Der Benutzer kann mit ihnen den Kontext der Bildersuche im Speziellen verlassen und auf eine andere Webseite geführt werden. Das kann zur Folge haben, dass der Proband sich nicht für ein Bild des Basissystems entscheidet, sondern für eines der weiterführenden Ansicht. Um dies zu verhindern, war es erforderlich, alle weiterführenden Links zu entfernen. Auch Links, welche den

Benutzer zu anderen Produkten von Google führen, mussten entfernt werden.

4.3 Das Clipboard

Das Clipboard ähnelt dem Aufbau des Basissystems. Die Abbildung 4.4 gibt eine exemplarische Darstellung des Clipboards.

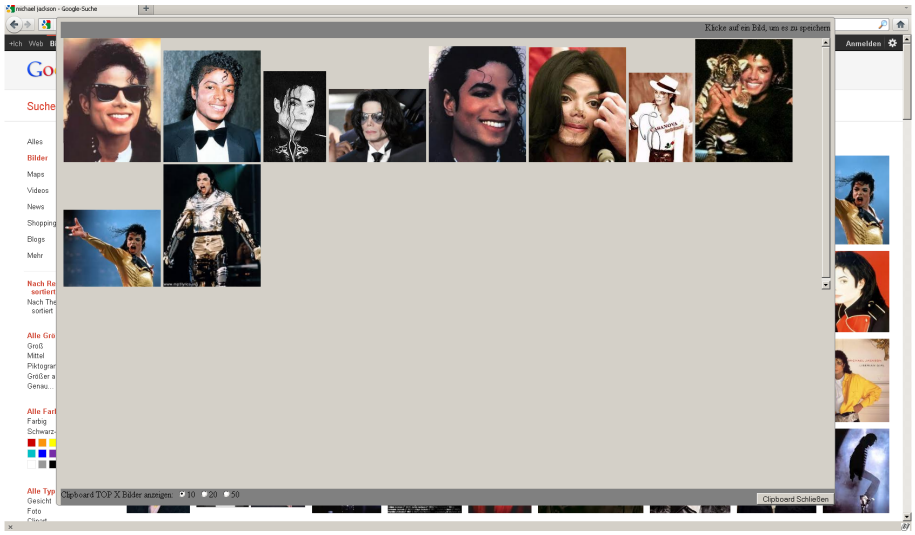

Abbildung 4.4: Clipboard von *EyeVisionSearch*

Den Hauptteil des Clipboards nimmt die Darstellung der Bilder ein, die wie beim Basisystem in Form von Thrumbnails dargestellt sind. Im unteren Bereich des Clipboards befinden sich die Schließen-Schaltfläche sowie die Optionen für die Darstellung verschieden großer Rangfolgen. Klickt man auf eines der Thrumbnails, öffnet sich ein Speichern Dialog, mit dem das angewählte Bild gespeichert werden kann.

Der Benutzer erhält keine Rückmeldung darüber, welche Bilder sich im Clipboard befinden. Erst wenn er das Clipboard öffnet, kann er die enthaltenen Bilder betrachten. Es wurde absichtlich auf eine Markierung der Bilder innerhalb des Basissystems verzichtet, da dies den Benutzer ablenken und den Vorteil der impliziten Informationsgewinnung zunichte machen würde.

4.4 Anforderungen

Die Anforderungen sind nach den jeweiligen Prozessen und Teilprozessen aufgegliedert.

Anforderung 1. Die Evaluation soll vollständig im Mozilla Firefox Browser durchgeführt werden.

Anforderung 2. Das System stellt sicher, dass der Proband gegebenenfalls die vorgesehene Variante des Clipboards für die Durchführung der Aufgabe verwenden kann.

Anforderung 3. Während der Durchführung sollen alle Ereignisse des Eyetrackers, die in Verbindung mit den Bildern der Google Bildersuche stehen, aufgezeichnet werden.

Anforderung 4. Während der Durchführung sollen alle Mausereignisse, die in Verbindung mit den Bildern des Google Bildersuche stehen, aufgezeichnet werden.

Anforderung 5. Das System stellt sicher, dass der Proband alle Fragen der Evaluation beantwortet.

Anforderung 6. Die Daten der Evaluation sollen in einem gängigen Datenformat abgespeichert werden.

Anforderung 7. Das Dateiformat soll für eine Auswertung mit gängigen Statistikprogrammen geeignet sein.

4.4.1 Clipboard

Hier werden die Eigenschaften des Clipboards beschrieben, die es erfüllen muss.

Anforderung 8. Das Clipboard muss eine reduzierte Darstellung der Bilder des Basissystems umsetzen.

Anforderung 9. Das Clipboard soll ein Ranking der Bilder vornehmen.

Anforderung 10. Es soll möglich sein, die Bilder im Clipboard zu speichern.

4.4.2 Basissystem

Die Anforderungen an das Basissystem betreffen vor allem die Änderungen, die am Basissystem vorgenommen werden müssen, aber auch Eigenschaften des Basissystems, die für die Evaluation von Bedeutung sind.

Anforderung 11. Als Basissystem muss die Google Bildersuche in deutscher Sprache verwendet werden.

4.4.3 Eyetracking

Die nachfolgenden Anforderungen beziehen sich auf die Nutzung des Eyetrackers.

Anforderung 12. Es soll eine Anbindung des Eyetrackers an das System durchgeführt werden.

Anforderung 13. Es soll ein Eyetracker der Firma Tobii genutzt werden.

Anforderung 14. Zur Anbindung des Eyetrackers muss das Text 2.0 Framework verwendet werden.

5 Konzeptioneller Entwurf *EyeVisionSearch*

In diesem Kapitel werden die konzeptionellen Abläufe von *EyeVisionSearch* beschrieben. Diese wurden auf Basis der Anforderungsanalyse entwickelt, um die geforderten Eigenschaften des Systems zu erfüllen. Neben den einzelnen Komponenten werden zusätzlich die Datenflüsse und die erstellten Dokumente beschrieben, die für den Ablauf des Systems notwendig sind.

5.1 Architektur von *EyeVisionSearch*

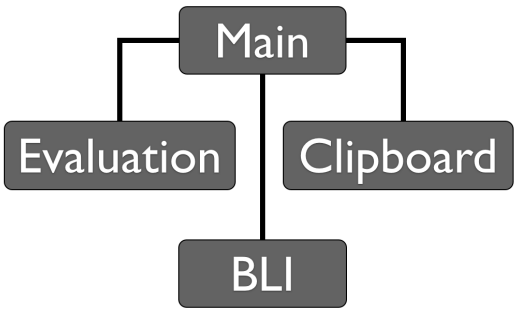

Abbildung 5.1: Darstellung der Komponenten und ihrer Beziehungen

Das System besteht aus den Komponenten *Main*, *Clipboard*, der *Bildersuchen Schnittstelle* (BL-Schnittstelle, BLI) und der *Evaluation* Komponente. Zwischen diesen Komponenten besteht eine ereignisbasierte Kommunikation. Die Ereignisse werden entweder direkt oder indirekt vom Benutzer ausgelöst.

Innerhalb des Systems bewegen sich zwei Nachrichtenströme: der *Evaluations-Datenstrom* und der *Evaluationsobjekt-Datenstrom*. Sie sind teilweise miteinander verbunden, da beide durch dieselben Ereignisse angestoßen werden.

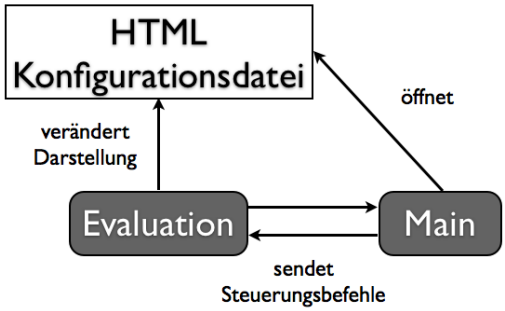

Abbildung 5.2: Informationsfluss des Evaluations-Datenstroms

Der Evaluations-Datenstrom steuert die Evaluation und umfasst die Komponenten Evaluation und Main. Die Komponente Evaluation ist verantwortlich für die Darstellung der Teilprozesse *Beschreibung* und *Fragebogen* der Evaluation. Zusätzlich werden hier alle Ereignisse und Eingaben des Benutzers innerhalb dieser Teilprozesse werden von der Komponente ausgewertet und an die Main Komponente weitergeleitet. Letztere zeichnet die Ereignisse für die spätere Auswertung auf. Der Teilprozess *Aufgabe* wird anderes modelliert. Er wird durch den Benutzer in der Evaluationskomponente ausgelöst, dort aber nicht ausgewertet, sondern direkt an Main weitergeleitet. Hier wird dann der zweite Datenstrom gestartet.

Der Evaluationsobjekt-Datenstrom umfasst die Komponenten Main, die BL-Schnittstelle sowie die Clipboard Komponente. Die BL-Schnittstelle ist hierbei verantwortlich für die Erkennung und Weiterleitung der Ereignisse, die innerhalb des Basissystems ausgelöst werden. Diese werden an die Main Komponente weitergeleitet, die diese aufzeichnet und auswertet. Abhängig von der Evaluationsgruppe werden die Ereignisse gewichtet und an das Clipboard weitergeleitet. Zudem werden einige spezielle Ereignisse, wie gewisse *XHR*-Ereignisse, von der Main Komponente erfasst, die ebenfalls aufgezeichnet und weitergeleitet werden.

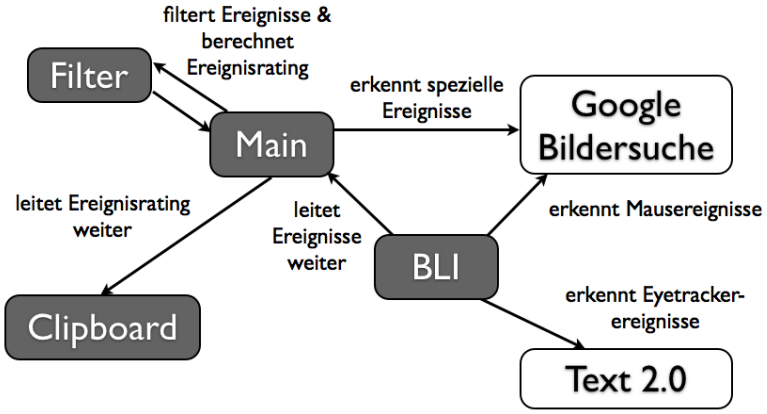

Abbildung 5.3: Evaluationsobjekt-Datenfluss zwischen den Komponenten

5.2 Komponenten

5.2.1 Main

Die Main Komponente bildet das Herzstück des Systems. Sie sorgt für die Weiterleitung der Ereignisse, die von den anderen Komponenten gesendet werden. Zusätzlich werden hier alle Ereignisse auch aufgezeichnet, um sie später für die Auswertung der Evaluation nutzen zu können. Neben den reinen Ereignissen wird weiterhin das Modell des Clipboards innerhalb der Main Komponente gespeichert. Dieses bildet die Grundlage für die Darstellung des Clipboards. Main beinhaltet zudem die Filter, die die erhaltenen Ereignisse von Main und der BL-Schnittstelle, in Abhängigkeit der Versuchsgruppe, für das Clipboard filtern und gewichten. Ausserdem ist Main für verschiedene weitere Aufgaben verantwortlich, beispielsweise für die Erfassung spezieller Ereignisse für den Clipboard-Datenfluss.

5.2.2 BL-Schnittstelle

Die Bildersuchen-Schnittstelle (BL-Schnittstelle, BLI) ist verantwortlich für die Einhaltung spezieller Vorgaben der Evaluation an das Basissystem sowie die Anbindung der notwendigen APIs. Wie bereits oben aufgeführt, wird ausschließ-

lich die Google Bildersuche als Basissystem verwendet. Die konzeptionelle Auslegung der Schnittstelle ist generisch gehalten, damit auch eine andere Bildersuche verwendet werden kann. BLI leitet alle Ereignisse des Basissystems oder Interaktionsereignisse des Nutzers mit dem Basissystem die Main Komponente weiter. Diese Ereignisweiterleitung ist zustandslos, was bedeutet, dass die BL-Schnittstelle alle Ereignisse, unabhängig von Zuständen wie der jeweiligen Versuchsgruppe, an die Main Komponente weiterleitet. Dieses Verhalten ist erwünscht, da alle Daten für die Auswertung der Evaluation genutzt werden. Die BL-Schnittstelle ist verantwortlich für die Anbindung des Eyetrackers und das Auslesen von Informationen. Das geschieht mit Hilfe des Text 2.0 Frameworks. Die erhaltenen Ereignisse werden ungefiltert an die Main Schnittstelle weitergeleitet.

5.2.3 Evaluationskomponente

Die Evaluationskomponente sorgt für den Ablauf und die Steuerung der Evaluation. Sie ist für die korrekte Darstellung der einzelnen Abschnitte der Evaluation und für die Darstellung des Fragebogens verantwortlich. Ereignisse wie der Fortschritt oder die Antworten des Nutzers werden an die Main Komponente weitergeleitet. Alle direkten Eingaben für die Evaluation werden über die Evaluationskomponente durchgeführt, etwa auch konfigurationsbezogene Einstellungen, wie die Versuchsgruppe.

5.2.4 Clipboard

Die letzte Komponente des Systems bildet das Clipboard. Zusätzlich ist es verantwortlich für die Darstellung der Rangfolge der ausgewählten Bilder. Zusätzlich bietet es auch Funktionen zur Anpassung des Inhaltes des Clipboards im Rahmen der gestellten Anforderungen. Schließlich existiert eine Funktion, mit der sich die Bilder innerhalb des Clipboards speichern lassen.

5.3 Dokumente

Das System wird durch mehrere Konfigurationsdokumente konfiguriert. Zum einen besitzt die Main Komponente eine eigene Konfigurationsdatei, da sie für

die Initialisierung der anderen Komponenten verantwortlich ist. Hier werden alle dynamischen Eigenschaften des Systems konfiguriert. Eine einfache und zugleich effiziente Möglichkeit stellt hierbei das *JSON* (JavaScript Object Notation) Format dar. Dieses bietet sich innerhalb einer browserbasierten Evaluation an und ist im Vergleich zu beispielsweise XML leichtgewichtiger.

Eine weitere Konfigurationsdatei ist die Evaluationsdatei, die den Ablauf der Konfiguration deklariert. Hierzu eignet sich die Verwendung von Web-Technologien, beispielsweise HTML für die inhaltliche Deklaration und CSS für die Darstellung. Diese Vorgehensweise hat den Vorteil, dass die Sprachen gut geeignet sind, um innerhalb einer browserbasierten Evaluation Verwendung zu finden.

Nach Abschluss der Evaluation werden die Ergebnisse ebenfalls in einer Ausgabedatei ausgegeben, um später bei der Auswertung der Evaluation einfach auswertbar zu sein. Das verwendete Format hierzu ist auch JSON, welches später durch das Evaluationstool übersetzt werden kann.

5.4 Auswertung

Neben dem Haupteil der Evaluation wird auch die Auswertung der Daten maschinell durchgeführt. Hierzu wurde ein Programm entwickelt, das die verschiedenen JSON-Dateien der jeweiligen Probanden als Eingabe besitzt und diese dann, in Abhängigkeit einer vorher definierten Strategie, in ein gewünschtes Dokument übersetzt. Bei dem Ausgabedokument handelt es sich um ein OpenDocument Spreadsheet, da dieses Format von den meisten Statistikanwendungen verstanden wird und zudem quellenoffen ist.

6 Prototyp

Für die Erstellung des Prototypen wurde das System als Addon, auch Extension genannt, eines Browsers umgesetzt. Ein solches Vorgehen bietet mehrere Vorteile. Zum einen kann so sehr einfach auf dem Basissystem aufgesetzt werden. Außerdem ermöglicht es die Anpassungen und Auswertung der Informationen des Basissystems, ohne dieses direkt zu beeinträchtigen.

Als Browser wurde hierbei der Mozilla Firefox in seiner Version ab 4.0 ausgewählt. Die Wahl fiel auf ihn, weil Extensions hier weitreichende Möglichkeiten haben. Zum Beispiel ist es möglich, gezielt Webseiten auszuwerten und ihre Darstellung zu modifizieren. Zudem bietet die Extension API des Mozilla Firefox die weitreichende Erkennung spezieller, browserbasierter Ereignisse, die für den Lösungsansatz notwendig sind. Außer den genannten Vorteilen existieren Zugriffsmöglichkeiten auf das Dateisystem des Rechners, welches die Erstellung von Ausgabedateien vereinfacht.

