Universität Hamburg
MIN-Fakultät
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Bachelorarbeit

Barrierefreiheit im Internet

Werkzeuggestützte Analyse von Barrieren auf Webseiten

Maximilian Blochberger
⟨9blochbe@informatik.uni-hamburg.de⟩

Matrikelnummer 6132298

Studiengang Software-System-Entwicklung

29. Oktober 2013

Erstgutachter: Dr. Guido Gryczan
Zweitgutachter: Dr. Axel Schmolitzky

Inhaltsverzeichnis

1 Einleitung

Barrieren, ganz Allgemein, hindern eine Person an der Erreichung eines Ziels. Einige *Barrieren* kann man umgehen, falls man die Möglichkeit dazu hat. Sollte man jedoch, zum Beispiel aufgrund einer Behinderung, eingeschränkt in der Wahrnehmung solcher Möglichkeiten sein, wird es schwieriger, sein Ziel zu erreichen.

1.1 Ziel

Diese Arbeit soll dabei helfen zu verstehen, was *Barrieren* von *Webauftritten* sein können und welche Auswirkungen sie möglicherweise haben. Ferner soll im Rahmen dieser Arbeit ein Analyse-Werkzeug entwickelt werden. Dieses soll dazu dienen, *Barrieren* von *Webauftritten* zu erkennen und somit Web-Entwicklern die Möglichkeit geben, diese zu entfernen. So können Inhalte möglichst gleichberechtigt für alle zugänglich gemacht werden.

Außerdem soll das Analyse-Werkzeug menschliche Prüfer bei der Auswertung unterstützen. Dies gilt insbesondere für *Barrieren*, die schlecht automatisch ermittelt werden können.

1.2 Aufbau

Zunächst werden die wichtigen Grundlagen erläutert. Erstens soll dort dem Leser das Grundwissen vermittelt werden, welches für nachfolgende Abschnitte benötigt wird. Zweitens werden die in dieser Arbeit verwendeten Begriffe eingeführt(siehe Abschnitt 2, Seite 6).

Danach wird *Barrierefreiheit* genauer untersucht und beispielhaft einzelne *Barrieren* genauer betrachtet (siehe Abschnitt 3, Seite 14). Zu jeder *Barriere* wird auch erläutert, warum sie problematisch ist (siehe Abschnitt 3.3, Seite 21).

Der Funktionsumfang des im Rahmen dieser Arbeit entwickelten Analyse-Werkzeuges wird dann vorgestellt und die technischen Details erklärt (siehe Abschnitt 4, Seite 26)

Am Ende folgt eine Zusammenfassung, zu den in der Arbeit betrachteten Themen (siehe Abschnitt 6, Seite 37) und ein kurzer Ausblick (siehe Abschnitt 7, Seite 38).

1.3 Motivation

Das Internet spielt in der heutigen Zeit eine wichtige Rolle, es dient beispielsweise als Informationsquelle, als Medium der Kommunikation und Unterhaltung. Keinem Menschen sollte dieses Mittel vorenthalten werden.

Ich arbeite neben dem Studium bei der *froglogic GmbH*[1]. Dort beschäftige ich mich hauptsächlich mit dem Testen von *Webauftritten*; speziell mit dem Testen der Darstellung von *Webseiten*.

Es wurde von Kunden angefragt, ob die Firma zukünftig auch Produkte mit Möglichkeiten zum Testen von *Benutzerfreundlichkeit* und *Barrierefreiheit* anbieten könnte. Mir wurde angeboten, dieses Thema als Bachelorarbeit zu behandeln.

[1] http://www.froglogic.com, Letzter Zugriff: 27. September 2013

2 Grundlagen

2.1 Barrierefreiheit

Der Begriff der *Barrierefreiheit* kommt aus dem Bauwesen. [Duden (2013)] Er besagt, dass ein Gebäude als *barrierefrei* gilt, wenn es trotz eventueller Beeinträchtigungen von Menschen ohne Probleme für diese zugänglich ist. So muss zum Beispiel als Alternative für eine Treppe ein rollstuhlgerechter Zugang vorhanden sein, denn "*niemand darf wegen seiner Behinderung benachteiligt werden.*" [Grundgesetz, Artikel 3, Absatz 3]

Menschen können durch Behinderungen eingeschränkt sein. Beeinträchtigungen können jedoch auch, besonders im höheren Alter, beispielsweise durch verzögerte Reaktionen oder schlechtere Sicht entstehen.

2.1.1 Barrierefreiheit im Internet

Barrierefreiheit im Internet kann man sich ähnlich vorstellen wie die *Barrierefreiheit* aus dem Bauwesen. Hierbei handelt es sich allerdings nicht um den Zugang zu einem Gebäude, sondern um den Zugang zu Information, Kommunikation und Unterhaltung.

Kurz gesagt: *Barrierefreiheit* ist die Möglichkeit für den *Seitenbesucher*, sein Ziel selbstständig zu erreichen – auch wenn dieser eine Behinderung hat.

Die *Barrieren*, die in dieser Arbeit vorgestellt werden, werden vordergründig durch Behinderungen thematisiert. Der Begriff der *Barrierefreiheit* beschränkt sich hierbei jedoch nicht nur auf die Behinderung an sich, sondern auf allgemeine Beeinträchtigungen. Da eine Suchmaschine beispielsweise keine Möglichkeit hat, zu erkennen, was sich auf einem Bild befindet, ist sie wie ein Mensch mit starker Sichtbeeinträchtigung auf Alternativtexte angewiesen. [Kent u. Bäckmann (2006)]

Wenn im Folgenden von *Barrierefreiheit* die Rede ist, ist damit die *Barrierefreiheit* im Internet gemeint.

2.1.2 Gesetzliche Regelungen

"*Träger der öffentlichen Gewalt*" sowie "*gewerbsmäßige Anbieter*" sind dazu verpflichtet, *Webauftritte barrierefrei* zu gestalten. [Behindertengleichstellungsgesetz, §11 Absatz 1 – 2]

In einzelnen Bundesländern wurden "*Verordnungen zur Verwendung von Gebärdensprache und anderen Kommunikationshilfen, sowie Verordnungen über* Barrierefreie *Dokumente in Verwaltungsverfahren erlassen.*" [Hellbusch u. Probiesch (2011), Seite 46] Dies betrifft nicht nur *Webseiten*, sondern beispielsweise auch Portable Document Formats (PDFs) und andere Dokumente.

In der Barrierefreie-Informationstechnik-Verordnung (BITV 2.0) erkennt man, dass sich die gesetzlichen Regelungen stark an den Web Content Accessibility Guidelines (WCAG) orientieren. [Barrierefreie-Informationstechnik-Verordnung, Anlage I (zu §3 & §4 Absatz 1)]

Andere Länder beziehen sich in der Regel auch auf die WCAG. Einige Länder tragen teilweise

sogar zur Verbesserung des WCAG-Standards bei. [Brewer u. a. (2006)]

2.2 Barrieren

Barrieren hindern eine Person an der Erreichung eines Ziels. Das Auffinden von Informationen, mit anderen Menschen zu kommunizieren oder sich zu unterhalten, können solche Ziele beim Surfen im Internet sein.

Was genau kann jemandem an der Erreichung seines Ziels im Internet hindern?

Es gibt einerseits technische und andererseits nicht-technische *Barrieren*. Die Standards konzentrieren sich überwiegend auf die technischen *Barrieren*, da man diese durch automatische Tests erkennen kann. Dies ermöglicht das Erstellen von Softwarelösungen zum Validieren der *Webseiten*.

Technische *Barrieren* können durch technische Hilfsmittel wie *Webseiten*-Validatoren erkannt werden (siehe Abschnitt 3.2.1, Seite 18). Die Existenz von Alternativtexten zu Bildern kann beispielsweise automatisiert überprüft werden.

Bei nicht-technischen *Barrieren* hingegen benötigt man in der Regel menschliche Hilfe, um sie zu erkennen (siehe Abschnitt 3.2.2, Seite 20). Solche *Barrieren* können möglicherweise auch durch Einsatz von *Heuristiken* erkannt werden. Ob Alternativtexte angemessen Bilder ersetzen können, kann nicht durch technische Hilfsmittel entschieden werden und bedarf daher die Beurteilung durch Menschen.

2.3 Benutzerfreundlichkeit

Als *benutzerfreundlich* zählt eine *Webseite* für einen *Nutzer*, wenn er es als "angemessen" erachtet, wie er sein Ziel erreicht. Angemessenheit heißt an dieser Stelle beispielsweise, dass der Weg zum Ziel für ihn intuitiv ist. Auch andere Empfindungen des *Nutzers* spielen hier eine Rolle, wie zum Beispiel der optische Eindruck, die Verständlichkeit der Texte und vieles mehr.

Wenn man sein Ziel nicht erreichen kann, gilt die *Webseite* als nicht *barrierefrei* – demzufolge ist sie auch nicht *benutzerfreundlich*.

In dieser Arbeit wird versucht, den Mehrwert hervorzuheben, der durch *Barrierefreiheit* für *Benutzerfreundlichkeit* entsteht. Mehr zu dem Thema findet sich vor allem in Abschnitt 3.1.2 auf Seite 15.

2.4 Web-Techniken

2.4.1 XML

Extensible Markup Language (XML) ist eine Sprache zur Strukturierung von Dokumenten. XML-Dokumente können sowohl von Menschen, als auch von Maschinen gelesen werden.

Vereinfacht gesagt besteht ein Dokument im XML-Format aus Tags und Attributen. Attribute werden Tags zugeordnet und können Werte haben. Tags können weitere Tags oder Text-

Elemente umschliessen. So kann man zum Beispiel einen Hyperlink (“Anker”) wie in Listing 1 darstellen.

```
<a href="http://www.informatik.uni-hamburg.de">
  Uni Hamburg, Fachbereich Informatik
</a>
```

Listing 1: XML/HTML Beispiel: Link zur Fachbereichsseite

`<a>` ist ein Tag und `href` ist ein Attribut mit einem Uniform Resource Identifier (URI) als Wert.

In diesem Text werden folgende Konventionen verwendet, zum Beispiel:

- `<img>`, `<a>`-Tag
- `alt`, `href`-Attribut

Die hier aufgelisteten XML-Grundlagen dienen dem Verständnis dieser Arbeit und sind deshalb weder ausführlich noch komplett.

2.4.2 HTML

Hypertext Markup Language (HTML) ist ein Ableger von XML. Sie ist eine Auszeichnungssprache mit der man Inhalte strukturiert darstellen kann.

In dieser Arbeit wird hauptsächlich HTML von Relevanz sein. Das Beispiel in Listing 1 ist sowohl ein HTML-, als auch ein XML-Quelltextauszug.

Semantisches HTML

Als semantisches HTML bezeichnet man die korrekte Zuordnung semantischer Rollen von Text-Inhalten. Zum Beispiel sollte man eine Überschrift mit den `<h1>`-`<h6>`-Tags kennzeichnen. Wenn Überschriften semantisch als solche ausgezeichnet wurden (siehe Listing 2), kann ein *Screenreader* diese auch als solche vorlesen. Ist eine Überschrift nur optisch hervorgehoben (siehe Listing 3), wird ein *Screenreader* diese als normalen Fließtext vorlesen. Ebenso sollte man Elemente, die keine Überschriften sind, nicht als solche auszeichnen. Der Stil des `<h1>`-Tags kann beispielsweise so verändert werden, dass er optisch nicht vom Fließtext zu unterscheiden ist.

```
<h1>Uni Hamburg</h1>
```

Listing 2: HTML Beispiel: Semantisch ausgezeichnete Überschrift

```
<div style="font-size: 2em; font-weight: bold;">
  Uni Hamburg
</div>
```

Listing 3: HTML Beispiel: Optisches Äquivalent der Überschrift aus Listing 2[2]

Des Weiteren gibt es Auszeichnungssprachen wie zum Beispiel die Web Ontology Language (OWL), mit denen man Inhalte mit weiterer Semantik über die Rolle im Text hinaus anreichern

[2] Dies gilt für *Safari* (Version 6.0.5, *OS X*). Ergebnisse anderer Browser können abweichen.

kann. Um bei Leseschwierigkeiten oder Auffindung von semantischen Zusammenhängen zu helfen, kann man zum Beispiel das Wort “Erdbeere” auch als Frucht und essbar kennzeichnen. Dies ist jedoch nicht Bestandteil dieser Arbeit und nur der Vollständigkeit halber erwähnt.

HTML5
HTML5[3] ist der, sich noch in Entwicklung befindende, neue HTML-Standard. Er beinhaltet unter anderem neue semantische Elemente, mit denen man Inhalte besser auszeichnen kann. So wird es beispielsweise ein `<navigation>`-Tag geben, der ganze Navigationsbereiche kennzeichnet. Der neue `<progress>`-Tag zeichnet eine Fortschrittsanzeige aus, die besonders bei dynamischen Inhalten (siehe Abschnitt 2.4.5, Seite 10) wichtig sein kann. Mit dem `type='date'`-Attribut für einen `<input>`-Tag zeichnet man ein Eingabeelement eines Formulars aus, welches für Datumseingaben geeignet ist.

Viele HTML5 Elemente werden bereits von den meisten Web-Browsern unterstützt und zunehmend im Internet verwendet.

2.4.3 DOM

Mit dem Document Object Model (DOM) kann man ein XML- beziehungsweise HTML-Dokument modellieren. DOM dient als Schnittstelle zum Lesen und Verändern des Dokuments.