Um eine Anbindung eines Eyetrackers an eine Extension zu ermöglichen, wird das Framework Text 2.0 verwendet. Dieses ermöglicht es, Elemente von Webseiten mit Eyetracker-Events zu verknüpfen.

6.1 Mozilla Add-on SDK

EyeVisionSearch wird umgesetzt als Firefox *Extension*, also eine Erweiterung für den Webbrowser von Mozilla. Firefox bietet ab der Version 4.0 eine neue API zur Entwicklung von Extensions mit dem Namen Add-on SDK[1]. Bis zu der Version 3.6 verwendete Mozilla für ihre Produkte ausschließlich *XUL*, die XML User Interface Language. XUL ist eine XML-basierte Sprache zur Entwicklung von plattformübergreifenden Anwendungen. Sie wird von der Mozilla Foundation unterstützt.

[1] `https://addons.mozilla.org/en-US/developers/`

XUL besteht aus drei verschiedenen Dokument-Arten. Der Inhalt, also die verwendeten Oberflächenelemente, werden durch die XUL-Datei, einen XML-Dialekt, definiert. XUL bietet Elemente wie Schaltfächen und Eingabefelder an, um damit graphische Benutzerschnittstellen zu beschreiben. Das Aussehen dieser Oberflächen wird durch CSS-Dateien angepasst. Die Anwendungslogik wird danach beispielsweise durch JavaScript beschrieben. Der Nachteil von XUL ist die hohe Komplexität und der hohe Entwicklungsaufwand. Da XUL in erster Linie nicht für die Entwicklung von reinen Erweiterungen für Firefox gedacht ist, enthält die Sprache eine Vielzahl an Elementen, die nicht genutzt werden. Hierdurch ist der Ressourcenaufwand für Erweiterungen sehr hoch. Gerade im mobilen Anwendungsbereich haben sich hierbei große Probleme ergeben.

Das *Add-on SDK*, früher Jetpack SDK, ist ein neuer Ansatz von Mozilla, um die Addon Entwicklung zu vereinfachen. Es verbindet eine Reihe von JavaScript APIs mit Werkzeugen zur einfachen Erstellung von Addons für diverse Mozilla Produkte. Einige Eigenschaften, die für das weitere Verständnis des Systems notwendig sind, werden nun erläutert. Hierzu gehören Besonderheiten des Add-on SDK sowie einige APIs, die vom System verwendet werden. Das Add-on SDK befindet sich zum Zeitpunkt der Beschreibung in Version 1.0. Allerdings unterliegt es momentan einer hochfrequenten Entwicklung. Somit ist es möglich, dass einige Bestandteile der API verändert oder umgestaltet wurden, sollte der Quellcode des Systems auf einer anderen Versionsnummer des SDK ausgeführt werden. Auch der Ablauf der Befehle kann sich ändern.

6.1.1 Kommandozeile

Das Add-on SDK verwendet Kommandozeilenbefehle, um die verschiedenen Aktionen auszuführen, die für die Erstellung eines neuen Addons notwendig sind.

Listing 6.1: Erstellen des Addon Grundgerüsts unter Linux

```
1 cd <Name des Add-on SDK Ordners>
2 source bin/activate
3 cd <Name des neuen Addon Ordners>
4 cfx init
```

Mit dem Befehl *source* wird die angegebene virtuelle Umgebung geöffnet. Sie enthält den Befehl *cfx*, mit dem alle weiteren Aktionen des Add-on SDK gesteuert werden. Unter Windows wird diese Zeile durch *bin*
activate ersetzt. Danach steht ebenfalls der Befehl cfx zur Verfügung.

cfx init erstellt die initiale Ordnerstruktur des Addons innerhalb des momentanen Ordners. Auch erste Konfigurationen werden durchgeführt.

6.1.2 Ordnerstruktur

Add-on SDK erstellt bei der Initialiserung eines Addons eine Reihe von Ordnern und Dateien. In der nachfolgenden Darstellung werden nur die wichtigen Elemente dargestellt:

<Addon>/

- package.json
- lib/
  - main.js
- data/

Innerhalb des Addon Ordners wird vom Add-on SDK eine Datei mit dem Namen *package.json* angelegt. Sie enthält Konfigurationsinformationen über das Addon, wie beispielsweise Versionsnummer und Autor. Es hat das JSON Format. Der Ordner *lib* enthält alle Module des Addons. Zum Zeitpunkt der Initialisierung befindet sich nur die Datei main.js im Ordner. Sie ist das Hauptmodul, welches ausgeführt wird, wenn das Addon durch den Browser gestartet wird. Im Ordner *data* werden alle Dateien, auf die das Addon während seiner Ausführung zugreift, abgespeichert. Hier werden später Konfigurationsdateien, Bilder und nicht-modulare Skripte abgespeichert.

6.1.3 CommonJS - Modularität in JavaScript

Die einzelnen APIs werden mit Hilfe von *CommonJS* zusammengefügt. Diese bilden einen modularen Aufbau in JavaScript nach, wodurch auf einfache Weise Skripte mehrmals verwendet werden können. CommonJS ermöglicht die Erstellung von so genannten Modulen. Module sind einzelne, wiederverwendbare JavaScript-Skripts. Um Funktionen und Objekte in anderen Modulen

zugänglich zu machen, werden die Funktion *require* und das Objekt *exports* verwendet:

Listing 6.2: CommonJS: require und exports

```
1 // A.js
2 exports.hallo = "hallo";
3
4 // B.js
5 var a_hallo = require("A").hallo;
6 console.log(a_hallo + "welt");
```

Wie im Programmauszug 6.2 zu sehen, wird mit exports der String „hallo“ in Modul A für andere Module zugänglich gemacht. Im Modul B wird nun mit Hilfe der Funktion require das exports-Objekt von Modul A zugänglich. Abschließend kann der String durch den Aufruf der Variable, in dem er gespeichert wurde, verwendet werden. Die Namen der Module sind hierbei identisch mit ihrem Dateinamen. Somit wird Modul A durch die Datei ”A.js” beschrieben. Die Module werden, wie bereits erwähnt, in den Ordner *lib* abgelegt.

6.1.4 Allgemeine Eigenschaften der APIs

Die API von Add-on SDK ist umfangreich. Sie enthält einfache UI-Elemente, aber auch Schnittstellen zur Manipulation von Webseiten bis hin zu systemnahen Werkzeugen, wie I/O Schnittstellen. Darüber hinaus gibt es einige Eigenschaften, die sich durch die gesamte Menge an APIs ziehen. Das gesamte SDK unterstützt die ereignisorientierte Programmierung. Das bedeutet, dass eine Reihe von Objekten Ereignisse sendet, welche von anderen Objekten empfangen und verarbeitet werden können. Diese Eigenschaft wird intensiv in der Umsetzung des Systems genutzt, da die unterschiedlichen Komponenten miteinander kommunizieren müssen. Um die Programmierung zu vereinfachen, kann man in allen Bereichen der Addons das Objekt *console* verwenden. Dieses bietet dem Programmierer eine einfache Möglichkeit, Nachrichten entweder auf die Kommandozeile oder auf die JavaScript-Konsole auszugeben. Es erleichtert die Entwicklung des Addons, da man somit leicht erkennen kann, ob Variablen initialisiert oder Ereignisse richtig weitergeleitet wurden. Innerhalb aller Module ist das Modul *self* benutzbar. Es beschreibt das Addon und wird

vor allem für den Zugriff auf den Data-Ordner gebraucht. Der Data-Ordner enthält zum Beispiel nicht-modulare Skript-Dateien, die innerhalb eine Webseite ausgeführt werden. Auch Bilder, die in Dialogen angezeigt werden sollen, werden hier abgelegt.

6.1.5 Page-Mod API

Eines der wichtigsten Module für die Implementierung von *EyeVisionSearch* ist das Modul *page-mod*. Dieses ermöglicht die Ausführung von so genannten *Content Scripts* innerhalb einer bestimmten Webseite. Content Scripts sind JavaScript-Skripte, die innerhalb bestimmter, definierter Webseiten aufgerufen werden, um diese zu verändern und auszulesen. In *EyeVisionSearch* werden sie genutzt, um den Eyetracker mit der Google Bildersuche zu verbinden. Ausserdem werden einige Funktionen der Bildersuche abgeschaltet.

Ein einfaches Beispiel ist in 6.3 beschrieben.

Listing 6.3: page-mod: einfaches Beispiel

```
1 var pageMod = require("page-mod");
2 pageMod.PageMod({
3   include: "*.org",
4   contentScript: 'window.alert("Page matches ruleset");'
5 });
```

Das Beispiel erläutert, neben der Verwendung des Page-Mod Moduls, wie Objekte im Add-on SDK generiert werden. Viele der APIs von Add-on SDK exportieren so genannte Konstruktoren. Diese Funktionen erstellen Objekte mit Hilfe der ihnen übergebenen Konfigurationsobjekte. Das vereinfacht die Erstellung der Objekte und kaschiert zudem die eigentliche Funktion. Die Page-Mods erhalten mit *include* die Angabe, auf welchen URLs die jeweiligen Skripte ausgeführt werden sollen. Angegeben werden die Skripte entweder direkt als String mit *contentScript* oder als JavaScript-Datei mit *contentScriptFile*. Im gegebenen Beispiel wird das beschriebene Skript, das einen Alert-Dialog öffnet, auf allen Seiten ausgeführt, die als Top-Level Domäne *org* haben. Mit Page-Mods ist es zudem möglich, eine ereignisbasierte Kommunikation zwischen der jeweiligen Seite und dem Addon herzustellen. Hierzu werden so genannnte *Page Worker* verwendet, die eine Kommunikationsschnittstelle zwischen dem

aufrufenden Modul und dem aufgerufenen Modul herstellen.

Listing 6.4: Kommunikation zwischen PageMod und Addon

```
1 var workers = require("page-worker");
2
3 var page = workers.Page({
4   contentURL: "http://en.wikipedia.org/wiki/Internet",
5
6   contentScript:
7       'var elements = document.querySelectorAll("h2 > span");' +
8       'for (var i = 0; i < elements.length; i++)' +
9       ' self.port.emit("title", elements[i].textContent);'
10 });
11
12 page.port.on("title", function (title){
13   console.log(title);
14 });
```

Das Beispiel im Programmauszug 6.4 beschreibt den Aufruf eines Page Workers, der die Titel eines Wikipedia Artikels an das Addon sendet. Mit dem Konstruktor *Page* des Moduls *page-worker* erstellt das Addon den Page Worker. In dem übergebenen Konfigurationsobjekt wird mit *contentURL* die URL angegeben, auf welcher der Page Worker ausgeführt werden soll. Das Skript, welches auf der URL laufen soll, wird mit *contentScript* angegeben und verhält sich ähnlich wie Page-Mod. Besonders wichtig ist im Skript des Page Workers das Objekt *self.port* und das Gegenstück *port* auf Seiten des Addons. Diese stellen die Kommunikation zwischen den jeweiligen Endstellen her. Sie enthalten zwei Funktionen, nämlich *emit* und *on*, die für das Senden bzw. Empfangen der Nachrichten verantwortlich sind. Der Funktion *port.emit* kann eine beliebige Zahl an Parametern übergeben werden, meistens sind es aber zwei. Der erste Parameter benennt das Ereignis mit dem angegebenen String. Die folgenden Parameter müssen zu JSON serialisierbar sein. Diese Daten werden dann von der Endstelle mit Hilfe der Funktion *port.on* empfangen und verarbeitet. An port.on werden zwei Parameter übergeben. Der erste definiert den Namen des Ereignisses. Somit ist ein einfaches Mulitiplexing möglich. Der zweite Parameter ist eine Funktion, die auf den Erhalt der Nachricht reagiert und gegeben-

falls die übergebenen Daten verarbeitet. Die Darstellung 6.1 veranschaulicht diese Kommunikation. Sie ist von der Webseite des Add-on SDK übernommen worden[2].

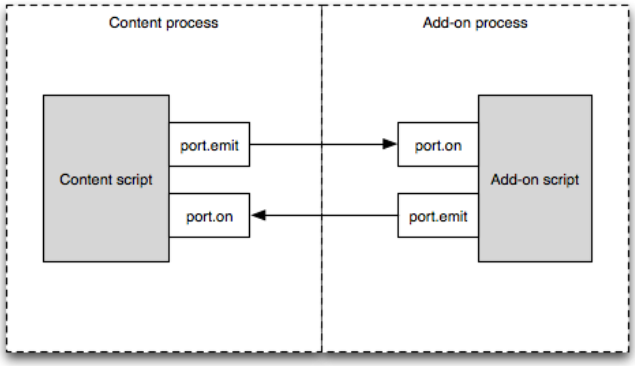

Abbildung 6.1: Schematische Darstellung der Kommunikation über Page Worker (Darstellung von addons.mozilla.org)

6.1.6 Panel - Ein einfacher Dialog

Das Modul Panel bietet eine einfache Möglichkeit, um eigene Dialoge zu erstellen. In *EyeVisionSearch* wird das Panel benutzt, um das Clipboard darzustellen. Der Vorteil hierbei ist, dass es leicht zu schließen ist und den Benutzer nicht bei seiner Ausführung der Aufgaben behindert. Der Inhalt des Panels wird mit Hilfe einer HTML-Datei beschrieben und kann durch CSS grafisch angepasst werden. Zudem ist es möglich, JavaScript-Skripte innerhalb des Panels auszuführen und so weitere Interaktivität zu erhalten. Das Beispiel im Programmauszug 6.5 erläutert die allgemeine Funktionsweise. Das Panel lässt sich leicht schließen, indem man den Fokus auf ein anderes Fenster legt. Somit ist es vor allem für UI Elemente geeignet, die nicht kritisch sind oder den Benutzer nicht behindern sollen.

Listing 6.5: Panel Beispiel

```
1 const data = require("self").data;
```

[2] https://addons.mozilla.org/en-US/developers/docs/sdk/1.0/media/content-scripting-events.png

```
2
3 var panel = require("panel").Panel({
4   // der Content "foo.html" wird aus dem
5   contentURL: data.url("foo.html"),
6   contentScriptFile: data.url("bar.js");
7 });
8
9 // panel anzeigen
10 panel.show()
```

An Panel werden ähnliche Parameter durch das Konfigurationsobjekt übergeben wie an Page-Mod. Es besitzt ebenfalls einen Page Worker, wodurch die Kommunikation zwischen dem Panel und dem Addon realisiert wird.

6.1.7 Widget API

Um dem Benutzer per Mausklick Zugang zum Clipboard zu gewähren, wird ihm ein Widget eingeblendet. Widgets sind bei Firefox kleine Elemente, die innerhalb der Addon Leiste von Firefox angezeigt werden. Sie können verwendet werden, um Mausereignisse zu erkennen oder auf einfache Weise Informationen innerhalb des Toolbars anzuzeigen. Beispielsweise können sie eingesetzt werden, um ein Panel zu öffnen oder JavaScript-Anweisungen auszuführen.

Listing 6.6: Widget

```
1 widgets.Widget({
2   id: "google-link",
3   label: "Widget with an image and a click handler",
4   contentURL: "http://www.google.com/favicon.ico",
5   onClick: function() {
6     require("tabs").activeTab.url = "http://www.google.com/";
7   }
8 });
```

Im Programmauszug 6.6 ist zu erkennen, dass Widgets wie alle bisherigen API Elemente durch einen Konstruktor-Aufruf erzeugt werden. Neben dem bereits bekannten Parameter *contentURL*, der in diesem Fall die angezeigte Abbildung auf dem Widget definiert, enthält das Konstruktionsobjekt noch weitere Parameter. Mit *id* wird der eindeutige Identifier des Widget Elements beschrie-

ben. Mit ihm kann das Widget Element referenziert werden. *label* gibt die Beschriftung des Widgets an, welches über einen Tooltip angezeigt werden kann. Mit der *onClick*-Funktion kann ein Skript an das Widget gebunden werden, welches durch Anklicken des Widgets ausgeführt wird.