Da XML-Dokumente aus Tags bestehen, die weitere Tags oder Text-Elemente umschliessen können, hat ein DOM eine Baumstruktur.

Für einen Beispiel-DOM-Baum siehe Abbildung 1. In dem Beispiel wurden auf die Attribute der Tags der Übersichtlichkeit halber verzichtet.

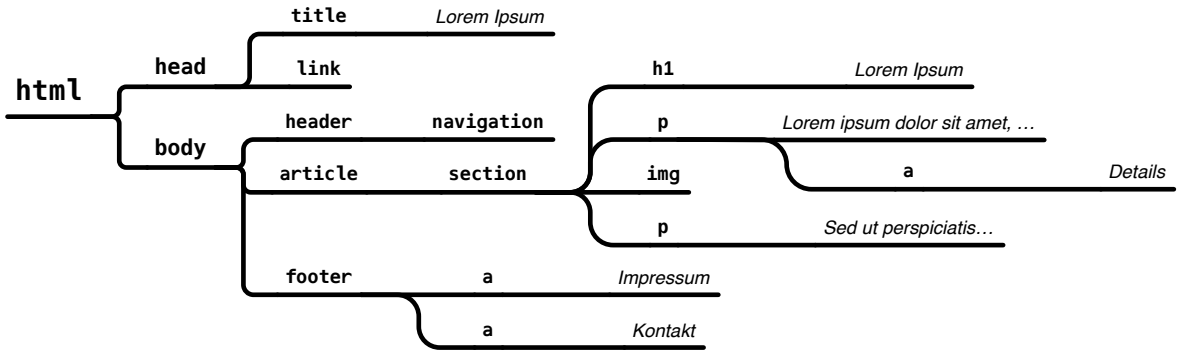

Abbildung 1: Beispiel: DOM-Baum

2.4.4 CSS

Cascading Style Sheets (CSS) ist eine Sprache, mit der man die Optik von beispielsweise HTML-Elementen anpassen kann. Dazu zählt vor allem auch das Layout. Der Wert des `style`-

[3] http://www.w3.org/TR/html5/, Letzter Zugriff: 27. September 2013

Attributs in Listing 3 ist beispielsweise CSS, welches den `<div>`-Tag wie eine Überschrift darstellt.

CSS sollte man immer als externes Dokument einbinden (siehe Listing 4) und nicht, wie in Listing 3, als Wert eines `style`-Attributs. Das erhöht die Austauschbarkeit von Designs. Damit kann man beispielsweise einen Stil mit höheren Kontrasten für ältere Menschen, die nicht mehr so gut sehen können, bereitstellen. Ein höherer Kontrast hilft dabei Umrisse besser zu erkennen, was älteren Leuten schwieriger fällt und diese möglicherweise daran hindert Text zu erkennen. Ebenso kann man Stile für mobile Geräte definieren – sodass deren Nutzer davon profitieren.

```
<link rel="stylesheet" type="text/css" href="style/highcontrast.css" />
```

Listing 4: HTML Beispiel: Externes Einbinden von CSS

2.4.5 Dynamische Inhalte

Dokumente, die nur aus HTML und CSS bestehen, sind statisch. Auf den meisten *Webauftritten* kann man allerdings mit den Dokumenten interagieren: Man kann sie verändern, etwas bewegen, eigene Inhalte hinzufügen oder entfernen und vieles mehr.

Dynamische Inhalte stellen in der Regel potentielle *Barrieren* dar. Das heißt nicht, dass man mit dynamischen Inhalten keine *Barrierefreiheit* erzielen kann. Im Gegenteil – man kann *Barrierefreiheit* mit dynamischen Inhalten sogar fördern. Dies ist jedoch nicht Bestandteil dieser Arbeit und hier nur erwähnt, um Missverständnissen vorzubeugen. Wichtig beim Einsatz dynamischer Inhalte ist, dass man sich bereits vorher Gedanken um die *Barrierefreiheit* macht, da ein nachträgliches Ändern mit viel Arbeit verbunden ist.

JavaScript

JavaScript ist eine Skript-Sprache. Im Web wird sie hauptsächlich dafür verwendet, das DOM und somit den Inhalt einer *Webseite* zu verändern. So kann zum Beispiel auf Nutzereingaben reagiert werden. Sie wird in der Regel auf dem Rechner des *Seitenbesuchers* vom Web-Browser interpretiert.

Wichtig für die *Barrierefreiheit* ist das Verwenden vom sogenannten unaufdringlichen (eng. *unobtrusive*) *JavaScript*. Unaufdringliches *JavaScript* besagt, dass *JavaScript* stets als optionale Ergänzung betrachtet wird. Das heißt, dass nur DOM-Operationen zum Verändern des Dokuments verwendet werden; dass *Nutzer* nicht auf bestimmte Eingabegeräte angewiesen sind und dass das *JavaScript* extern eingebunden wird. [Langridge (2005)]

Externes Einbinden kann ähnlich wie bei CSS geschehen (siehe Listing 5, Seite 10).

```
<script type="text/javascript" src="lib/js/effects.js"></script>
```

Listing 5: HTML Beispiel: Externes Einbinden von *JavaScript*

Um *JavaScript barrierefrei* zu gestalten, kann man sich an Accessible Rich Internet Applications (ARIA) bedienen. Dies wird in der Arbeit jedoch nicht behandelt und ist hier nur der Vollständigkeit halber aufgeführt.

Flash
Neben *JavaScript* wird oft auch Flash verwendet, um Inhalte von Webseiten dynamisch zu gestalten. Flash kann man ebenso *barrierefrei* gestalten, allerdings muss ein sogenannter Flash-Player auf dem System installiert sein. Ohne einen Flash-Player wird das System des *Nutzers* nicht in der Lage sein, die Inhalte anzuzeigen. Da insbesondere mobile Geräte keinen solchen Flash-Player installiert haben, führt es oft dazu, dass Nutzer sich die Seiten nicht ansehen können und ihnen somit Informationen vorenthalten werden.

2.4.6 Zusammenfassung

Das externe Einbinden von CSS und *JavaScript* hat sich weitestgehend als gutes Design (eng. *best practice*) für *Webseiten* durchgesetzt (siehe Abbildung 2).

Es trennt Inhalt von Darstellung und Funktionalität. So kann man den gleichen Inhalt ohne größeren Aufwand verschieden darstellen und mit verschiedenen Funktionen anbieten, was ein gutes Fundament für die *Barrierefreiheit* ist.

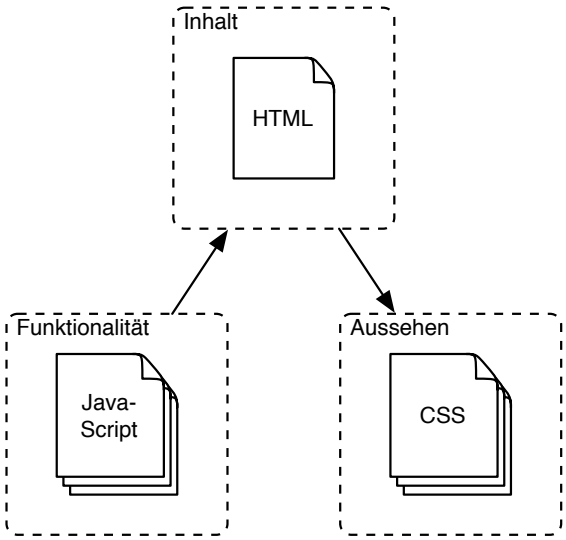

Abbildung 2: *Webseiten*-Design

Das *Webseiten*-Design erinnert an das, in der Software-Entwicklung bekannte, Model-View-Controller Entwurfsmuster[4]. Der Inhalt wäre in diesem Sinne das Modell und die Funktionalität der Controller. Der Web-Browser wäre dann ein View, der die Darstellungsbeschreibung aus den Nutzereinstellungen oder den CSS-Dateien liest.

Die Wiederverwendbarkeit durch dieses Vorgehen ist ein weiterer wichtiger Punkt für Web-Entwickler. Einerseits kann man für eine *Seite* verschiedene Versionen für unterschiedliche Sprachen und andererseits verschiedene Inhalte, zum Beispiel weitere *Seiten*, bereitstellen und sowohl Funktionalität, als auch Aussehen beibehalten. Verschiedene CSS-Dateien können für verschiedene Geräte oder Browser zur Verfügung gestellt werden – zum Beispiel für Smartphones, Beamer oder sogar Drucker. Ferner können *Nutzer* das Design auf ihre eigenen Bedürfnisse anpassen.

Außerdem macht es *Webseiten* auch performanter, da der Web-Browser beim Laden mehrerer HTML-Dateien nicht jedes Mal die CSS,- oder *JavaScript*-Dateien neu laden muss.[5]

Die HTML-Dateien selbst sollten linearisierbar, strukturiert und frei von Darstellung und Funktionalität sein. Linearisierbar heißt, dass die Seite auch in linearer Form den gleichen Inhalt vermittelt, zum Beispiel wenn man die Seite ohne jegliche Layoutinformationen be-

[4] Vgl. Gamma u. a. (2009), Seite 5 ff.
[5] Vgl. http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html, Letzter Zugriff: 27. September 2013

trachtet. Strukturiert heißt hier, dass die Seite so gut wie möglich semantisch angereichert sein sollte.

2.5 C/C++ & Qt

Das Analyse-Werkzeug, welches im Rahmen dieser Arbeit erstellt wurde, ist mit C/C++ und dem *Qt Framework* von *Digia*[6] (im Folgendem *Qt*) geschrieben.

Grundkenntnisse der Programmiersprache C/C++ werden hier nicht weiter eingeführt. Diese können zum Verständnis des Quelltextes des Analyse-Werkzeuges erforderlich sein, allerdings sind sie für das Verständnis dieser Arbeit nicht notwendig.

Grundkenntnisse der objektorientierten Programmierung werden vorausgesetzt.

2.5.1 Qt

Grundkenntnisse des *Qt Frameworks* sind für das Verständnis dieser Arbeit ebenfalls kaum erforderlich. Verständnishilfen sowie eine ausführliche Dokumentation des *Qt Frameworks* kann man im *Qt Project*[7] finden. Die Mechanismen von *Qt*, die zum Verständnis des objektorientierten Entwurfs erforderlich sind, werden in diesem Abschnitt beschrieben.

Signale & Slots

Der *Signals & Slots* Mechanismus wird für die Kommunikation zwischen Objekten verwendet. [Qt Project (2013), Signals & Slots[8]]

Ein Objekt kann Signale senden, zum Beispiel wenn sich der Zustand des Objektes geändert hat. Andere Objekte können diese Signale empfangen und weitersenden oder auf die Veränderung des Zustandes durch Ausführen einer Methode reagieren. Eine solche Methode nennt man Slot. Signale können auch Parameter an Slots übergeben. Ein Objekt, welches Signale sendet, kann diese auch selbst empfangen. Um auf Signale mit Slots oder dem Senden weiterer Signale reagieren zu können, muss man diese verbinden.

Der Mechanismus ist mit dem Beobachter-Entwurfsmuster zu vergleichen. Beobachter muss man bei einem Subjekt registrieren, ähnlich wie man Signale und Slots bzw. andere Signale miteinander verbinden muss. Das Subjekt ruft dann, zum Beispiel bei einer Zustandsänderung, eine Methode an den Beobachter auf, was dem Senden eines Signals entspricht. Dies kann ebenfalls mit Parametern geschehen. [Gamma u. a. (2009), Seite 287 ff.]

Im Gegensatz zum Beobachter-Muster besteht die Abhängigkeit zwischen Signalen und Slots erst zur Laufzeit. Ein Nachteil davon ist, dass man Fehler, zum Beispiel in der Signatur der Methoden, nicht in der Übersetzungszeit feststellen kann. Ein Vorteil ist, dass keine direkte rekursive Abhängigkeit mehr besteht, wenn ein Objekt die Rolle als Subjekt und Beobachter zugleich einnimmt. Ein weiterer Vorteil ist, dass man Signale durchreichen kann, ohne explizit

[6] http://qt.digia.com, Letzter Zugriff: 22. September 2013
[7] http://www.qt-project.de, Letzter Zugriff: 22. September 2013
[8] http://qt-project.org/doc/qt-5.1/qtcore/signalsandslots.html, Letzter Zugriff: 22. September 2013

den Aufruf an allen registrierten Beobachtern manuell durchzuführen. Außerdem kann man die Parameter auch ignorieren, falls diese für eine Reaktion nicht erforderlich oder bereits anderweitig vorhanden sind.

Signale werden in dieser Arbeit in Diagrammen als gepunktete Pfeile dargestellt. Die Pfeilspitze zeigt in Richtung des Signal-Empfängers (siehe Abbildung 3).

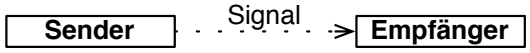

Abbildung 3: Darstellung von Signalen

3 Analyse

3.1 Umsetzung von Barrierefreiheit

3.1.1 Probleme

Viele Unternehmen schenken der *Barrierefreiheit* ihrer *Webauftritte* keine große Beachtung.

Oft wird damit argumentiert, dass behinderte Menschen nicht zur Zielgruppe gehören. Unterstützt wird dieses Argument auch dadurch, dass sich kaum jemand darüber beschwert, dass etwas auf einer *Webseite* nicht *barrierefrei* ist. Eventuell auftretende Probleme zu melden kann allerdings schon daran scheitern, dass der Weg sie zu melden ebenfalls unüberwindbare *Barrieren* haben kann. Wenn Menschen mit Behinderungen nicht zur Zielgruppe gehören oder nur einen sehr kleinen Teil ausmachen, wird es meist als ökonomisch nicht sinnvoll eingeschätzt, sich um *Barrierefreiheit* zu kümmern.