6.1.8 Notifications - Erkennung von Browser-Events

Die Verwendung des Moduls *observer-service* ist notwendig, da die Google Bildersuche intensiv XMLHttpRequest verwendet, um Informationen mit dem Web Server auszutauschen. Eine der fortgeschritteneren APIs innerhalb von Add-on SDK ist das Modul *observer-service*. Mit ihm ist es möglich, verschiedene Ereignisse des Browsers zu erkennen und darauf zu reagieren. Diese Ereignisse passieren browserweit und können normalerweise von Addons nicht erfasst werden. Hierzu gehören beispielsweise das Starten und Beenden des Browsers, verschiedene Session Ereignisse oder, wie im gegebenen Fall verwendet, das Senden bzw. das Empfangen eines XmlHttpRequests, also einer JavaScript gesteuerten Anfrage an einen Webserver. Die API enthält eine Funktion mit dem Namen *add*. Diese erlaubt es dem Addon, bestimmte Browserereignisse zu erkennen. Zwei spezielle Ereignisse, die in *EyeVisionSearch* wichtig sind, sind die Observer Notifications *http-on-modify-request* und *http-on-examine-response*. Die erste Observer Notification wird ausgelöst, wenn ein XMLHttpRequest ausgeführt wird. Das Ereignis wird vor dem Senden des Requests ausgeführt, was die Veränderung der Header-Information zulässt. Das Ereignis *http-on-examine-response* wird ausgelöst, wenn ein HTTP Response empfangen wird. Beide Ereignisse übergeben ein *nsIHttpChannel* Objekt. Mit dem Interface können die Header ausgelesen und verändert werden. Das ist wichtig, da das System nur auf einige Ereignisse reagieren muss.

Listing 6.7: Observer Notification Beispiel

```
1 // Empfange HTTP Request
2 obs.add("http-on-examine-response", function(subject) {
3     subject.QueryInterface(Ci.nsIHttpChannel);
4
5     // ist der Request im JSON Format
6     if(subject.contentType.search("json") >= 0)) {
```

```
7           // bearbeite JSON Request
8       }
9   }
```

Wie im Programmauszug 6.7 zu sehen ist, muss das Interface noch an das erhaltene *subject* gebunden werden. Dies geschieht mit der Funktion *QueryInterface* des *nsISupports*, welche das generelle Interface aller XPCOM Interfaces, beispielsweise nsIHttpChannel, ist. Sie ermöglicht die Nutzung des angegebenen Interfaces auf dem ausführenden Objekt. Die verschiedenen Header können nun untersucht werden. Im Beispiel wird der Header Content-Type nach "jsonüntersucht, um festzustellen, ob der Response ein JSON-Dokument enthält.

6.2 Text 2.0

Das Text 2.0 Framework[3] ermöglicht die Nutzung eines Eyetrackers in Java und Webanwendungen. Erstellt wurde es am Deutschen Forschungszentrum für künstliche Intelligenz[4]. Alle Eyetracker der Firma Tobii[5], die die TET API verwenden, können mit Text 2.0 genutzt werden. Das System dieser Diplomarbeit verwendet das Text 2.0 Framework in der Version 1.4 beta 11. Auch dieses Framework unterliegt einer starken Entwicklung. Somit ist es möglich, dass beschriebene Unterprogramme ein anderes Aussehen oder Verhalten aufweisen, sollten sie in einer anderen Version ausgeführt werden.

6.2.1 Komponenten von Text 2.0

Das Text 2.0 Framework besteht aus mehreren Programmen, die Browser und Eyetracker miteinander verbinden. Zuerst muss der Tracking Server gestartet werden. Ist dieser nicht gestartet, interpretieren die weiteren Programme die Mausbewegungen als Blickfixierung und nutzen somit den Eyetracker nicht. Das ermöglicht das Testen von Programmen ohne den Einsatz des Eyetrackers. Abbildung 6.2 zeigt den Text 2.0 Server in der Tray. Das Icon ist als Ampel dar-

[3] http://code.google.com/p/text20/
[4] http://www.dfki.de
[5] http://www.tobii.com

gestellt. Wenn diese grün ist, funktioniert die Verbindung zwischen Eyetracker und Text 2.0.

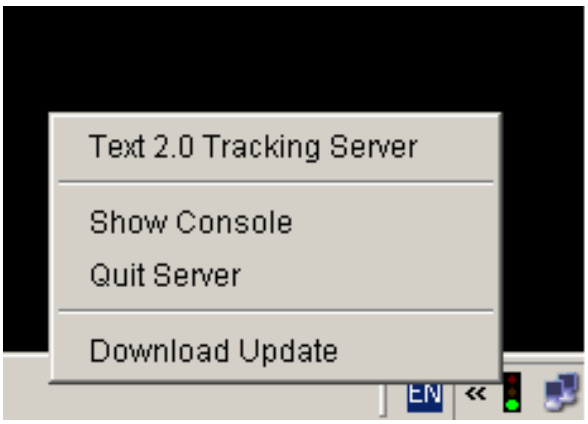

Abbildung 6.2: Screenshot des Text 2.0 Servers

Das zweite Programm, welches benötigt wird, ist das Diagnosetool von Text 2.0. Dieses wird zur Kalibrierung des Eyetrackers sowie für die Überprüfung der Aufnahme- und Verbindungsqualität genutzt.

Die Abbildung 6.3 stellt das Diagnosetool dar. Im oberen Bereich werden die Daten des Tracking Servers graphisch dargestellt. Der linke Bereich stellt die Position der Augen im Verhältnis zum Bildschirm mit Kreisen dar. Je nach Signalqualität sind diese grün, gelb oder rot. Der rosa Punkt gibt die Position der Blickfixierung des Benutzers an. Sakkaden werden durch einen rosa Strich dargestellt. Die nächste Darstellung ist die Entfernung der Positionen der beiden Augen vom Bildschirm beziehungszweise vom Eyetracker. Bei beiden Anzeigen stellt der rote Rahmen den optimalen Bereich dar, in denen sich die Augen des Probanden befinden sollten. Die Anzeige rechts stellt die Pupillengröße des Probanden mit türkisen Balken dar.

Der untere Bereich liefert weitere Informationen über die Konfiguration des Tracking Servers sowie die Kalibrierungsmöglichkeiten. Soweit nicht anders vorgesehen, wird der Eyetracker mit Hilfe des *"Perform hardware recalibration"* Eintrags kalibriert. Bei der Kalibrierung wird der Proband aufgefordert, sich Punkte auf dem Bildschirm anzusehen. Diese werden durch einen blauen Kreis markiert, dessen Mittelpunkt schwarz ist. Der Kreis wechselt zwischen

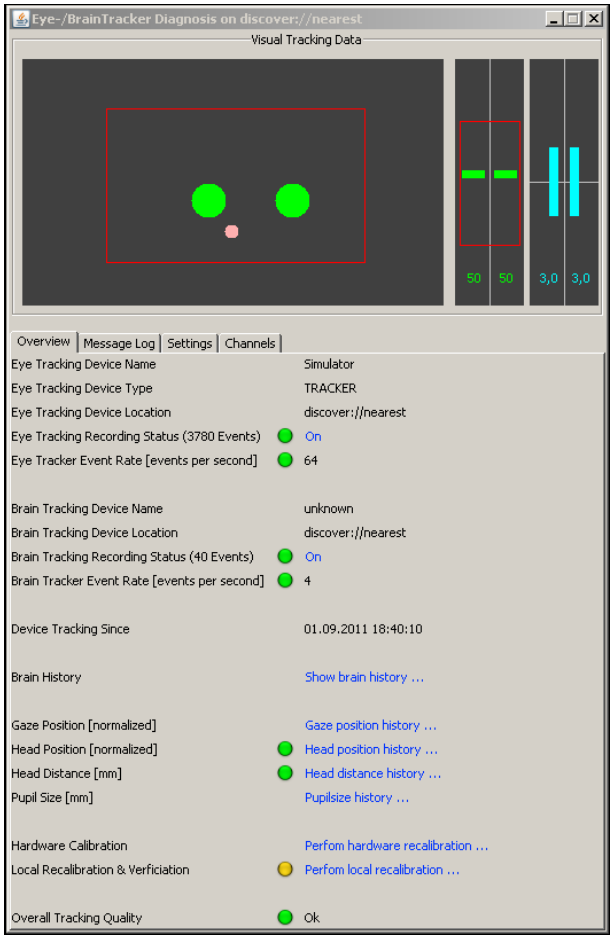

Abbildung 6.3: Screenshot des Text 2.0 Diagnose Programms

den verschiedenen Kalibrierungspunkten und verweilt dann auf diesen. Zeitgleich wird der blaue Kreis verkleinert, um so dem Probanden zu signalisieren, dass er den Mittelpunkt nun fixieren soll. Nach etwa sechs bis sieben Fixierungen von Kalibrierungspunkten wird die Kalibrierung beendet. Gegebenenfalls wiederholt das System ungenaue Kalibrierungspunkte. Nach Abschluss der Kalibrierung kann das System nun verwendet werden. Natürlich ist die Kalibrierung nicht bei jedem Probanden Pflicht. Allerdings werden so Probleme im weiteren Verlauf vermieden.

6.2.2 Kommunikation zwischen Eyetracker und Webseiten

Die Kommunikation zwischen dem vorhandenen Eyetracker und einer Webseite wird durch die Einbindung des Text 2.0 Applets in den DOM-Baum der Webseite bewirkt. Das wird durch die Einbindung des JavaScript-Skripts *text20.js* ermöglicht. Text 2.0 verwendet *jQuery*, eine JavaScript Bibliothek, sowie die Erweiterung *jQuery Timers*.

Listing 6.8: Einbinden von Text 2.0 in eine Webseite

```
1 <script type="text/javascript" src="jquery.js"></script>
2 <script type="text/javascript" src="jquery.timers.js"></script>
3 <script type="text/javascript" src="text20.js"></script>
4
5 <script type="text/javascript">
6     text20.core.init()
7 </script>
8
9 <body><div onFixation="alert('Hello again')">Hello World</div></body>
```

In der Programmauszug 6.8 ist erkennbar, wie Text 2.0 verwendet wird. Mit der Funktion *text20.core.init()* wird das Text 2.0 Applet in die Webseite eingebunden. Es dauert eine gewisse Zeit, bis die Verbindung zwischen dem Applet und dem Eyetracker hergestellt ist, generell etwa zwischen 5 und 10 Sekunden. Ist die Verbindung hergestellt, sendet das Applet in einem fest vorgegebenen Intervall die Koordinaten des Fixationspunkts des Benutzers über das Text 2.0 JavaScript-Skript an die Webseite. Innerhalb des Skripts werden die Koordinaten in die jeweiligen Elemente übersetzt, die sich an der Koordinate befinden. Werden zutreffende Elemente gefunden, wird bei diesen überprüft, ob bei ihnen eines der zulässigen Attribute von Text 2.0 vorhanden ist. Die zulässigen Attribute sind hierbei *onFixation*, *onGazeOver* und *onGazeOut*. *onFixation* wird ausgeführt, wenn das Element mindestens seit der letzten Übermittlung fixiert wird. *onGazeOver* und *onGazeOut* sind vergleichbar mit den jeweiligen ähnlich benannten Mausereignissen und werden ausgelöst, wenn der Benutzer anfängt oder aufhört, das Element anzusehen. Wird ein passendes Attribut erkannt, wird der Wert des Attributs per *eval()* ausgeführt.

Neben der Umsetzung von Eyetracker Ereignissen durch Attribute kann über einen Listener der Zustand des Eyetrackers ermittelt werden. Die Funktion hierfür heisst *text20.connector.listener*. Der Funktion werden zwei Parameter übergeben. Der erste Parameter beschreibt den Namen des gewünschten Ereignisses, der zweite die Callback-Funktion, die aufgerufen wird, wenn das Ereignis ausgelöst wird. Zwei wichtige Ereignisse sind *INITIALIZE* und *fixation*. Die Funktion *INITIALIZE* wird ausgelöst, wenn die Initialisierung des Applets abgeschlossen ist und der Eyetracker Informationen liefern kann. Dieses Ereignis ist äusserst nützlich, da die Initialisierung aufgrund der langen Initialisierung asynchron verläuft. Die Funktion *fixation* wird ausgelöst, wenn eine Fixierung innerhalb der Webseite erkannt wird. Das Ereignis enthält verschiedene Informationen über Position, Art und Dauer der Fixierung.

6.3 Implementierung der Komponenten

Nachdem die unterstützenden APIs beschrieben sind, werden die einzelnen Implementierungen der Komponenten besprochen, wie sie im Kapitel 5 als allgemeines Konzept für den Aufbau des Systems vorgestellt wurden. Alle Komponenten teilen sich den gemeinsamen Namensraum *evs*. Komponenten, die als Content Script ausgeführt werden, sind allerdings nicht im selben Namensraum wie Addon Komponenten, sondern erstellen ihren eigenen Namensraum mit demselben Namen. Innerhalb der Content Scripts können aber mehrere Komponenten im selben Namensraum aufzufinden sein.

6.3.1 Main

Main stellt die Hauptkomponente des Systems dar. Hier werden die weiteren Komponenten initialisiert. Zudem sorgt Main für die Sicherung der Ereignisse. Auch die Formatierung der eingehenden Ereignisse aus der BLI Komponente zu Clipboard Ereignissen findet innerhalb von Main statt.

6.3.2 Initialisierung

Die Initialisierung von Main beginnt mit dem Laden der Konfigurationsdatei *data/config.json*. Sie wird mit Hilfe von AJAX dem System zugänglich gemacht

und wird mit der Hilfsklasse *evs.Config* ermöglicht. Sie lädt per XHR die JSON Serialisierung und übersetzt sie mit *JSON.parse* in ein Objekt, das umgesetzt und an Main zurückgegeben wird. Anhand der Konfigurationswerte erstellt Main nun die anderen Komponenten. Da diese als Content Scripts implementiert sind, werden sie über das Modul page-mod geladen. Sie bilden das Kommunikationsgegenstück zu den eigentlichen Content Scripts. Alle eingehenden Page Worker Ereignisse laufen zentral über Main und werden mit Hilfe der Liste *Main.handlers* behandelt. Dies vereinfacht das Finden der zugehörigen Funktionen, da sie zentral in Main abgelegt sind.

Listing 6.9: Main.handlers Beispiel

```
1 // event handlers in Main
2 this.handlers = {
3   // BLI events
4   bli: function(data) {
5     config.storage.add(data);
6
7     if(config.group) {
8       config.clipboard.send("data",
9         config.filter.filter(data)
10      );
11  }},
12 // ...
13 }
14 // pageMod in BLI
15 BLI.pageMod = require("page-mod").PageMod({
16 // ...
17   onAttach: function(worker) {
18     // ...
19     worker.port.on("bli", function(data) {
20       main.handlers.bsi(data);
21     };
22   }
23 // ...
24 });
```

Wie in dem Programmauszug 6.9 zu sehen, werden alle Nachrichten des BLI Content Scripts über den Port ”bli” geleitet. Die Informationen in *data* sind

standardisiert und werden von der angegebenen Funktion in *Main.handlers* verarbeitet. Die Funktion *config.filter.filter(data)* hat hierbei eine besondere Rolle. Mit ihr, genauer der Funktion *evs.Filter*, werden die einkommenden Ereignisse des BLI, je nach Gruppe des Probanden, gefiltert. Filtern heisst hierbei, dass die Ereignisse in neue Ereignisse für das Clipboard übersetzt werden. Zusätzlich zum Page-Mod müssen bei der Komponente BLI noch die jeweiligen observer notifications registriert werden. Google kommuniziert, wie bereits erwähnt, sehr intensiv über XHR und lädt so zum Beispiel zusätzliche Filter oder neue Seiten nach. Diese Ereignisse müssen erkannt und an BLI weitergeleitet werden, da BLI verschiedene Listener bei den neu geladenen Elementen registrieren muss. Mit Hilfe des Quellcodes in dem Programmauszug 6.10 werden die beiden Ereignisse erkannt. Da die Such-API von Google *RESTful* ist, kann man die Requests und Responses anhand ihrer URI unterscheiden.

Listing 6.10: Observer Notifications

```
1  obs.add("http-on-examine-response", function(subject) {
2    subject.QueryInterface(Ci.nsIHttpChannel);
3
4    if(subject.originalURI.host.search("images.google.de") >= 0) {
5              if(subject.contentType.search("javascript") >= 0) {
6        if(subject.originalURI.path.search("imgevent") >= 0) {
7          config.bli.send("imgevent", null);
8        }
9      }
10     else if(subject.contentType.search("json") >= 0) {
11       if(subject.originalURI.path.indexOf("search") >= 0) {
12         config.bli.send("reload",null);
13       }
14     }
15   }
16 });
```

Im Beispiel wird ein neuer Notification Listener für das Ereignis "http-on-examine-response" registriert. Es wird ausgelöst, wenn ein XMLHttpRequest empfangen und ausgewertet wird. Zuerst wird das Interface *nsIHttpChannel* für das Event-Objekt registriert. Dieses ist notwendig, um die Http Header auszulesen. Danach wird überprüft, ob die empfangene Antwort von Google

stammt. Zwei Antworten von Google sind besonders hervorzuheben: *images.google.de/imgevent* und *images.google.de/search*. */imgevent* werden gesendet, wenn der Benutzer eine neue Seite von Bildern anfragt, beispielsweise durch Herunterscrollen. Der Inhalt ist ein Javascript, welches die neuen Bilder über XHR nachlädt. Die zweite Antwort, */search*, wird gesendet, wenn der Benutzer einen Filter verwendet oder eine neue Anfrage an Google stellt. Der Inhalt der Antwort ist hierbei eine JSON Serialisierung der neuen Bilder. Diese beiden Anfragen sind wichtig, da so festgestellt werden kann, ob neue Bilder dem Benutzer präsentiert werden. Sind alle Komponenten initialisiert, erzeugt Main ein neues Tab-Fenster, in dem die HTML-Konfigurationsdatei für die Evaluations-Komponente geladen wird. Dies stößt die Evaluation an und versetzt Main in einen reaktiven Zustand, da nun die Steuerung des Systems durch die Evaluations-Komponente und den Benutzer übernommen wird.

6.3.3 Speicherung der Kommunikation

Main ist zuständig für die Speicherung der verschiedenen Nachrichten, die zwischen den Komponenten gesendet werden. Die Speicherung wird später zusätzlich genutzt, um die Auswertung der Evaluation durchzuführen. *evs.Storage* speichert eine Sequenz von Ereignissen ab, die durch die Funktion *add* erweitert werden kann. Mit *save* kann nun die Liste als JSON Serialiserung im Dateisystem gespeichert werden. Hierzu ist die Hilfsfunktion *evs.io.saveData* verantwortlich. Sie wird im nächsten Abschnitt beschrieben.

6.3.4 Hilfsfunktionen

Eine weitere wichtige Aufgabe von Main ist die Bereitstellung verschiedener Hilfsfunktionen, da Main die einzige Komponente ist, die einen tiefergehenden Zugang zu den APIs von Add-on SDK hat. Wie im Abschnitt 6.3.3 erwähnt, werden in *evs.io* mehrere Funktionen für den Zugriff auf das Dateisystem zur Verfügung gestellt. Die Methoden sind *evs.io.saveData* und *evs.io.saveImage*. *evs.io.saveData* speichert einen Text direkt auf dem Desktop unter dem angegebenen Dateinamen ab. Diese Funktion wird verwendet, um die JSON Serialisierung von *evs.Storage* zu speichern. In der folgenden Darstellung wird das Programm im Einzelnen besprochen. Der Quellcode ist hierbei aus dem Mozilla

Developer Center, kurz MDC, entnommen[6].

Listing 6.11: evs.io.saveData Quellcode

```
1  evs.io.saveData = function(fileName, data) {
2    let {Ci, Cc, Cu, components} = require("chrome");
3
4    try {
5      let NetUtil = Cu.import("resource://gre/modules/NetUtil.jsm").NetUtil;
6      let FileUtils = Cu.import("resource://gre/modules/FileUtils.jsm").FileUtils;
7
8      // saves file to desktop
9      let file = FileUtils.getFile("Desk", [fileName]);
10
11     let ostream = FileUtils.openSafeFileOutputStream(file)
12
13     let converter = Cc["@mozilla.org/intl/scriptableunicodeconverter"].
14       createInstance(Ci.nsIScriptableUnicodeConverter);
15       converter.charset = "UTF-8";
16
17     let istream = converter.convertToInputStream(data);
18
19     // The last argument (the callback) is optional.
20     NetUtil.asyncCopy(istream, ostream, function(status) {
21       if (!components.isSuccessCode(status)) {
22         throw "data could not be saved to file " + fileName;
23       }
24
25     });
26   } catch(e) {
27     console.log(e);
28     throw e;
29   }
30 };
```

Zuerst werden die notwendigen Add-on SDK Module für den Zugang auf das Dateisystem geladen. Danach werden die so genannten *JavaScript Modu-*

[6] https://developer.mozilla.org/en/Code_snippets/File_I\%2F\%2FO#Creating_an_.0AnsIFile.0A_object_(.22opening.22_files)

le, kurz JSM, geladen. Diese Art von Modulen steht Plug-in Programmierern schon in XUL zur Verfügung und ist in etwa vergleichbar mit den Modulen des Add-on SDK. Die zwei hier verwendeten sind *FileUtils.jsm*[7] und *NetUtils.jsm*[8]. *FileUtils.jsm* wird verwendet, um Dateien und Verzeichnisse auf dem lokalen System zu lesen und zu schreiben. *NetUtils.jsm* ist eine API für die einfache Nutzung von netzwerkbezogenen Aufgaben. In diesem Fall wird damit die asynchrone Speicherung der Datei überwacht. Mit der Funkton *FileUtils.getFile* wird eine Datei erstellt oder geöffnet, falls sie schon existiert. Der erste Parameter ist hierbei eine String-basierte Konstante für ein spezielles Verzeichnis innerhalb des Systems. In diesem Fall ist der Parameter "Desk", der den Desktop des Benutzers referenziert. Der zweite Parameter enthält einen Array von Dateinamen, in diesem Fall nur einen. Danach wird ein Unicode Converter geladen, mit dem die Daten in UTF-8 formatiert werden. Das erhöht die Lesbarkeit der gesammelten Daten, zumal *EyeVisionSearch* im deutschsprachigen Raum eingesetzt wird. Zuletzt werden per *NetUtil.asyncCopy* die gegebenen Daten in die gegebene Datei kopiert. Sollte dieser Vorgang fehlerhaft sein, wird eine Ausnahmebehandlung angestoßen. Eine ähnliche, wenn auch kompliziertere Hilfsfunktion ist *evs.io.saveImage*, mit der ein gegebenes Bild im Dateisystem gespeichert wird. Hierbei hat der Benutzer die Auswahl, wo die Datei gespeichert werden soll. Dies wird durch die Verwendung eines Dialog Fensters ermöglicht. Die Funktion besteht aus drei Teilen, *openSaveDialog*, *loadImage* und *saveTo*. *openSaveDialog* erstellt einen „Speichern unter…“ Dialog mit Hilfe einer Instanz des Interfaces *nsIFilePicker*. nsIFilePicker erstellt einen “Speichern unter..."Dialog in Gestalt des verwendeten Betriebssystems. Mit der Funktion *appendFilters* können die Typen definiert werden, welche das Dialog Fenster anzeigen soll. Mit *init* wird das Fenster initialisiert, unter Zuhilfenahme der gegebenen Parameter. Der erste Parameter definiert das window Element, in dem der Dialog dargestellt werden soll. Der zweite Parameter gibt den Titel des Dialog-Fensters an. Als letzter Parameter wird der Modus des Dialogs definiert, *nsiFilePicker.modeSave* entspricht hierbei der “Speichern unter…” Darstellung. Der Dialog kann nun mit *show* angezeigt werden. Wird der Dialog daraufhin geschlossen, wird eine Status Variable zurückgegeben.

[7] `https://developer.mozilla.org/en/JavaScript_code_modules/FileUtils.jsm`
[8] `https://developer.mozilla.org/en/JavaScript_code_modules/NetUtil.jsm`

Ist diese *nsIFilePicker.returnOK*, wurde der Dialog per OK beendet. Der ausgewählte Dateiname ist unter *nsIFilePicker.file.path* zu finden. Das Bild wird per XMLHttpRequest geladen. Hierzu wird die Funktion *loadImage* verwendet.Vor dem Versenden der Anfrage muss der Mime-Typ auf "text/plain; charset=x-user-defined" gesetzt werden, da sonst versucht wird, die Bilddatei als XML Dokument zu interpretieren. Im Nachfolgenden wird der Content-Type des Bildes ermittelt, indem der Header des XMLHttpRequest ausgelesen wird. Danach wird das Bild in der ausgewählten Datei gespeichert. Das funktioniert anders als bei der *evs.io.saveData* Funktion. In diesem Fall wird kein Converter verwendet, sondern das Bild direkt als *nsILocalFile* erstellt und dann in diese Datei gestreamt. Sollte dies gelingen, wird noch ein Ereignis im Storage gespeichert und ein Alert an den Benutzer gesendet, um diesem den Speichervorgang zu bestätigen.

6.3.5 Kommunikation vom Eyetracker zum Clipboard

Ein entscheidender Punkt innerhalb des Systems ist die Weiterleitung der Eyetracker-Informationen bis hin zum Clipboard. Als Veranschaulichung dient hierbei Abbildung 5.3. Durch die Umsetzung der BLI Schnittstelle als Content Script wird dafür gesorgt, dass es beim Aufruf der Google Bildersuche geladen wird. Die erste Aufgabe besteht nun darin, die Google Bildersuche auf die gewünschten Eigenschaften zu reduzieren, um beispielsweise die *Weiterführenden Ansichten* sowie weitere Hyperlinks zu entfernen. Zudem wird eine Liste der bisher geladenen Bilder samt Reihenfolge an die Main Komponente weitergeleitet. Danach wird das Text 2.0 Framework in die Google Bildersuche eingebunden. Anzumerken ist hierbei, dass das Text 2.0 Framework von sich aus annimmt, dass das Text 2.0 Applet in der Umgebung angewendet wird, von der es auch geladen wurde. Das ist allerdings in diesem Kontext nicht der Fall. Deshalb musste das Framework für *EyeVisionSearch* um die Information des Class Path ergänzt werden. Im Programmauszug 6.12 ist zu sehen, wie die Übergabe dieser Information funktioniert.

Listing 6.12: Erweiterung der Text 2.0 API

```
1 text20.connector.config.archive = jar;
2
```

```
3 // API erweitert um den Class Path der Aufgabe
4 text20.connector.config.classPath = classpath;
5 ...
6 text20.core.init();
```

Somit kann das Text 2.0 Framework auch außerhalb der Position des Applets verwendet werden. Bevor der Eyetracker initialisiert ist, wird ein Overlay über die Darstellung der gefundenen Ergebnisse gelegt. Das soll verhindern, dass der Proband sich die Bilder ansieht und eine Entscheidung frühzeitig trifft, bevor der Eyetracker die Augenbewegungen erkennen kann. Wie im Abschnitt 6.2.2 erwähnt, teilt Text 2.0 über den Listener *INITIALIZE* den angebundenen Callback-Funktionen mit, dass der Eyetracker nun verwendet werden kann. Sollte dieses Ereignis ausgelöst werden, wird das Overlay entfernt und die Bilder des Suchergebnisses mit den Attributen *onGazeOver* und *onGazeOut* belegt. Zusätzlich werden vom System Ereignisse für die Mausinteraktion des Benutzers mit den Bildern aufgezeichnet, speziell das Betreten (*onMouseOver*), das Verlassen (*onMouseOut*) und das Verweilen (*hover*). Das Ereignis *hover* wird verwendet, da Google die vergrößerte Ansicht des Bildes nach einer kurzen Verweildauer auf einem Bild anzeigt und damit diese Darstellung registriert werden kann.

Sollte ein Ereignis ausgelöst werden, wird eine Nachricht von der BLI Schnittstelle an die Main Komponente gesendet mit folgenden Informationen:

- time: Zeitstempel (Millisekunden seit 1. Januar 1970)
- type: Art des Ereignisses (gazeover, gazeout, mouseover...)
- id: URI des Ergebnisobjekts
- : misc: weitere Informationen

Innerhalb der Main Komponente werden nun die Nachricht, wie in Programmauszug 6.9 beschrieben, zuerst gespeichert und danach die Information an das Clipboard weitergeleitet. Zuvor wird die Nachricht aber durch einen Filter in eine *Ranking* Nachricht umgesetzt, da das Clipboard nur die momentane Reihenfolge der Bilder speichert. Die eigentlichen Ereignisse werden innerhalb der Main Komponente gespeichert. Somit wird eine klare Trennung von Darstellung und Modell erreicht. Welcher Filter auf den Nachrichten angewendet

wird, wird durch die gewählte Gruppe des Probanden bestimmt, da jede Gruppe andere Auswahlkriterien besitzt. Die Ranking Informationen, die an das Clipboard weitergeleitet werden, enthalten die URI des Bildes als Identifikator und den Wert, um den das Ranking steigen oder sinken soll. Innerhalb vom Clipboard werden die Rankingwerte zusammengerechnet und beim Aufruf in absteigender Reihenfolge dargestellt.

6.3.6 Der Evaluationswizard

Im Folgenden wird der Evaluationswizard beschrieben, der den Probanden durch die Evaluation leitet. Er setzt den Evaluationsdatenfluss um, der durch Abbildung 5.2 veranschaulicht wird. Nach der Initialisierung der einzelnen Komponenten öffnet die Main Komponente die *Konfigurationsdatei* der Evaluation im Browser. Die Konfigurationsdatei ist ein HTML Dokument mit einem definierten JavaScript Anteil. Sie besteht aus einer Serie von DIV-Elementen der Klasse *section* mit eindeutigen IDs. Diese DIV-Elemente werden *Sektionen* genannt. Innerhalb der Sektionen sind alle HTML Elemente erlaubt. Auch die Darstellung kann durch CSS frei angepasst werden. Zusätzlich wird ein spezielles Anchor-Element mit der Klasse *link-next* vorgegeben, mit dem die Evaluationskomponente angewiesen wird, dem Probanden die nächste Sektion zu präsentieren. Dieses Element ist notwendig, damit der Proband mit der Evaluation fortfahren kann, sofern keine andere Regel definiert ist.

Neben dem Inhalt und der Darstellung der einzelnen Sektionen definiert die Konfigurationsdatei zudem die Gruppen und die Reihenfolge, in der sie dem Probanden einer Gruppe präsentiert werden. Hierbei handelt es sich um ein JavaScript Objekt mit dem Namen *evs_sequences*. Dieses Objekt enthält Listen von Selektoren. Die Schlüssel für den Zugriff auf diese Listen sind die jeweiligen Gruppennamen. Bei den Selektoren gibt es zwei Spezialisierungen. Entweder handelt es sich um einen CSS Selektor für die jeweilige Sektion über die eindeutige ID oder um das Steuerzeichen *$*, welches die Evaluation anweist, ein neues Fenster mit dem gegebenen Basissystem aufzurufen. Das wird erreicht, indem die Evaluationskomponente eine Nachricht an die Main-Komponente sendet. Die Main Komponente ruft daraufhin das Basissystem in einem neuen Fenster auf.

Zudem wird eine Sonderregel für das Fortfahren der Evaluation definiert. Die Evaluation fährt nur dann fort, wenn der Proband das neu erstellte Fenster schließt. Wird das Fenster geschlossen, sendet die Main-Komponente eine Nachricht an die Evaluationskomponente, damit diese fortfahren kann. Dieses Vorgehen vereinfacht die Steuerung der Evaluation und sorgt einerseits für eine klare Trennung der Evaluationsanzeige und der Ausführung der Aufgabe im Basissystem, andererseits ist der Zustand der Evaluation immer konsistent mit der Aufgabenstellung. Ausserdem erlaubt es dem Probanden, die Aufgabenstellung noch einmal zu lesen.

Die Evaluation beginnt, indem die Konfigurationsdatei geladen wird. Zuerst wird eine Auswahl der Gruppen präsentiert, die als Schlüssel innerhalb des *evs_sequences* Objekts definiert sind. Wird eine Gruppe ausgewählt, beginnt die Evaluationskomponente mit der sequenziellen Darstellung der einzelnen Sektionen. Unter normalen Umständen wird das Fortfahren mit der Evaluation durch einen Link der Klasse *link-next* gesteuert. Wird dieser angewählt, stellt die Evaluationskomponente entweder die nachfolgende Sektion dar oder sendet eine Nachricht an die Main-Komponente. Das Verhalten ist abhängig vom nächsten Element in der Gruppenliste des evs_sequences-Objekts . Zudem werden alle vorhandenen Eingabefelder, Auswahllisten und Textelemente analysiert.

Ein weiteres Eingabe-Objekt, das innerhalb der Evaluation zur Verfügung steht, ist das *Slider* Element. Das Slider Element wird mit Hilfe der JavaScript Bibliothek *jQuery UI*[9] erstellt. Der Slider ist ein DIV-Element der Klasse *slider*. Es handelt sich um eine Balkendarstellung, auf dem ein so genannter Handler positioniert werden kann. Der Handler ist zu Beginn unsichtbar, um dem Probanden keine Standardeinstellung zu präsentieren. Erst nachdem er eine Position auf dem Slider selbst angeklickt hat, wird der Handler sichtbar. Mit dem Slider ist es möglich, dem Probanden das Gefühl zu vermitteln, sich innerhalb eines Intervalls für einen Wert entscheiden zu können. Er bietet weitreichendere Einstellungsmöglichkeiten als beispielsweise eine Liste Optionen. Ausserdem ist es aus statistischer Sicht notwendig, dass die ermittelten Werte kontinuierlich sind, da sonst die verwendeten statistischen Tests nicht

[9] `http://jqueryui.com`

angewendet werden können.