Die meisten großen Firmen haben bereits *Webauftritte*. Diese nachträglich *barrierefrei* zu gestalten ist mit vielen Problemen verbunden und schwieriger, als *Barrierefreiheit* schon beim Design der *Webpräsenz* zu beachten. Das betrifft insbesondere dynamische Inhalte.

Dass sich viele Web-Entwickler gar nicht erst mit dem Thema *Barrierefreiheit* auseinandergesetzt haben, ist auch ein großes Problem. Sie kennen zwar die HTML Spezifikation des World Wide Web Consortium (W3C), die sie auch größtenteils einhalten, um Kompatibilität zu verschiedenen Web-Browsern und Endgeräten zu wahren. Oft kennen sie jedoch nicht die Spezifikationen, welche die *Barrierefreiheit* betreffen oder der Aufwand, die WCAG einzuhalten wird als zu hoch eingeschätzt. Einige Entwickler fühlen sich auch in ihrer Gestaltungsfreiheit eingeschränkt, wenn sie sich an die vorgegebenen Standards halten müssen. Andere denken, dass *Barrierefreiheit* bereits umgesetzt wird, da sie sich an die Standards halten. *Barrierefreiheit* ist allerdings weit mehr als nur die Einhaltung der Standards (siehe Abschnitt 3.2.2, Seite 20).

Fertige *Webseiten* werden oft nicht ausreichend auf mögliche *Barrieren* getestet. Die Validation von *Webseiten* wird teilweise sogar ausgetrickst. Ein bekannter Trick ist es, ein leeres `alt`-Attribut zu einem `<img>`-Tag anzugeben, da viele Analyse-Werkzeuge nur prüfen, ob das Attribut existiert, nicht aber ob es Text enthält. Ein leeres `alt`-Attribut wird in der Regel für dekorative Bilder verwendet, also Bildern, die nur für die Darstellung der *Webseite* dienen und keinen inhaltlichen Beitrag leisten. Besser wäre es, dekorative Bilder mithilfe von CSS einzubinden und jedem `<img>`-Tag einen aussagekräftigen Alternativtext zuzuweisen, da diese sonst nicht von einem *Screenreader* vorgelesen werden können. Da sich technisch nicht feststellen lässt, ob ein Bild dekorativen Charakter hat, müssen menschliche Prüfer hinzugezogen werden.

Viele *Webauftritte* sind auf bestimmte Größen fixiert, zum Beispiel 800×600. Eine Änderung dieser Größe kann dazu führen, dass man alle Inhalte, die Teil der *Webauftritt* sind überprüfen und eventuell sogar anpassen muss.

Viele Unternehmen wissen nicht, dass es gesetzliche Grundlagen für *Barrierefreiheit* gibt (siehe Abschnitte 2.1.2, Seite 6). Es gibt einige Fälle, in denen Unternehmen dafür angeklagt

wurden, dass ihre *Webpräsenz* nicht *barrierefrei* war[9]. Als Präzedenzfall gilt der *Webauftritt* der Olympischen Spiele Sydney 2000[10]. In der Regel befürchten die meisten Unternehmen nicht, dass sie dafür belangt werden, wenn ihre *Webauftritte* nicht *barrierefrei* sind, da es in Deutschland nur sehr wenig Medienaufmerksamkeit für solche Fälle gibt.

[Hellbusch u. Probiesch (2011)]

3.1.2 Wer profitiert denn davon?

Wie in Abschnitt 3.1.1 erwähnt, vermuten Unternehmen, dass "behinderte" Menschen oft nicht zu ihrer Zielgruppe gehören. Die Zugänglichkeit für Menschen mit Behinderungen ist nicht das einzige, was man durch *Barrierefreiheit* erreichen kann.

In dieser Arbeit werden deshalb nicht nur Menschen mit Behinderungen, sondern auch Menschen mit Beeinträchtigungen thematisiert.

Webauftritte sind finanziell wichtig für Unternehmen. Sie spielen eine wichtige Rolle in der Kundengewinnung und man kann auf ihnen meist sogar Käufe abwickeln. Ein Großteil der Internet-Nutzer mit nicht zu unterschätzender Kaufkraft sind ältere Menschen. Alte Menschen haben gehäuft Seh- und Hörprobleme, die sich auch auf die Nutzung einer *Webseite* auswirken. Zum Beispiel können höhere Kontraste bei einem älteren *Seitenbesucher* schon sehr viel bewirken. Das Vernachlässigen solcher Probleme kann bereits starke finanzielle Auswirkungen haben. Sollte beispielsweise ein Kunde Probleme damit haben, einen Kauf online abzuwickeln, so entscheidet dieser sich möglicherweise dafür, das Produkt letztendlich nicht zu erwerben.

Beeinträchtigungen müssen kein Dauerzustand sein, sie können auch kurzzeitig auftreten. Wenn eine Person zum Beispiel einen Unfall erlitten hat, muss sie womöglich durch einen gebrochenen Arm auf die Navigation mit einem Zeigegerät verzichten. Eine Beeinträchtigung muss nicht einmal eine Person direkt betreffen. Es wäre auch vorstellbar, dass die Batterie eines Zeigegerätes leer ist und man deshalb auf das Gerät verzichten muss. In beiden Fällen wäre der *Seitenbesucher* dann auf Tastaturnavigation angewiesen. Der zweite Fall zeigt, dass *Barrierefreiheit* und *Benutzerfreundlichkeit* oft das Gleiche bedeuten.

Software
Was ist, wenn es sich bei dem *Seitenbesucher* gar nicht erst um einen Menschen handelt? Das Scheitern von Software an *Barrieren* kann folgenschwere, insbesondere finanzielle, Auswirkungen haben.

Suchmaschinen durchsuchen regelmäßig automatisiert *Webauftritte*, um Informationen darüber zu erhalten, was man dort finden kann. Da die Datenflut im Internet unbeherrschbar ist, gibt es Relevanzbewertungen, um möglichst gute, beziehungsweise treffende, Suchresultate liefern zu können. Die meisten großen Suchmaschinen halten die genauen Relevanzkriterien ge-

[9] Vgl. Karl Grooves: List of Web Accessibility-Related Litigation and Settlements: http://www.karlgroves.com/2011/11/15/list-of-web-accessibility-related-litigation-and-settlements/, Letzter Zugriff: 27. September 2013

[10] W3C Web Accessibility Initiative (WAI): A Cautionary Tale of Inaccessibility: Sydney Olympics Website: http://www.w3.org/WAI/bcase/socog-case-study, Letzter Zugriff: 27. September 2013

heim, bei vielen ist man sich in der Search Engine Optimization (SEO) einig, dass sie zutreffen. Sollte sich zum Beispiel ein gesuchter Begriff im Seitentitel (`<title>`-Tag) oder in Überschriften wiederfinden, dann ist die *Webseite* wahrscheinlich relevanter, als wenn der Suchbegriff nur im Text steht. Eine hohe Relevanz ist selbstverständlich wichtig für die Kundengewinnung. Ebenso können blinde Menschen die Informationen aus Seitentitel und Überschriften verwenden um herauszufinden, an welcher Stelle die Information steht, die sie interessiert.