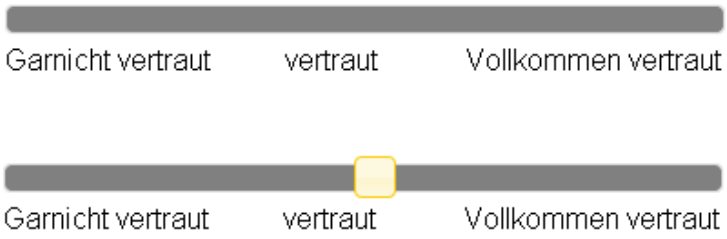

Abbildung 6.4: Slider, oben: Initialzustand, keine Darstellung des Handler; unten: Handler Darstellung nach Interaktion mit Benutzer

Die Evaluationskomponente überprüft vor dem Fortfahren der Evaluation, ob alle Eingabefelder ausgefüllt sind. Das soll sicherstellen, dass der Proband alle Fragen beantwortet hat. Nicht bearbeitete Eingabefelder werden farblich markiert, damit der Proband weiß, wo er noch Daten eingeben muss. Sind alle Daten eingegeben, werden sie ausgewertet. Die Auswertung der Informationen wird dann an die Main-Komponente gesendet. Danach wird das nächste Sektionselement ermittelt und angezeigt.

6.3.7 Clipboard Darstellung und Kommunikation

Das Clipboard dient der Darstellung der ausgewählten Bilder des Systems. Es wird durch ein Panel des Add-on SDK realisiert und entspricht der Darstellung im Abschnitt 4.4. Es kann aufgerufen werden, wenn der Benutzer das *EyeVisionSearch*-Widget anklickt oder die vorgesehene Tastenkombination drückt. Das Clipboard ist sehr groß dimensioniert, um zu verhindern, dass der Proband die Bilder des Clipboard mit den Bildern der Bildersuche im Hintergrund vergleicht. Gleichzeitig lässt es sich leicht schließen, entweder durch Betätigung der Schließen Schaltfläche oder duch einen Klick ausserhalb des Panels.

Innerhalb des Clipboards werden dem Probanden die Bilder in der ermittelten Reihenfolge präsentiert. Hierbei wird eine ähnliche Thumbnail-Darstellung verwendet, wie sie der Proband von der Google Bildersuche kennt. Durch

Anklicken der Bilder wird der Speichervorgang des Bildes ausgelöst, der bereits im Abschnitt 6.3.4 besprochen wurde. Unterhalb der Thumbnail-Darstellung befindet sich eine Optionsliste, mit deren Hilfe der Benutzer die Anzahl der Bilderauswahl bestimmen kann. Voreingestellt sind die Top 10 Bilder. Das Clipboard erhält während der Evaluation die vorher gefilterten Nachrichten der BLI-Schnittstelle über die Main-Komponente. Diese werden in einem Objekt abgespeichert. Die Nachrichten enthalten die URI des Bildes sowie ein Rating, welches von der *evs.Filter* Funktion errechnet wurde. Dieser Wert wird, falls die URI bereits gespeichert wurde, auf das bestehende Rating addiert. Wird nun das Clipboard dargestellt, werden die Bilder nach absteigendem Rating sortiert präsentiert.

6.4 Auswertungstool

Das Auswertungstool *tool-evaluation* wurde für die Diplomarbeit konzipiert, um die JSON-Dokumente mit den aufgezeichneten Ereignissen in ein OpenDocument Spreadsheet zu übersetzen. Das Programm ist in Java geschrieben und benutzt *JSON in Java*[10] für das Parsen der JSON-Dokumente. Zur Erstellung des OpenDocument Spreadsheets wird die Bibliothek *JOpenDocument*[11] verwendet.

6.4.1 JSON in Java

JSON in Java ist eine API, mit der man das JSON data interchange format in Java verwenden kann. Es ist eine sehr leichtgewichtige API, die sich einfach in verschiedene Projekte einbinden lässt.

Der Quellcode in 6.13 dient als kurze Einführung in die Funktionsweise der API.

Listing 6.13: JSON in Java

```
1 // JSON-Datei output.json
2 File f = new File("output.json");
3
```

[10] http://www.json.org/java/index.html
[11] http://jopendocument.org/

```
4 // JSON Objekt aus der Datei parsen
5 JSONObject obj = new JSONObject(
6   new JSONTokener(new FileReader(f))
7 );
8
9 // alle Eigenschaftsnamen extrahieren
10 String[] name = JSONObject.getNames(obj);
11
12 // alle Key-Value Paare auflisten
13 for(String key : names) {
14   log.info(key + " = " + obj.get(key).toString());
15 }
```

JSON in Java enthält die Klasse *JSONTokener*, mit der man mehrere Formen von JSON-Repäsentationen parsen kann. Für den gegebenen Anwendungsfall eignet sich die Übergabe eines Readers, spezieller eines FileReaders. Der JSON-Tokener wird dann als Parameter für den passenden Konstruktor verwendet, also entweder *JSONObject*, wenn die JSON-Repräsentation ein Javascript Objekt serialisiert hat, oder *JSONArray*, wenn es sich um einen Javascript Array handelte. Sowohl JSONObject als auch JSONArray besitzen Methoden, um ihre Eigenschaften und Werte auszulesen. Während es sich bei JSONArray um eine geordnete Sequenz handelt, ist JSONObject ein assoziatives Datenfeld. Somit unterscheiden sich die Zugriffsarten.

6.4.2 JOpenDocument

JOpenDocument ist eine API zur einfachen Erstellung von OpenDocument Dateien. Das *Open Document Format for Office Applications*[12], kurz OpenDocument, ist ein XML-basiertes Dateiformat zur Repräsentation von Dokumenten aus der Textverarbeitung oder der Tabellenkalkulation. Ursprünglich von Sun Microsystems entwickelt, wird die Standardisierung in der Zwischenzeit von dem Technical Committee des OASIS Industriekonsortiums[13] durchgeführt. Zudem wurde der Standard als “ISO/IEC 26300:2006 Open Document Format

[12] http://www.oasis-open.org/standards
[13] http://www.oasis-open.org

for Office Applications (OpenDocument) v1.0“ [14] von der internationalen Vereinigung ISO definiert. Das Format hat den Vorteil, dass es sich um ein offenen Standard handelt und somit weit verbreitet ist. Obwohl JOpenDocument sowohl zur Erstellung von Textdokumenten als auch für Tabellen genutzt werden kann, wird in dem gegebenen Anwendungsfall der Einsatz auf die Erstellung von Tabellen beschränkt. Der Quellcode in 6.14 soll als einführendes Beispiel in JOpenDocument dienen und ist der Webseite des Projekts entnommen.

Listing 6.14: JOpenDocument

```
1  // Zuerst muss ein Datenmodell erstellt werden
2  final Object[][] data = new Object[6][2];
3  data[0] = new Object[] { "January", 1 };
4  data[1] = new Object[] { "February", 3 };
5  data[2] = new Object[] { "March", 8 };
6  data[3] = new Object[] { "April", 10 };
7  data[4] = new Object[] { "May", 15 };
8  data[5] = new Object[] { "June", 18 };
9
10 // Der Array dient zur Benennung der Kopfelemente der Spalten
11 String[] columns = new String[] { "Month", "Temp" };
12
13 /*
14  * der Konstruktor fuer neue Tabellen nimmt unter anderem
15  * javax.swing.TableModel als Konfigurationsparameter an
16 */
17 TableModel model = new DefaultTableModel(data, columns);
18
19 // Datei, in der wir die Tabellen speichern wollen
20 final File file = new File("temperature.ods");
21
22 // Tabelle erstellen und das Spreadsheet speichern
23 SpreadSheet.createEmpty(model).saveAs(file);
```

In dem Beispiel 6.14 werden Daten generiert und diese mit Hilfe der Klasse javax.swing.TableModel als Spreadsheet generiert. Es wird danach in der angegebenen Datei abgespeichert.

[14] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485

6.4.3 Das Programm *tool-evaluation*

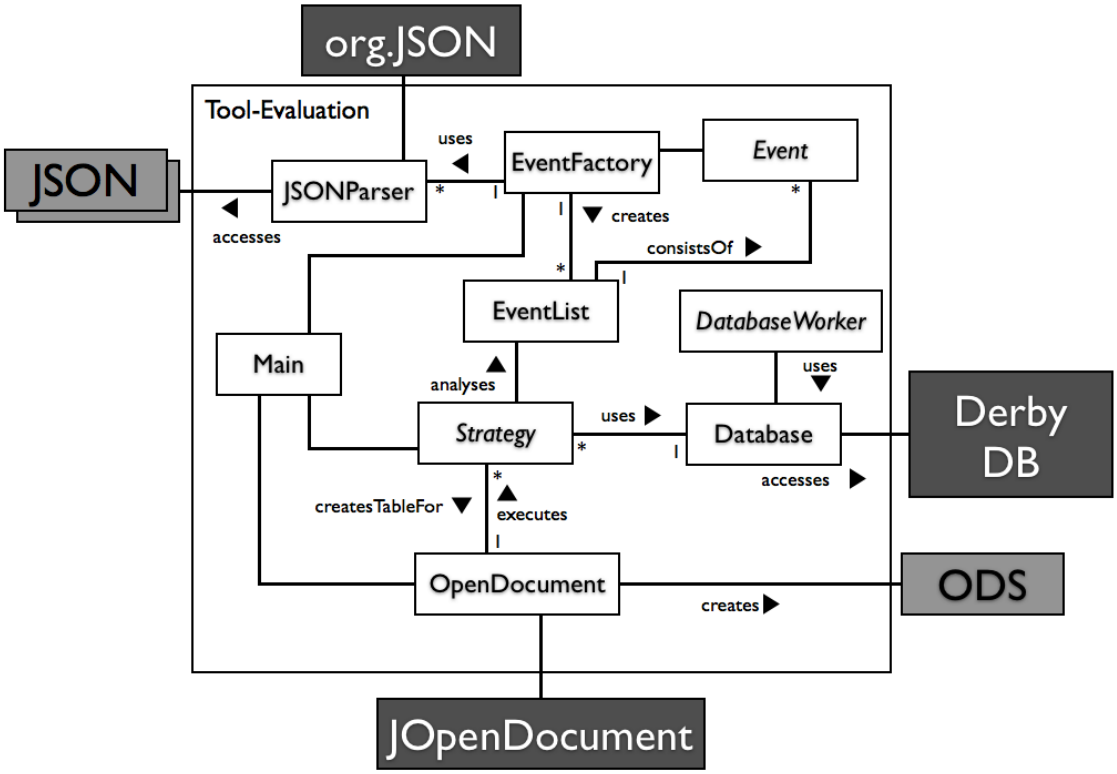

Abbildung 6.5: Klassendiagramm des Auswertungstools mit verwendeten Bibliotheken (dunkelgrau) und zugehörigen Dokumenten (hellgrau)

Nachdem nun beide Schnittstellen-APIs zum Einlesen und Ausgeben der jeweiligen Dokumente besprochen wurden, wird das Augenmerk auf den Aufbau des Auswertungstools gelegt. Hierzu muss zuerst erläutert werden, in welcher Form die Daten innerhalb der JSON-Dateien gespeichert sind.

Die Evaluationsergebnisse werden als Sequenz von Objekten in den JSON-Dateien abgelegt. Es handelt sich also um ein Array von speziellen Objekten. Diese Objekte haben alle zwei Eigenschaften, nämlich *time* und *type*. Bei time handelt es sich um einen Zeitstempel, der die verstrichenen Millisekunden seit

dem 1.1.1970 markiert. Bei type handelt es sich um eine Zeichenkette, die den Ereignistyp beschreibt. Es gibt mehrere Ereignisse. Einige von ihnen werden in der nachfolgenden Liste genannt.

- Evaluationsereignisse - Ereignisse, die beim Durchlauf der Evaluation entstehen.
  - *start* - Beginn der Evaluation.
  - *end* - Ende der Evaluation.
  - *next* - Anforderung der nächsten Sektion.
- Tracking-Ereignisse - Ereignisse, die durch das Verhalten des Probanden ausgelöst werden.
  - *gaze-Ereignisse* - Werden ausgelöst, wenn der Proband mit seinem Blick ein Bild ansieht oder von ihm wegsieht.
  - *mouse-Ereignisse* - Werden durch die Bewegungen der Maus über einem Bild oder von einem Bild weg ausgelöst.
  - *weitere Ereignisse* - Beispielsweise das Eingeben eines neuen Suchbegriffes.

Abhängig vom Typ des Ereignisses enthalten die Objekte noch weitere Informationen, die in dem Unterobjekt *payload* innerhalb des Ereignisobjekts gespeichert sind.

Nachdem die Struktur der JSON-Repräsentation beschrieben ist, kann diese mit Hilfe von JSON in Java in eine Repräsentation in Java Objekte übersetzt werden. Hierzu wurden die Klassen *eyevisionsearch.json.JSONParser* und *eyevisionsearch.logic.events.EventFactory* erstellt, die für die Übersetzung verantwortlich sind.

Listing 6.15: JSONParser und EventFactory

```
1 ArrayList<File> files;
2 ArrayList<EventList> eventLists;
3
4 // jede Datei wird geparsed und in eine EventList umgewandelt
5 for(File file : files)
6   eventLists.add(EventFactory.parseEvents(
7     new JSONParser(file)));
```

Wie im Programmauszug 6.15 zu sehen, enthält JSONParser einen Konstruktor mit einem Parameter. Dieser verweist auf die Datei, die es zu parsen gilt. Die Methode *parseEvents* der Klasse EventFactory wird aus dem gegebenen JSONParser eine *EventList* erstellt. Diese ist eine Liste von Spezialiserungen der Klasse *Event*.

Die abstrakte Klasse *Event* beschreibt die grundlegende Struktur von Evaluationsereignissen. Sie enthält Methoden, um den Typ und den Zeitstempel des Ereignisses auszulesen. Alle weiteren Ereignisse leiten sich von ihr ab.

Mit dieser Repräsentation lässt sich nun das OpenDocument Dokument erstellen. Hierbei wird das Strategy Pattern verwendet, um ein einfaches Wechseln zwischen verschiedenen gewünschten Repäsentationen zu ermöglichen.

Listing 6.16: Die Klasse Strategy in der Nutzung

```
1 //StrategyImplementation als Beispiel einer Implementation von Strategy
2 Strategy s = new StrategyImplementation();
3 s.addList(eventList);
4 TableModel t = s.compile();
```

Die *eyevisionsearch.logic.strategies.impl.DurationStrategy* erstellt beispielsweise eine Tabelle, die die Bearbeitungszeit der einzelnen Evaluationen errechnet.

Als Superklasse dient hierbei die abstrakte Klasse *eyevisionsearch.logic.strategies.Strategy*, die im Wesentlichen zwei Methoden enthält: *setList*, die eine Liste mit EventList-Objekten erfordert und *compile*, die verantwortlich ist für die eigentlichen Berechnungen. Sie gibt ein *org.javax.swing.TableModel* Objekt zurück.

Für alle Strategien ist es möglich, Zugang zu einer Derby Datenbank herzustellen, falls das vom Programmierer gewünscht ist. Aufbau und Zugang werden durch das Singleton *eyevisionsearch.logic.db.Database* gewährleistet. Um die Datenbank mit Daten zu füllen, werden Spezialisierungen der abstrakten Klasse *eyevisionsearch.logic.db.workers.DatabaseWorker* implementiert. Diese enthält die Methode *execute* und bekommt eine Liste von EventList-Objekten übergeben. Die Informationen, die in den EventList-Objekten gespeichert sind, werden je nach Implementierung in Tabellen innerhalb der Datenbank umgesetzt. Strategy-Programmierer können nun Zugriff auf diese Tabellen mit Hilfe

der *executeQuery* Methode von Database erhalten. Diese Methode akzeptiert als Parameter ein SQL-Statement und liefert ein *org.java.sql.ResultSet* zurück. Sie kann verwendet werden, falls die Vorteile einer Datenbank gewünscht sind oder benötigt werden.

Die einzelnen Elemente werden in der Klasse *eyevisionsearch.Main* zusammengeführt. Main enthält die statische Methode *execute*, die für die Ausführung der einzelnen Komponenten verantwortlich ist. Sie interpretiert auch die Konfigurationsdatei *config.json*, welche definiert, wo die jeweiligen Eingangsdaten sich befinden und die Ausgabedatei gespeichert werden soll. Zudem wird hier auch die Reihenfolge der verwendeten Strategien angegeben.

Neben den jeweiligen Strategien können nun in der verwendeten Tabellenkalkulation die statistischen Untersuchungen vorgenommen werden. Die Auswertungsmethoden werden in Kapitel 7 genauer beschrieben.

Das Evaluationstool ist so umgesetzt, dass es einfach ist, weitere Strategien und DatabaseWorker als Plugin zu implementieren. Die Strategy-Plugins müssen jeweils die Implementierung der Klasse Strategy in dem Ordner *eyevisionsearch.logic.strategies.impl* ablegen. Um die Strategie dann auszuführen, wird in der Konfigurationsdatei von Main der Name der Klasse angegeben.

DatabaseWorker werden umgesetzt, indem eine Implementierung der Klasse DatabaseWorker in dem Ordner *eyevisionsearch.logic.db.worker.impl* abgelegt wird. Um auf diese Klasse dann zugreifen zu können, wird über die Methode *requestDatabaseWorkerExecution* der Klasse Database der DatabaseWorker ausgeführt. Als Parameter wird der Name der Klasse als String übergeben. Die Methode ist so implementiert, dass der DatabaseWorker nur einmal ausgeführt wird, um fehlerhaftes Verhalten zu vermeiden.

7 Evaluation

Im nächsten Schritt gilt es, *EyeVisionSearch* zu evaluieren und zwar anhand folgender Hypothesen:

- Hypothese *H1* - „Die Einbindung des Clipboards ermöglicht, die Aufgaben in gleicher Vollständigkeit zu lösen, wie es mit der Baseline der Fall ist“
- Hypothese *H2* - „Der Einsatz eines Clipboards beschleunigt die Entscheidungsfindung bei der Bildersuche“
- Hypothese *H3* - „Das Clipboard erleichtert das Durchsuchen der Ergebnismenge“

Die Hypothesen sind mit dem Grundgedanken der Gebrauchstauglichkeit von *EyeVisionSearch* und der Frage nach dem Mehrwert der Erweiterung der Google Bildersuche aufgestellt worden. Die Hypothese *H1* kontrolliert die Wahrung der Effektivität der Clipboard-Varianten im Vergleich zur Baseline. Die Hypothese *H2* zielt auf die Effizienz der verschiedenen Varianten ab. Sie wird überprüft, indem die Bearbeitungsdauer der einzelnen Gruppen für die jeweiligen Aufgaben verglichen wird. Die Hypothese *H3* soll die Zufriedenheit des Benutzers beschreiben. Um sie zu untersuchen, werden die Meinungen der Probanden betrachtet und miteinander verglichen. Somit ist zu erkennen, dass die angestrebte Verbesserung des Systems vor allen in den Bereichen der Effizienz und der Benutzerzufriedenheit erwartet wird, während die Effektivität des Systems zumindest gleich bleiben sollte.

7.1 Entwurf

Um festzustellen, ob die Hypothesen zutreffen, wird eine summative Evaluation in Form des „Between-Subjects“ Designs durchgeführt und aufgabenbasie-

rend entworfen. „Between-Subjects“ bedeutet, dass die Probanden jeweils eine Variante des Evaluationsobjekts verwenden. In der Evaluation werden drei verschiedene Varianten evaluiert. Die Variante *A* verwendet *EyeVisionSearch* ohne die Unterstützung eines Clipboards. Hier werde die Eingabeinformationen zwar aufgezeichnet, aber nicht in ein Ranking oder eine Darstellung umgesetzt, sondern ausschließlich für die Auswertung verwendet. Diese Variante bildet die Baseline für den Vergleich, da sie der täglichen Verwendung der Google Bildersuche am nächsten kommt. Die Variante *B* des Evaluationsobjekts bietet dem Benutzer ein Clipboard an, das die Reihenfolge der Bilder in der Google Bildersuche übernimmt. Das erste Bild der Google Bildersuche erhält den Ranking-Wert 1. Bei allen folgenden Positionen wird dieser Wert für das jeweilige Bild halbiert. Stellt der Proband mehrere Anfragen an Google, in denen ein Bild jeweils vorkommt, wird der Ranking-Wert des Bildes aufsummiert. Das Evaluationsobjekt *C* bringt die Bilder in eine Ordnung, die es durch die Informationen des Eyetrackers ermittelt. Hierbei sind vor allem die Anzahl der Ereignisse sowie die Dauer der Fixierung entscheidend.

7.1.1 Teilnehmer

Es wurden 32 Probanden evaluiert, die zufällig und gleichmäßig über die drei Gruppen verteilt wurden. Sie waren zum Zeitpunkt der Evaluation im Alter zwischen 22 und 31 Jahren. Unter den Probanden waren 23 Männer und 9 Frauen. 20 Probanden studierten Informatik, CV oder Wirtschaftsinformatik, sieben Probanden studierten im Bereich Pädagogik und vier Probanden hatten eine abgeschlossene Ausbildung oder einen akademischen Grad in einem anderen Themengebiet. Ein Proband studierte im Bereich Geisteswissenschaften. Abbildung 7.1 zeigt die statistisch relevanten Informationen über die drei Probandengruppen. Alle Probanden wurden nach ihrer Erfahrung mit der Google Bildersuche und der Häufigkeit der Nutzung gefragt. Wie die Daten zeigen, verwenden alle Probanden die Bildersuche regelmäßig und sind mit der Nutzung sehr vertraut.

Bei zwei Probanden konnten die ermittelten Ergebnisse nicht für die Auswertung verwendet werden. Bei einem Proband kam es zu einem technischen Fehler des verwendeten Computers kurz vor Ende der Evaluation, wodurch die

		A	B	C
Alter[Jahre]	Mittelwert	25	26,1	26
	σ	3,528	2,923	2,667
Erfahrung mit Bildersuchen*	Mittelwert	3,411	4,188	3,745
	σ	0,791	0,791	0,91
Nutzung von Bildersuchen**	Mittelwert	2,8	3,3	3,3
	σ	0,632	1,252	0,949

*: Erfahrung reicht von:
1 = “Garnicht vertraut” bis 5 = “Vollkommen Vertraut”

**: Nutzung reicht von:
1 = „Garnicht“, 2 = „Monatlich“, 3 = „Wöchentlich“, 4 = „Täglich“, 5 = „Mehrmals täglich”

Abbildung 7.1: Allgemeine Informationen der Probanden nach Gruppen

Aufzeichnungen des Versuchs vernichtet wurden und somit nicht ausgewertet werden konnten. Die Ergebnisse eines weiteren Probanden wurden nicht ausgewertet, da die Person durch absichtliches Betrachten gewisser Bilder versuchte, die Technik des Systems zu verstehen und dadurch die Ergebnisse unbrauchbar wurden.

7.1.2 Aufgaben

Die Aufgaben, die den Probanden gestellt werden, modellieren alltägliche Anwendungen für die webbasierte Bildersuche. Sie orientieren sich an den aufgestellten Szenarien in Kapitel 2. Abbildung 7.4 präsentiert eine Liste der einzelnen Aufgaben. In der Evaluation führen die Probanden sechs verschiedene Aufgaben durch, die jeweils paarweise gruppierbar sind. Alle Aufgaben beinhalten die Stellung der Anfrage, das Durchsuchen der Ergebnismenge und die Speicherung des gewünschten Bildes. Abhängig vom Probanden, kann eine weitere Anfrage an die Bildersuche gestellt werden; das ist aber nicht unbedingt notwendig. Die erste Aufgabengruppe umfasst die Aufgaben T1 und T4 und enthält und beschreibt den genannten Ablauf. Die Aufgaben T2 und T5

	Aufgabe
T1	Suche ein Portraitbild von “Michael Jackson”
T2	Suche ein großes Bild (mind. 1200x900 Pixel) zum Thema: “Romantischer Rhein”
T3	Suche 5 Bilder von Sehenswürdigkeiten in Berlin
T4	Suche ein Portraitbild von “Lady Gaga”
T5	Suche eine Clipart Darstellung vom “Ampelmännchen”
T6	Suche 3 Bilder von Loriot im Zusammenhang mit “Weihnachten”

Abbildung 7.2: Aufgaben in der Evaluation

erweitern das Grundmodell um die Nutzung von Filtern, die Google den Benutzern anbietet. Bei Gruppe der Aufgaben T3 und T6 muss der Proband mehrere Bilder speichern. Hierbei ist es dem Probanden überlassen, ob er eine einzelne allgemeine Anfrage an die Suche stellt oder mehrere spezifische.

7.1.3 Erhobene Daten

Für die Evaluation werden eine Reihe von abhängigen Variablen gemessen, um feststellen zu können, wie die verschiedenen Varianten sich unterscheiden. Während der Durchführung der jeweiligen Aufgaben werden die folgenden Meßwerte aufgezeichnet:

- Augenbewegungen
- Mausbewegungen
- Suchanfragen
- Verwendung von Filtern
- Speichern von Bildern
- ggf. Öffnen und Schließen des Clipboards

- Antworten der Fragebögen

Die Fragebögen teilen sich in mehrere aufgabenbezogene Fragebögen und in einen abschließenden Fragebogen auf. Die Fragen werden in Abbildung 7.3 und 7.4 dargestellt und sind auf der Basis des IsoMetrics Usability Inventory [20] erstellt worden. Zusätzlich kann der Proband, wenn der dies möchte, ein freies Feedback, schriftlich oder mündlich am Ende der Evaluation geben.

	Aufgabenbezogener Fragebogen	
	A	B/C
Q1	Mit der Google Bildersuche kann ich zusammenhängende Arbeitsabläufe vollständig bearbeiten.	Mit Google&Clipboard kann ich zusammenhängende Arbeitsabläufe vollständig bearbeiten.
Q2	Google Bildersuche bietet mir alle Möglichkeiten, die ich für die Bearbeitung der Aufgabe benötige.	Google&Clipboard bietet mir alle Möglichkeiten, die ich für die Bearbeitung der Aufgabe benötige.
Q3	Die Ergebnisse von der Google Bildersuche entsprachen meinen Vorstellungen bei der Formulierung meiner Anfrage.	Die Ergebnisse von Google&Clipboard entsprachen meinen Vorstellungen bei der Formulierung meiner Anfrage.
Q4	Es müssen zuviele Eingabeschritte für die Bearbeitung der Aufgabe durchgeführt werden.	Es müssen zuviele Eingabeschritte für die Bearbeitung der Aufgabe durchgeführt werden.
Q5	Die Darstellung der Ergebnisse war für die Bearbeitung der Aufgabe geeignet.	Die Darstellung der Ergebnisse war für die Bearbeitung der Aufgabe geeignet.
Q6		Das Clipboard hat mich bei der Bearbeitung der Aufgabe unterstützt.
Q7		Die Darstellung der Ergebnisse der Google Bildersuche war übersichtlich.
Q8		Die Darstellung im Clipboard war übersichtlich.
Q9		Die Bilder im Clipboard waren für die Aufgabenstellung passend ausgewählt.
Q10		Die Reihenfolge der Bilder im Clipboard war für die Aufgabenstellung passend ausgewählt.

Abbildung 7.3: Aufgabenbezogene Fragebögen, aufgeteilt in Gruppe A (Baseline) und Gruppe B/C (Clipboard-Varianten)

7.1.4 Auswertung der erhobenen Daten

Die erhobenen Daten werden jeweils pro Gruppe und Aufgabe mit den Programmen OpenOffice.org 3.3.[1] und IBM SPSS 19[2] ausgewertet. Hierzu gehört die Ermittlung des Mittelwerts und der Standardabweichung. Zudem werden die Daten der Gruppen pro Aufgabe jeweils paarweise mit Hilfe des Mann-Whitney U-Test verglichen mit einem Signifikanzniveau von $\alpha = 0.05$.

[1] http://www.openoffice.org

[2] http://www-01.ibm.com/software/analytics/spss/

	Abschließender Fragebogen	
	A	B/C
F1	Die Verwendung der Google Bildersuche war für mich intuitiv.	Die Verwendung der Google&Clipboard war für mich intuitiv.
F2	Die Anwesenheit des Eyetrackers hat mich bei der Bearbeitung der Aufgaben gestört.	Die Anwesenheit des Eyetrackers hat mich bei der Bearbeitung der Aufgaben gestört.
F3	Es war zu jeder Zeit erkennbar, ob die Google Bildersuche noch arbeitet.	Es war zu jeder Zeit erkennbar, ob die Google&Clipboard noch arbeitet.
F4		Der Wechsel zwischen der Google Bildersuche und dem Clipboard war auf einfache Weise möglich.
F5		Das Clipboard ist eine sinnvolle Erweiterung zur Google Bildersuche.
F6		Der Umgang mit dem Clipboard war für mich leicht zu erlernen.
F7		Bei der Arbeit mit dem Clipboard traten Systemfehler auf.

Abbildung 7.4: Finale Fragebögen, aufgeteilt in Gruppe A (Baseline) und Gruppe B/C (Clipboard-Varianten)

7.2 Durchführung

Die Evaluation wird mit jeweils einem Probanden durchgeführt. Dem Probanden wird mitgeteilt, dass die Evaluation den Zweck habe, webbasierte Bildersuchen zu evaluieren. Zu Beginn wird der Eyetracker kalibriert, wie in 6.2 beschrieben. Nach der Kalibrierung wird dem Probanden signalisiert, dass er nun die Evaluation beginnen kann. Hierzu wird ihm das Browserfenster mit der geladenen Konfigurationsdatei auf dem Bildschirm dargestellt. Die jeweilige Gruppe wurde ihm, ohne sein Wissen, vorher zufällig zugewiesen. Zudem wird ihm angeboten, dass er Rückfragen stellen darf, falls ihm die Aufgabenstellung unklar ist. Allerdings bleibt es dabei, dass er die Aufgabe alleine lösen muss. Zur Einführung erhält der Proband eine Beschreibung der Evaluation sowie eine Einführung zur Google Bildersuche. Sollte seine Variante ein Clipboard enthalten, wird ihm dieses ebenfalls erklärt. Danach führt der Proband eine Beispielaufgabe durch, die ihn in die Vorgehensweise der Evaluationsaufgaben einführen und ihm die Verwendung des Systems erläutern soll. Sie wird bei der späteren Auswertung nicht beachtet. Danach führt er die einzelnen Aufgaben durch.

Nachdem der Proband die Aufgabe bearbeitet hat, wird ihm ein aufgabenbezogener Fragebogen präsentiert, den er ausfüllen muss. Die Antwortmöglichkeiten werden, wie bereits beschrieben, durch einen Slider repräsentiert,

der zwischen „Keine Zustimmung“ und „Volle Zustimmung“ positioniert wird. Sind alle Aufgaben vollständig bearbeitet und die zugehörigen Fragebögen ausgefüllt, muss der Proband noch einen abschließenden Fragebogen ausfüllen. Zudem kann er ein freies Feedback aufschreiben. Nachdem dies geschehen ist, werden ihm noch in einem Interview kurze Fragen über den Verlauf der Evaluation und die Verwendung des Evaluationsobjekts gestellt. Falls während der Evaluation Problembereiche aufgefallen sind, werden diese kurz angesprochen. Danach ist der Proband mit der Evaluation fertig.

7.3 Ergebnisse

Die Ergebnisse der Evaluation werden nun aufgeteilt nach Effizienz, Effektivität und Benutzerzufriedenheit.

7.3.1 Effektivität

Um die Effektivität zu messen, wurde beobachtet, inwieweit es den Probanden möglich war, die Aufgabe zu lösen. Hierzu wurde überprüft, ob die Probanden die geforderte Anzahl an Bildern pro Aufgabe speichern. Alle Probanden konnten die Aufgabe im geforderten Umfang lösen, mit drei Ausnahmen:

- Proband P07 (Gruppe C) speicherte bei Aufgabe T6 nur ein Bild.
- Proband P25 (Gruppe A) speicherte bei Aufgabe T1 kein Bild.
- Proband P27 (Gruppe C) hat in Aufgabe T3 ein Bild weniger gespeichert.

7.3.2 Effizienz

Um die Effizienz der verschiedenen Varianten zu vergleichen, werden die benötigte Dauer, die Menge der registrierten Ereignisse der Maus, die Menge der „Gaze In“ Ereignisse sowie die Menge der Anfragen an die Google Bildersuche der jeweiligen Gruppen miteinander verglichen. Die Abbildung 7.5 zeigt hierzu die Mittelwerte und Standardabweichung. Die Graphik 7.6 stellt die Ergebnisse als Average-High-Low Diagramm dar. Zudem wurde hinsichtlich der Dauer der einzelnen Aufgabenbearbeitungen der jeweiligen Gruppen pro Aufgabe

paarweise der Mann-Whitney U-Test angewendet mit einem Signifikanzniveau von $\alpha = 0.05$. Hierbei zeigte sich ein signifikanter Unterschied zwischen den Gruppen A und C in Aufgabe T1.