Viele *Seitenbesucher* gelangen über eine Suchmaschine auf eine *Webseite* – auch wenn sie bereits aktive Kunden des betreibenden Unternehmens sind. Eine wichtige Rolle spielt hierbei die Suche nach Produkten. Falls die *Webpräsenz* des Unternehmens nicht oder an einer schlechten Position, als Ergebnis einer Suche auftaucht, hat dies negative finanzielle Auswirkungen für das Unternehmen. Wenn die Indizierung der Suchmaschine durch *Barrieren* erschwert oder verhindert wird, kann dies den Zuwachs potentieller Neukunden verringern und sogar bestehende Kundenverhältnisse gefährden.

[Kent u. Bäckmann (2006)]

Web-Browser können durch die einfache Einhaltung von Standards auf *Webseiten* Funktionen bereitstellen, von denen jeder profitieren kann. Zum Beispiel kann man sich mit einer Erweiterung[11] für *Opera* die Gliederung einer *Webseite* anzeigen lassen (siehe Abbildung 4).

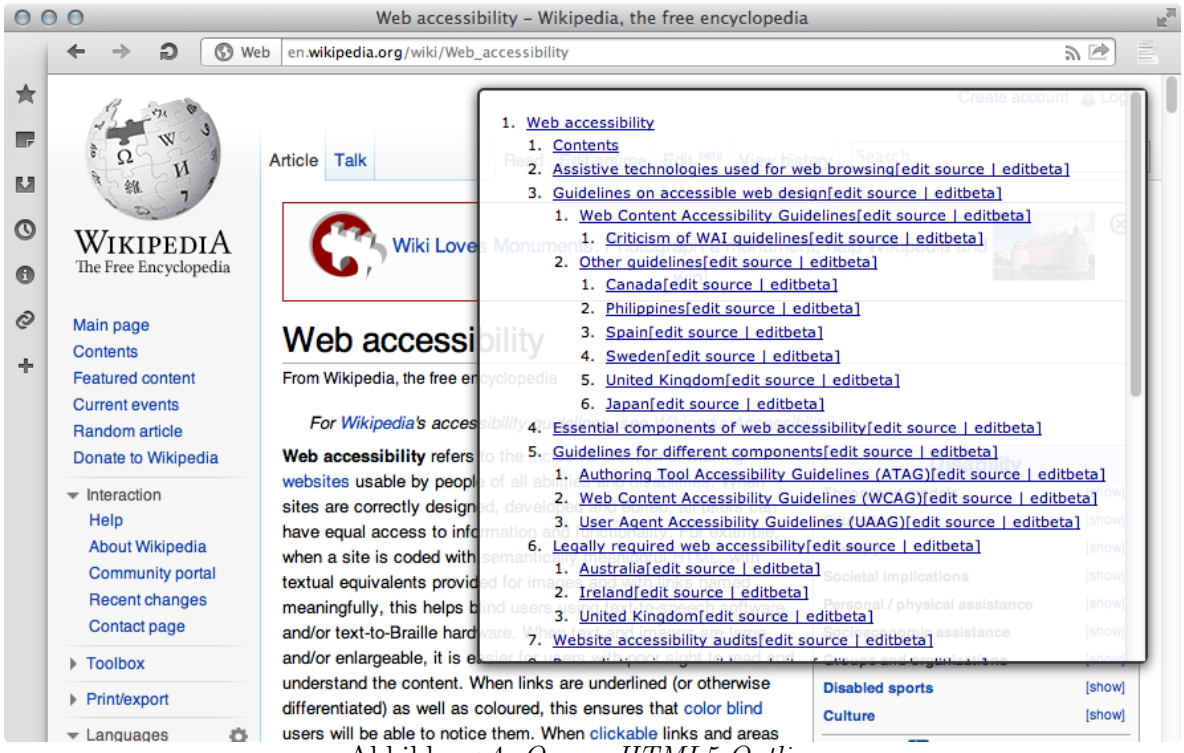

Abbildung 4: *Opera*: *HTML5 Outliner*

Mit *Safari* kann man sich eine *Webseite* ablenkungsfrei anzeigen lassen (siehe Abbildung 5). Dies ermöglicht nicht nur das Lesen in großen Buchstaben und mit hohem Kontrast, sondern

[11] *HTML5 Outliner* (*Opera* Erweiterung): https://addons.opera.com/en/extensions/details/html5-outliner/, Letzter Zugriff: 27. September 2013

bietet auch die Möglichkeit, die *Seite* in der Ansicht auszudrucken. Damit kann man Tinte sparen, indem unwichtige Dinge oder Farben nicht mitgedruckt und der Platz auf dem Papier optimal genutzt werden kann.