Bei Gruppe B in Aufgabe T5 ist ein Meßwert entfernt worden, da hier der Proband fast 6 Minuten für die Durchführung der Aufgabe benötigte und somit die Ergebnisse verfälscht hätte. Wie in Abbildung 7.6 zu sehen, sind die Werte ohne diesen Ausreißer vergleichbar mit denen der Variante C.

7.3.3 Benutzerzufriedenheit

Für die Messung der Benutzerzufriedenheit wurden die Fragebögen der Evaluation ausgewertet. Bei den Fragebögen gibt es pro Aufgabe jeweils einen aufgabenbezogenen sowie einen abschließenden Fragebogen. Für jede Frage konnten die Probanden ihre Antwort über Slider-Elemente, wie in Kapitel 6.3 beschrieben, auf einem Intervall frei wählen. Die beiden Enden des Slider-Elements wurden dabei mit „Keine Zustimmung und „Volle Zustimmung“ beschrieben. Für die Evaluation wurden die so erhaltenen Werte auf einer fünfstufigen Likert-Skala mit Werten von 1 für „keine Zustimmung“ bis 5 für „Volle Zustimmung“ abgebildet. Für diese wurden jeweils die Mittelwerte und Standardabweichungen errechnet. Diese werden in den Abbildungen 7.7 und 7.8 dargestellt.

Im nächsten Schritt galt es, die Daten auf signifikante Unterschiede zwischen den Gruppen zu testen. Hierzu wurde der Mann-Whitney U-Test mit einem Signifikanzniveau von $\alpha = 0,05$ verwendet. Die Ergebnisse werden in den Abbildungen 7.3.3 und 7.10 präsentiert.

7.4 Diskussion der Ergebnisse

Im Folgenden werden die Teilhypothesen diskutiert, die am Anfang des Kapitels aufgestellt wurden.

7.4.1 Effektivität

Auch wenn einige Probanden keine Bilder gespeichert haben, kann man nicht davon ausgehen, dass dies auf die jeweiligen Varianten zurückzuführen ist.

		T1		T2		T3	
		Mittelwert	σ	Mittelwert	σ	Mittelwert	σ
Dauer [s]	A	45,151	21,083	94,355	65,338	153,913	103,697
	B	66,057	26,413	84,725	43,129	160,604	75,935
	C	74,686	39,555	108,064	52,692	207,911	75,644
“Gaze In” Ereignisse	A	34,5	55,806	73,4	97,694	143,8	260,762
	B	37,3	54,148	42,3	73,124	107,5	140,61
	C	50,4	68,696	58,1	105,123	113,8	140,764
Maus Ereignisse	A	20,8	15,186	74	50,621	112,2	68,21
	B	34	17,01	46	29,799	102,8	44,299
	C	36,9	23,69	59,5	54,114	148,3	88,839
Anwendung von Filtern & Suchanfragen	A	1	0	3	1,633	2,8	3,12
	B	1,3	0,675	2,1	0,316	1,7	1,059
	C	1,5	0,972	2,4	0,966	3,1	2,558
Clipboard geöffnet	B	0,9	0,568	1	0,471	1,3	0,949
	C	0,7	0,483	0,9	0,994	1,7	1,829
Bilder gespeichert über Clipboard	B	0,6	0,516	0,4	0,516	1,6	1,578
	C	0,1	0,316	0,3	0,483	1,2	1,687

		T4		T5		T6	
		Mittelwert	σ	Mittelwert	σ	Mittelwert	σ
Dauer [s]	A	49,362	26,842	44,574	21,499	134,51	65,122
	B	68,731	20,935	105,432	122,268	135,486	69,474
	C	66,108	17,77	55,156	13,729	138,808	101,546
“Gaze In” Ereignisse	A	25,5	58,439	20,667	31,492	109,714	155,414
	B	41	42,032	50,833	168,347	74	159,278
	C	40,875	50,011	25,714	28,925	84,5	179,355
Maus Ereignisse	A	20,375	10,391	17,875	9,25	97,857	79,855
	B	39,333	27,134	66,667	108,31	62,167	26,447
	C	36,375	22,36	36,625	30,298	86,375	122,518
Anwendung von Filtern & Suchanfragen	A	1,375	0,744	1,875	1,126	2,143	1,927
	B	1,167	0,408	2,5	1,761	1,167	0,408
	C	1,25	0,463	2,125	2,416	2,25	2,816
Clipboard geöffnet	B	1	0,894	1,443	0,408	0,833	0,408
	C	0,75	0,463	0,5	0,535	1	0,756
Bilder gespeichert über Clipboard	B	0,167	0,408	0,667	0,516	0,667	0,816
	C	0,375	0,518	0,25	0,463	1	1,414

Abbildung 7.5: Aufbereitung der Dauer und der Ereignisse nach Aufgaben und Gruppen

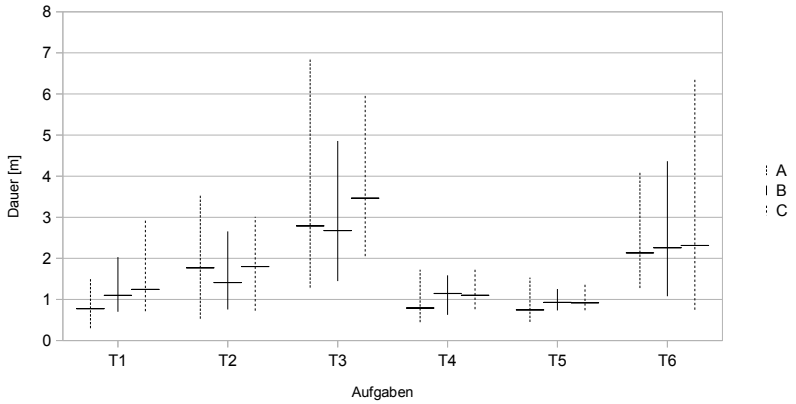

Abbildung 7.6: Average-High-Low Diagramm der Dauer, aufgeschlüsselt nach Gruppen

Zwar haben zwei Probanden bei der Eyetracker-basierten Variante kein bzw. ein Bild zu wenig gespeichert, allerdings ist diese Variante von der Darstellung und Verwendung des Clipboards identisch zur Variante B, bei der kein Proband ein Bild vergessen hat. Somit kann man davon ausgehen, dass es keine Unterschiede zwischen den Varianten im Bezug auf die Effektivität gibt. Daher kann die Hypothese *H1* als erfüllt angesehen werden. Bei insgesamt etwa 485 gespeicherten Bildern sind sechs fehlende Bilder im Rahmen der Fehlertoleranz.

7.4.2 Effizienz

Wie in den Abbildungen 7.5 und 7.6 zu erkennen, gibt es einen auffälligen Unterschied in der Dauer der Aufgaben T1 und T4. Der Signifikanztest von Gruppe A und C in Aufgabe T1 ergab einen signifikanten Unterschied zwischen den beiden. Gruppe A bearbeitete im Mittel die Aufgabe etwa 25 bzw. 8 Sekunden schneller als die beiden anderen Varianten. Der Unterschied lässt sich dadurch erklären, dass diese beiden Aufgaben relativ schnell auszuführen sind, da nur ein einzelnes Bild gespeichert werden soll. Neben dem einfachen Abspeichern öffneten die meisten Probanden der beiden anderen Gruppen noch das Clip-

		T1		T2		T3	
		Mittelwert	σ	Mittelwert	σ	Mittelwert	σ
Q1	A	4,173	0,76	4,233	0,656	4,251	0,769
	B	3,708	1,375	3,917	1,241	4,083	1,081
	C	3,868	1,009	3,904	1,047	3,825	1,245
Q2	A	4,577	0,488	4,143	0,718	4,101	1,03
	B	3,909	1,371	3,848	1,185	4,163	0,984
	C	4,14	0,92	4,143	1,027	3,619	1,122
Q3	A	4,425	0,739	3,212	0,756	4	0,844
	B	3,857	0,804	3,565	1,098	3,757	1,093
	C	3,884	1,183	3,901	0,727	3,593	1,245
Q4	A	1,229	0,192	1,509	0,666	2,044	1,272
	B	1,852	0,936	1,876	0,924	1,639	0,575
	C	1,437	0,589	1,743	0,945	1,675	0,731
Q5	A	4,492	0,421	4,028	0,917	4,025	0,906
	B	4,036	0,897	4,199	0,627	4,059	1,046
	C	3,859	1,04	3,528	1,147	3,667	0,992
Q6	A						
	B	3,019	1,385	3,415	1,377	3,696	1,137
	C	1,916	1,007	2,547	1,222	2,299	1,213
Q7	A						
	B	4,091	0,928	4,075	0,668	4,175	0,833
	C	3,557	1,272	3,872	0,85	3,909	0,834
Q8	A						
	B	4,369	0,662	4,241	0,577	4,067	0,836
	C	4,152	0,704	3,896	0,909	3,6	1,232
Q9	A						
	B	3,633	0,916	3,472	1,068	3,88	0,745
	C	3,161	1	3,301	0,86	3,068	1,143
Q10	A						
	B	3,337	0,705	3,041	0,915	3,561	0,618
	C	2,929	0,435	2,968	0,632	3,052	0,964

		T4		T5		T6	
		Mittelwert	σ	Mittelwert	σ	Mittelwert	σ
Q1	A	4,133	0,798	4,343	0,536	3,905	0,592
	B	3,876	1,388	4,102	1,25	3,813	0,983
	C	4,072	0,847	3,758	1,176	3,5	1,32
Q2	A	4,26	0,623	4,12	0,685	3,743	0,904
	B	3,824	1,404	4,376	0,944	3,571	1,22
	C	4,313	0,727	4,192	0,673	3,097	0,792
Q3	A	4,113	0,891	4,347	0,573	2,677	1,147
	B	3,678	0,722	4,156	1,262	3,116	1,026
	C	4,362	0,538	3,472	0,84	3,293	1,272
Q4	A	1,762	1,048	1,752	1,088	1,885	0,855
	B	1,282	0,248	1,247	0,247	1,373	0,432
	C	1,388	0,451	1,674	0,792	1,545	0,633
Q5	A	4,193	0,788	4,128	0,788	3,692	1,028
	B	3,876	0,895	4,551	0,637	3,38	1,084
	C	4,025	0,783	3,594	0,923	3,31	0,982
Q6	A						
	B	2,173	1,255	3,813	1,55	2,211	1,052
	C	3	1,298	2,002	1,13	2,79	1,486
Q7	A						
	B	3,996	0,801	4,04	0,846	3,513	1,254
	C	3,742	1,002	3,783	1,067	3,587	0,982
Q8	A						
	B	3,716	0,82	4,124	0,822	3,42	0,637
	C	4,288	0,623	3,608	1,293	3,093	1,675
Q9	A						
	B	2,769	0,554	4,033	1,525	2,667	0,747
	C	3,407	1,361	2,905	1,338	2,917	1,311
Q10	A						
	B	2,256	0,848	3,684	1,459	2,838	0,686
	C	3,527	1,262	2,712	1,301	2,785	1,328

Abbildung 7.7: Auswertung der aufgabenbezogenen Fragebögen nach Aufgaben und Fragen/Gruppen (1 = keine Zustimmung, 5 = Volle Zustimmung)

	A		B		C	
	Mittelwert	σ	Mittelwert	σ	Mittelwert	σ
F1	4,191	0,653	4,348	0,531	3,161	1,382
F2	1,48	0,681	1,209	0,258	1,349	0,709
F3	3,161	1,184	4,077	0,962	3,345	1,367
F4			4,235	0,719	4,239	1,195
F5			3,492	0,573	4,761	1,125
F6			4,369	0,717	4,343	1,046
F7			1,256	0,649	2,283	1,578

Abbildung 7.8: Aufbereitung der abschließenden Fragebögen nach Aufgaben und Fragen/Gruppen (1 = keine Zustimmung, 5 = Volle Zustimmung)

		T1	T2	T3	T4	T5	T6
	A-B	.705	.820	.880	.897	.948	.796
Q1	A-C	.849	.545	.570	.752	.528	.600
	B-C	.759	.820	.649	.795	.795	.747
	A-B	.545	.880	.820	.897	.301	.846
Q2	A-C	.344	.597	.257	.599	.598	.141
	B-C	.940	.791	.139	.517	.795	.439
	A-B	.075	.406	.734	.115	.697	.519
Q3	A-C	.211	.140	.545	.598	.115	.293
	B-C	.597	.677	.677	*.071*	.302	.518
	A-B	.226	.496	.733	.439	.196	.175
Q4	A-C	.650	.970	.449	.400	.598	.293
	B-C	.288	.703	.970	.948	.648	.897
	A-B	.364	.910	.910	.519	.245	.519
Q5	A-C	.272	.406	.496	.674	.344	.462
	B-C	.704	.150	.273	.795	.243	.897
Q6	B-C	*.069*	.131	**.021**	.121	**.027**	.560
Q7	B-C	.256	.545	.595	.651	.897	.948
Q8	B-C	.569	.363	.427	.155	.796	.796
Q9	B-C	.496	.307	*.070*	.401	**.045**	.519
Q10	B-C	.307	.290	.151	*.093*	*.071*	1.000

Abbildung 7.9: Aufgabenbezogener Fragebogen: Mann-Whitney U-Test Ergebnisse über Fragen und Gruppen (Signifikanzniveau $\alpha = 0.05$)

	Sig.		
	A-B	A-C	B-C
F1	.706	.880	.791
F2	.569	.620	.819
F3	.070	.791	.325
F4			.447
F5			.162
F6			.594
F7			.137

Abbildung 7.10: Abschließender Fragebogen: Mann-Whitney U-Test Ergebnisse über Fragen und Gruppen (Signifikanzniveau $\alpha = 0.05$)

board, welches nicht für Gruppe A vorhanden war. Auch wenn es sich dabei um eine einfache Handlung handelt, fällt diese im Verhältnis zur restlichen Zeit der Aufgabe ins Gewicht. Bei Aufgabe T2 und T3 schneidet die Variante B besser ab, als die anderen beiden Varianten. Es ergibt sich kein Vorteil für die Clipboard-Varianten im Bezug auf die Bearbeitungsdauer bei den Aufgaben T1, T4, T5 und T6. Nur bei den Aufgaben T2 und T3 sind die Probanden mit der Google-basierten Clipboard-Variante schneller. In keiner Aufgabe schneidet die Eyetracker-basierte Variante besser ab als die anderen Varianten. Somit muss die Hypothese *H2* verworfen werden.

7.4.3 Benutzerzufriedenheit

Wie die statistische Auswertung der Daten belegt, gibt es nur wenige signifikante Unterschiede über die Gruppen verteilt. Der Mann-Whitney U-Test zeigt, wie in Abbildung 7.3.3 dargestellt, bei der Frage Q6 in Aufgabe T3 (siehe Abbildung 7.11) und bei den Fragen Q6 und Q9 in der Aufgabe T5 (siehe Abbildung 7.12 sowie Abbildung 7.13) signifikante Unterschiede an, jeweils im Vergleich zwischen den beiden Clipboard-Varianten.