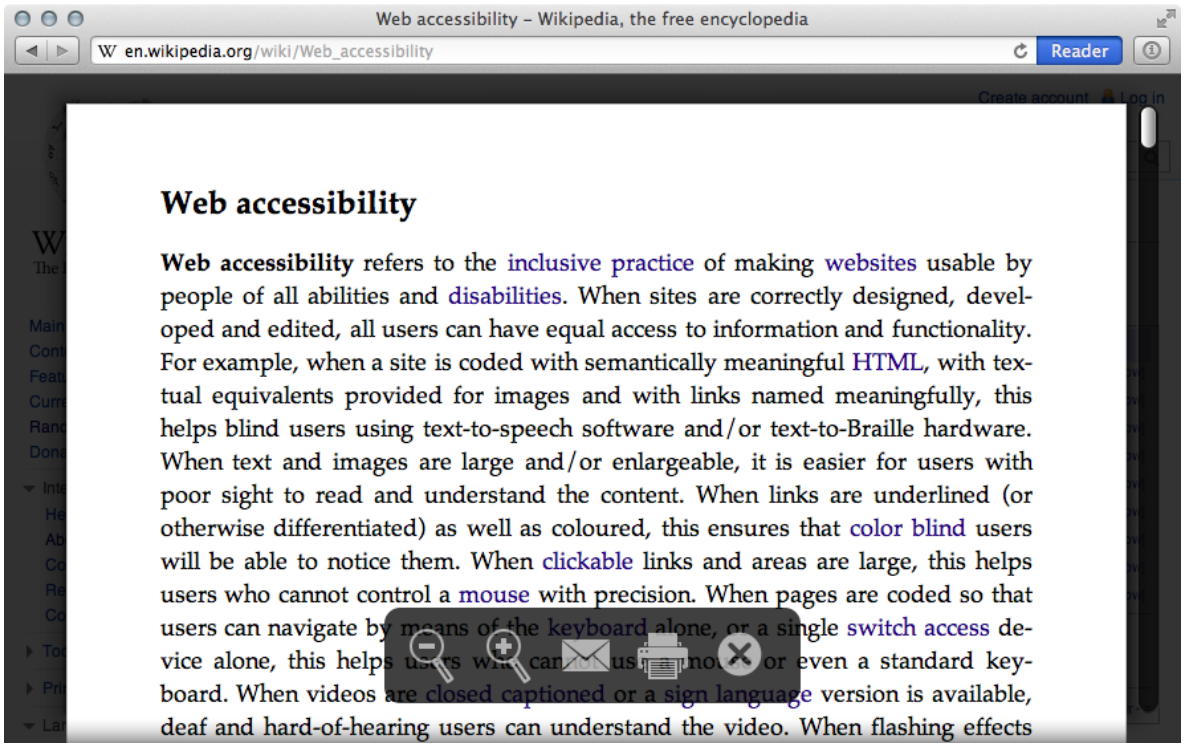

Abbildung 5: *Safari*: Reader

Aktuelle Entwicklungen

Besonders im Bereich der mobilen Geräte hat sich in letzter Zeit viel getan. Mit der Einführung von sogenannten Smartphones hat man eine herkömmliche Tastatureingabe durch Touch- und Gestensteuerung abgeschafft. In der Sprachsteuerung gibt es immer neue Fortschritte.

Was aber, wenn auf herkömmliche Eingabegeräte, wie Tastaturen oder Zeigegeräte verzichtet wird? Dann wird die *Barrierefreiheit* von *Webauftritten* daran gemessen, ob man die *Webseiten* mit einem Gerät bedienen kann. Die *Barrierefreiheit* der Geräte selbst wird hier nicht weiter thematisiert. Es sollte klar sein, dass sich Sprachsteuerung nicht ohne Weiteres von einer Person mit Sprachschwierigkeiten nutzen lässt.

Es folgen zwei Beispiele, die von *barrierefreien Webseiten* profitieren werden.

Google Glass[12] ist eine Brille mit einem kleinen Display, die mit Touch-Gesten oder Sprachsteuerung bedient werden kann. Die Brille hat einen Web-Browser[13], dessen Navigation durch gutes Design von *Webseiten* profitieren könnte (siehe Abbildung 2, Seite 11).

[12] *Google Glass*: http://www.google.de/glass/start/, Letzter Zugriff: 27. September 2013
[13] *Google Glass*: *XE7* release notes: https://support.google.com/glass/answer/3214744, Letzter Zugriff: 27. September 2013

Ein wünschenswertes Szenario wäre zum Beispiel, wenn man per Sprachsteuerung den Befehl geben könnte: "*ok glass, show section 'simple sharing'*" und der Browser dann direkt zu dem gewünschten Abschnitt "Simple sharing" springen würde.

***iOS* im Auto**[14] integriert *Apples* mobiles Betriebssystem in Autos. Die Besonderheit ist hierbei, dass alles ohne Hinsehen funktionieren soll, da Ablenkungen schwerwiegende Folgen haben können. Hier schlüpft der Fahrer, im Bezug auf die Nutzung des Gerätes, in die Rolle eines Blinden.

Ein denkbares Szenario wäre der Besuch einer *Webseite*. Da man die *Seite* nicht selbst lesen kann, müsste das System die Rolle eines *Screenreaders* übernehmen und Inhalte vorlesen.

Solche Entwicklungen werden die *Barrierefreiheit* im Internet positiv beeinflussen und die Einhaltung der Standards wichtiger machen. Allerdings werden neue Geräte auch neue Probleme für die *Barrierefreiheit* mit sich bringen. Durch innovative Bedienungsarten werden vermutlich neue Lösungen erfordert, um die Interaktion mit diesen Geräten *barrierefrei* zu gestalten – dies wird hier allerdings nicht näher betrachtet.

3.2 Klassifikation von Barrieren

3.2.1 Technische Barrieren

Technische *Barrieren* sind *Barrieren*, an denen technische Mittel scheitern, wie zum Beispiel Web-Browser, *Screenreader* oder andere. Sie können meistens automatisiert überprüft werden.

Es folgen zwei Beispiele technischer Barrieren.

Validität
HTML muss nicht immer valides XML sein. [15] So ist bei `<li>`-Tags beispielsweise der schließende Tag optional. [W3C (1999), 10.2 Unordered lists (`UL`), ordered lists (`OL`), and list items (`LI`)[16]] Die meisten modernen Browser korrigieren diese XML-Invalidität und erstellen einen validen DOM-Baum.

Angenommen eine *Webseite* hat eine Listenstruktur wie in Listing 6. Die meisten Browser können die *Seite* ohne Probleme anzeigen, da sie die Struktur reparieren, wie man am Beispiel von *Safari* in Abbildung 6 sehen kann. Auch eine Überprüfung mit den meisten HTML-Validatoren wird keine Warnung ausgeben. Ändert man nun den Dokumenttyp auf Extensible Hyptertext Markup Language (XHTML), dann schlägt zwar die Validierung fehl, aber die Seite wird weiterhin korrekt im Browser angezeigt.

Moderne Browser korrigieren sogar invalide HTML-Strukturen. In Listing 7 überschneidet sich ein `<strong>`-Tag mit den `<li>`-Tags, was dazu führt, dass der Quelltext invalid ist. Auch diese

[14] *iOS7* – Was is neu: *iOS* im Auto: http://www.apple.com/de/ios/whats-new/#carintegration, Letzter Zugriff: 27. September 2013
[15] Dies gilt nicht für XHTML, welches valides XML sein muss.
[16] http://www.w3.org/TR/1999/REC-html401-19991224/struct/lists.html#edef-LI, Letzter Zugriff: 27. September 2013

invalide Struktur wird korrigiert, wie man auch hier am Beispiel von Safari in Abbildung 7 sehen kann. Diese Fehler werden von Web-Entwicklern auf Anhieb oft nicht wahrgenommen und erst mit einem HTML-Validator bemerkt.

```
<ul>
  <li>Eintrag 1
  <li>Eintrag 2
  <li>Eintrag 3
</ul>
```

Listing 6: HTML Beispiel: Problematische Listenstruktur

```
<ul>
  <li><strong>Eintrag 1
  <li>Eintrag 2</strong>
  <li>Eintrag 3
</ul>
```