Bei der Auswertung der abschließenden Fragebögen ergaben sich keine signifikanten Unterschiede, wie in Abbildung 7.10 zu sehen. Hier wurde ebenfalls der Mann-Whitney U-Test mit einem Signifikanzniveau von $\alpha = 0,05$ verwendet und die drei Gruppen jeweils paarweise verglichen. Die zwei Aufgabenstellungen, bei denen die Unterschiede auftaten, waren zum einen die Suche

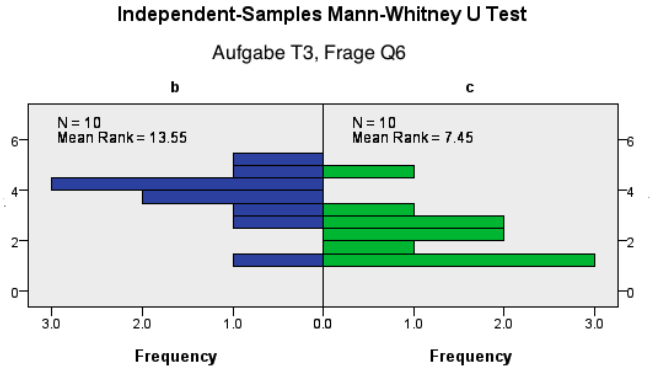

Abbildung 7.11: Detailansicht Mann-Whitney U-Test Ergebnis: Aufgabe T3, Frage Q6

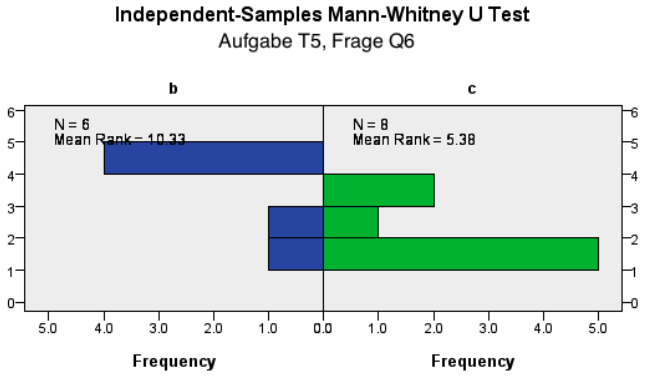

Abbildung 7.12: Detailansicht Mann-Whitney U-Test Ergebnis: Aufgabe T5, Frage Q6

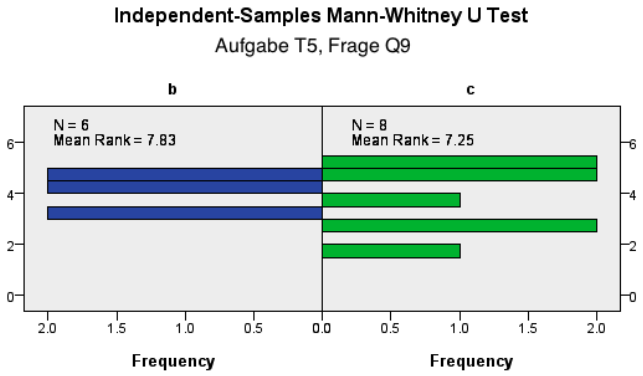

Abbildung 7.13: Detailansicht Mann-Whitney U-Test Ergebnis: Aufgabe T5, Frage Q9

nach fünf Bildern von Sehenswürdigkeiten in Berlin und zum anderen die Suche nach einer Clipart-Darstellung des Ampelmännchens. Der signifikante Unterschied trat bei den Fragen „Das Clipboard hat mich bei der Bearbeitung der Aufgabe unterstützt“ und „Die Bilder im Clipboard waren für die Aufgabenstellung passend ausgewählt.“ auf, wobei das Ergebnis der letzten Frage sich nur bei der zuletzt genannten Aufgabe signifikant anders darstellte. Bei drei Unterschieden schnitt die Variante B besser ab als die Variante C des Systems. Beide Varianten zeigen jeweils keinen signifikanten Unterschied zu Variante A.

Zuletzt werden die Fragen des abschließenden Fragebogens betrachtet. Hier ist die Frage F1 interessant, da sie einen direkten Vergleich zwischen den Varianten zulässt. Zudem werden die Werte der Fragen F4 bis F7 betrachtet. Der Mann-Whitney U-Test der oben genannten Fragen ergab keine Unterschiede. Somit unterscheiden sich die verschiedenen Varianten in ihrer Einfachheit nicht signifikant. Anhand der interpretierten Daten kann gesagt werden, dass die Verwendung eines Clipboards, welches primär auf den Informationen der Google Bildersuche beruht, keinen Mehrwert für den Benutzer bietet. Somit trifft die Hypothese *H2* nicht zu.

7.4.4 Kommentare der Probanden

An dieser Stelle werden ausgewählte Kommentare der Probanden vorgestellt. Diese sind nicht direkt Teil der qualitativen Auswertung durch das kontrollierte Experiment, sondern dienen als Überblick, wie die Probanden das System an sich empfanden. Über alle Gruppen verteilt fiel auf, dass bei den Aufgaben T2 und T3 die meisten Schwierigkeiten auftraten. Bei T2 lag dies am gesuchten Begriff „Romantischer Rhein“. Zum einen ist die Entscheidung, ob ein Bild romantisch ist oder nicht, subjektiv. Zum zweiten ist schlecht zu bestimmen, ob der abgebildete Fluss auf den Bildern wirklich der Rhein ist. Bei T3 waren Sehenswürdigkeiten in Berlin gesucht. Einige Probanden kannten sich in Berlin aus und begannen, jeweils spezifische Anfragen für Sehenswürdigkeiten zu stellen. Die Mehrheit allerdings verwendete eine allgemeine Anfrage und blätterte die Ergebnisse von Google durch. Hier ergab sich das Problem, dass einige dieser Probanden nicht wussten, ob die Bilder tatsächlich Sehenswürdigkeiten in Berlin darstellten. Ein Proband erwähnte im Interview, dass er diese Aufgabe nicht mit Hilfe der Google Bildersuche lösen würde, sondern beispielsweise mit Hilfe von Wikipedia, um sich anhand der Sehenswürdigkeiten die Bilder herauszusuchen. Ein ähnliches Bild stellte sich auch bei einem Probanden in der Aufgabe T6 heraus, die vom gleichen Szenario abstammt. Hier waren drei Bilder von Loriot gesucht, die mit dem Thema Weihnachten zusammenhängen. Der Proband kannte Loriot nicht und hatte somit Schwierigkeiten, die Aufgabe zu bewältigen.

Was die beiden Gruppen mit den Clipboard-Erweiterungen betrifft, zeigte sich ein sehr gemischtes Bild an Meinungen. Einige Probanden empfanden das Clipboard besonders bei der Aufgabe T1, T3 sowie T4 nützlich. Bei T1 und T4 sollte jeweils ein Portrait zu einer bestimmten Person gesucht werden. Hier eignet sich das Clipboard bei einer großen Auswahl an Bildern, da es die Auswahl einschränkt. Bei anderen Aufgaben, wie T2, stellte sich auch im Clipboard das Problem, dass die Probanden nicht entscheiden konnten, ob das Bild den Rhein darstellt. Wie bereits erwähnt, ergab sich das Problem schon bei der Google Bildersuche selbst.

Ein Teil der Probanden empfand das Clipboard als eine unnötige Ergänzung. Dies lässt sich dadurch erklären, dass ihnen das Clipboard nicht mehr Infor-

mationen bietet als die Bildersuche, sondern eine Reduzierung der Informationsmenge darstellt. Bei der Durchführung der Aufgaben kam es zu Problemen, die spezifisch für die jeweiligen Clipboardvarianten sind. Bei B-Gruppe kam es vor, dass die Probanden sich bei der Anfrage vertippten. Dies führte in dieser Variante zu falschen Kandidaten. In der C-Gruppe kam es vor, dass Probanden das Clipboard vor der Eingabe einer Anfrage öffneten. Sie waren daraufhin überrascht, dass das Clipboard leer war, obwohl dieses Verhalten in der Einleitung beschrieben war. Allerdings verwirrte es die jeweiligen Probanden dennoch. Bei einem Probanden der Gruppe C kam es in einer Aufgabe zum Ausfall der Übermittlung von Informationen zwischen Eyetracker und *EyeVisionSearch*, wodurch keine Daten empfangen wurden und das Clipboard leer blieb.

7.4.5 Interpretation

Das schlechte Abschneiden der Clipboard-Varianten in den Punkten der Effizienz und Benutzerzufriedenheit lässt sich auf folgenden Sachverhalt zurückführen: Zum einen waren die Probanden aller Gruppen im Allgemeinen vertraut mit der Google Bildersuche und benutzten sie relativ regelmäßig, wie in Abbildung 7.1 zu sehen. Im Gegensatz hierzu war das Clipboard für die meisten Teilnehmer ein neues Konzept. Das lässt sich vor allem aufgrund von Rückfragen in Bezug auf das Clipboard belegen. Im schriftlichen Feedback kam häufiger die Frage auf, wofür das Clipboard notwendig sei. Außerdem wurde häufig gefragt, wie die Heuristik des Clipboards funktioniert. Allerdings war es aufgrund des Versuchsaufbaus nicht möglich, diese Frage während der Evaluation zu beantworten. Die Tatsache, dass die Handlungen des Probanden zwar die Auswahl beeinflussen, aber unklar ist, wie dies geschieht, hatte eine irritierende Wirkung.

Zum anderen ist die Konzeption der Google Bildersuche sehr gut geeignet, um eine große Menge an Bildern zu sichten. Gerade die Änderungen in den letzten Monaten, vor allem das Nachladen der Bilder im Hintergrund beim Bildlauf, vereinfachen die Nutzung. Somit ist es möglich, viele Bilder ohne oder mit geringen Wartezeiten zu betrachten. Obwohl diese Eigenschaften auch in die Konzeption des Clipboards eingeflossen sind, entwickelte sich aus der Reduk-

tion der Ergebnismenge kein Vorteil für den Nutzer.

8 Fazit

Abschließend lässt sich über den Einsatz von Eyetrackern zur Bedienung von webbasierten Bildersuchen sagen, dass es zumindest in der gewählten Form zusammen mit einem Clipboards keine signifikante Verbesserung zum traditionellen Modell darstellt. Statistisch ergeben sich wenig Unterschiede zwischen den einzelnen Variationen. In der qualitativen Auswertung ließ sich jedoch erkennen, dass unter Umständen bei manchen Aufgaben ein Nutzen aus dem Clipboard gezogen werden kann.

Auch wenn keine statistische Signifikanz erkennbar ist, ergeben sich bei einzelnen Probanden positive Effekte für die Bedienung von Bildersuchen. So stellt die Reduktion der Menge an Bildern bei eindeutigen Suchanfragen mit einer sehr großen Ergebnismenge einen Vorteil für diese Probanden dar. Beispiele hierfür sind die Portraitsuchen oder die Suche nach Sehenswürdigkeiten. Dieser Effekt ist aber abhängig von den Vorlieben des Benutzers und kann nicht verallgemeinert werden.

8.1 Problembehandlung

Aus technischer Sicht problematisch waren die Seiteneffekte, die sich durch den Aufbau der verwendeten Bildersuche ergaben. Zum einen war es nicht einfach, an die notwendigen Informationen des Basissystems zu gelangen, da viele Aktionen durch die Javascript- und AJAX-basierte Umsetzung für einen Aussenstehenden kompliziert nachzuvollziehen sind. Abhilfe konnte hier in einem gewissen Rahmen durch den verwendeten Browser geschaffen werden. Dennoch ergaben sich viele Probleme mit der fehlerfreien Kommunikation. Als Beispiel ist der unterschiedliche Zeitpunkt zu nennen, zu dem das Ereignis einer neuen Anfrage an die Erweiterung weitergeleitet wird. Dieses Problem kann wahrscheinlich durch die Nachbildung des Basissystems mit Hilfe der

Google Search API beseitigt werden.

Anfänglich ergaben sich auch technische Schwierigkeiten mit der Benutzung des Eyetrackers. Wie sich herausstellte, arbeitet der Eyetracker zu bestimmten Tageszeiten in unzureichendem Maße. Dies ließ sich auf die ungünstigen Lichtverhältnisse zurückführen, die in dem benutzten Raum zu bestimmten Uhrzeiten herrschten. Zudem existierte ein Problem mit der Wärmeentwicklung des Eyetrackers. Die Qualität der Messergebnisse nahm bei längerer, kontinuierlicher Nutzung des Eyetrackers stark ab. Dieses Problem wurde durch die Eingrenzung der Evaluationen auf optimale Tageszeiten sowie durch Pausen zwischen den Evaluationen verbessert.

Die Einbindung der Text 2.0 Javascript-Schnittstelle in eine fremde Webseite funktionierte nach anfänglichen Schwierigkeiten relativ gut. Das Hauptproblem bestand hierbei wieder darin, den Zeitpunkt zu finden, zu dem das Basissystem fertig aufgebaut ist, ausgehend von den gegebenen AJAX Ereignissen. Das Problem wurde schließlich durch eine zeitverzögerte Anbindung der Javascript-Schnittstelle eingegrenzt. Dieses Vorgehen machte es wiederum notwendig, ein Overlay über die Bilder einzublenden, damit die Probanden sich nicht die Bilder ansehen konnten, bevor die Eyetracker Daten erkannt wurden.

8.2 Ausblick

Als Feld für Verbesserungen wäre die Auswertung der Inhalte der Bilder zu nennen. So könnte man beispielsweise ähnliche Bilder innerhalb des Clipboards clustern und dem Probanden diese als eine Art „Bildhaufen“ präsentieren, die er dann durchsuchen kann. Ausserdem könnte man Dubletten eliminieren, um den Inhalt des Clipboards konzentrierter zu gestalten. Dies wäre eine herausfordernde Umsetzung, da die aufwändigen inhaltsbasierten Verfahren mit den großen Informationsmengen, die durch den Eyetracker entstehen, in Einklang gebracht werden müssten.

Literaturverzeichnis

[1] Alexa, the Web Information Company: Top sites. http://www.alexa.com/topsites (2010)

[2] Hyvönen,Eero and Saarela,Samppa and Viljanen,Kim: Intelligent Image Retrieval and Browsing Using Semantic Web Techniques – A Case Study. In: Proceedings of the International SEPIA Conference at the Finnish Museum of Photography. (2003)

[3] Google Insights: Suchbegriff: Wallpaper, weltweit, Januar 2008-Januar 2011 [Stand: 15.01.2011]. http://www.google.com/insights/search/#q=wallpaper&date=1%2F2008'%2037m&gprop=images&cmpt=q (2011)

[4] Biedert, R., Buscher, G., Schwarz, S., Möller, M., Dengel, A., Lottermann, T.: The Text 2.0 Framework – Writing Web-Based Gaze-Controlled Realtime Applications Quickly and Easily. In: Proceedings of the International Workshop on Eye Gaze in Intelligent Human Machine Interaction (EGIHMI) held in conjunction with IUI 2010. (2010)

[5] Google Insights: Bildersuche allgemein, weltweit [Stand: 15.01.2011]. http://www.google.com/insights/search/#gprop=images (2011)

[6] Hjørland, B., Christensen, F.S.: Work tasks and socio-cognitive relevance: A specific example. Journal of the American Society for Information Science and Technology **53** (2002) 960–965

[7] Wikipedia: Relevance — wikipedia, the free encyclopedia (2011) [Online; accessed 2-November-2011].

[8] Borlund, P.: The concept of relevance in IR. Journal of the American Society for Information Science and Technology **54** (2003) 913–925

[9] Lindsay, P.H., Norman, D.A.: Einführung in die Psychologie: : Informationsaufnahme und -verarbeitung beim Menschen. Springer (1981)

[10] Klami, A., Saunders, C., de Campos, T.E., Kaski, S.: Can relevance of images be inferred from eye movements? In: Proceeding of the 1st ACM international conference on Multimedia information retrieval. MIR '08, New York, NY, USA, ACM (2008) 134–140

[11] Google: more than meets the eye. http://googleblog.blogspot.com/2009/02/eye-tracking-studies-more-than-meets.html (2009)

[12] Hardoon, D.R., Anjaki, A., Poulamäki, K., Kaski, S.: Information Retrieval by Interferring Implicit Queries from Eye Movements. (2007)

[13] Corsato, S., Mosconi, M., Porta, M.: An eye tracking approach to image search activities using rsvp display techniques. In: Proceedings of the working conference on Advanced visual interfaces. AVI '08, New York, NY, USA, ACM (2008) 416–420

[14] Jeff, A.J., Jaimes, R., Pelz, J., Grabowski, T., Babcock, J., fu Chang, S.: Using human observers' eye movements in automatic image classifiers. In: Proceedings of SPIE Human Vision and Electronic Imaging VI. (2001) 373–384

[15] Pasupa, K., Klami, A., Saunders, C.J., Kaski, S., Szedmak, S., Gunn, S.R.: Learning to rank images from eye movements. In: In HCI '09: Proceeding of the IEEE 12th International Conference on Computer Vision (ICCV'09) Workshops on Human-Computer Interaction. (2009) 2009–2016

[16] Hardoon, D.R., Pasupa, K.: Image ranking with implicit feedback from eye movements. In: Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications. ETRA '10, New York, NY, USA, ACM (2010) 291–298

[17] Kozma, L., Klami, A., Kaski, S.: GaZIR: gaze-based zooming interface for image retrieval. In: Proceedings of the 2009 international conference on Multimodal interfaces. ICMI-MLMI '09, New York, NY, USA, ACM (2009) 305–312

[18] Google: ooh-ahh-google-images-presents-nicer.html. http://googleblog.blogspot.com/2010/07/ooh-ahh-google-images-presents-nicer.html (2010)

[19] Shiels, M.: Google images top 1bn page views. http://www.bbc.co.uk/news/technology-10693439 (2010)

[20] Günther Gedigaa, Kai-Christoph Hamborg, I.D.: The IsoMetrics usability inventory: An operationalization of ISO 9241-10 supporting summative and formative evaluation of software systems. Behaviour and Information Technology **18** (1999) 151–164