Listing 7: HTML Beispiel: Invalide Listenstruktur

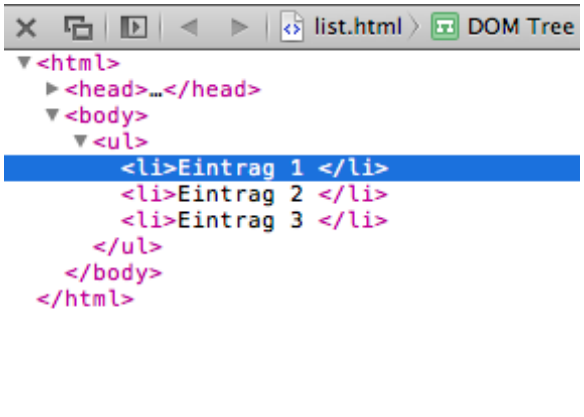

Abbildung 6: *Safari* korrigiert die problematische Struktur aus Listing 6

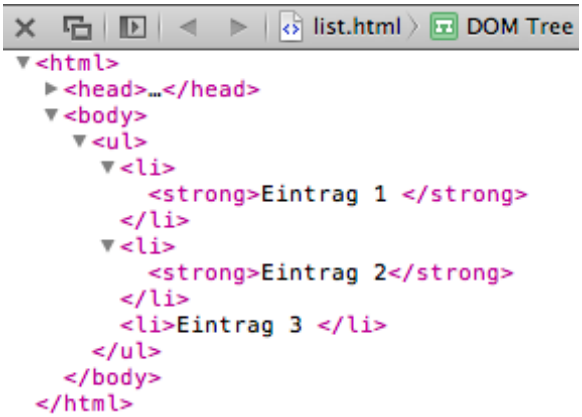

Abbildung 7: *Safari* korrigiert die invalide Struktur aus Listing 7

Solche Strukturen sollten vermieden werden, da sie potentiell problematisch sind. Es ist nicht sicher gestellt, dass sich Browser oder Browser-Versionen bei der Fehlerkorrektur gleich verhalten. Der Browser *links*[17] zeigt zum Beispiel nur den ersten Listeneintrag fett an. Falls ein Hilfsmittel, wie zum Beispiel ein *Screenreader*, nicht den korrigierten DOM-Baum eines Browsers verwendet, kann es zu Problemen kommen. Unter Umständen kann der Parser des Hilfsmittels den DOM-Baum gar nicht erst erstellen und somit ist keinerlei Information der *Webseite* für den *Seitenbesucher* erschließbar.

Es gibt viele weitere problematische Strukturen, die HTML und XML unterscheiden. Zum Beispiel sind in HTML Tag-Namen nicht abhängig von Groß- und Kleinschreibung. Sollte die Schreibweise von öffnenden und schließenden Tags unterschiedlich sein, kann es dazu führen, dass es bei Verwendung eines XML-Parsers zu Problemen kommt. Es hat sich durchgesetzt, Kleinbuchstaben für Tag-Namen zu verwenden.

Navigation

Es gibt verschiedene Arten von Navigation auf *Webseiten*. Einerseits kann man zwischen den einzelnen Elementen auf einer *Webseite* navigieren, andererseits kann man Links folgen und damit andere *Webseiten* besuchen.

[17] http://links.twibright.com, Letzter Zugriff: 27. September 2013

Die meisten Browser legen eine Chronik der besuchten *Seiten* an, die man als Browser-History bezeichnet. Man kann beispielsweise zurück zur letzten *Seite* springen.

Falls sich durch das Klicken auf einen Link ein Pop-Up-Fenster öffnet, so verliert man unter Umständen die Möglichkeit, in der Browser-History zurück zu gehen. Dies kann besonders für Personen vom Nachteil sein, die auf Tastatur-Navigation angewiesen sind. Pop-Ups blenden oft auch die Menü-Leiste aus, weswegen man die Pop-Ups möglicherweise über die gewohnten Navigationsmöglichkeiten nicht mehr schließen kann.

[Harper u. Yesilada (2008)]

3.2.2 Nicht-technische Barrieren

Nicht-technische *Barrieren* sind *Barrieren*, die nicht durch technische Hilfsmittel erkannt werden können.

Einige solcher *Barrieren* lassen sich mithilfe von *Heuristiken* ermitteln. Allerdings sind Ergebnisse, die durch *Heuristiken* gewonnen wurden unter Umständen falsch oder unvollständig und sollten durch einen menschlichen Prüfer verifiziert werden. Viele solcher *Barrieren* lassen sich allerdings gar nicht technisch feststellen.

"Barrierefreiheit *ist mehr als HTML und CSS*" [Hellbusch u. Probiesch (2011), Überschrift Abschnitt 1.4] – *Barrierefreiheit* ist weit mehr als das Einhalten der Standards. Im Folgenden sind einige nicht-technische *Barrieren* beispielhaft erläutert.

Verständlichkeit
Lern-, Aufmerksamkeits- oder Leseschwierigkeiten bieten einen großen Raum für mögliche *Barrieren*. Es ist also wichtig, Texte einfach und verständlich zu halten. Für Menschen mit Lese- und Schreibschwierigkeiten kann es hilfreich sein, den Text mit Symbolen anzureichern.

Sprache
Die Sprache der Texte sollte im Quelltext mit angegeben sein. *Screenreader* benötigen diese Information, um die korrekte Aussprache zu wählen. Wenn ein deutscher Text in englischer Aussprache vorgelesen wird, kann der Nutzer mit dem Resultat kaum etwas anfangen. Er muss gegebenenfalls die Aussprache selbst anpassen. [Hellbusch u. Probiesch (2011), Seite 419 ff.]

Chrome bietet beispielsweise die Möglichkeit an, eine fremdsprachige *Webseite* zu übersetzen. Auch wenn die Übersetzungsqualität zu wünschen übrig lässt, kann es helfen, bestimmte Informationen aus der *Webseite* zu erhalten.

CAPTCHAs
Ein Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) wird dazu verwendet, um Menschen von Programmen, sogenannten Bots, zu unterscheiden und somit eine *Webseite* vor Spam zu schützen.

Angenommen eine *Webseite* hält sich an alle Standards und es treten keine technischen Probleme auf, nun aber soll der Benutzer eine Frage beantworten, um zu zeigen, dass er ein Mensch

und kein Programm ist. Wenn dann nach der Farbe eines bekannten Logos gefragt wird, kann ein Mensch mit Sehbeeinträchtigung wie Blindheit oder Farbenblindheit diese Frage unter Umständen nicht beantworten. Auch Menschen ohne Sehbeeinträchtigung könnten dem Logo keine Aufmerksamkeit geschenkt haben und die Antwort somit nicht wissen.

Die gängigste Form von CAPTCHAs sind verzerrte Bilder von Worten oder Zeichenketten, die dann in ein Feld getippt werden müssen (siehe Abbildung 8). Es würde keinen Sinn ergeben, wenn man einem CAPTCHA-Bild den darauf sichtbaren Text als Alternativtext zuweisen würde. Bots könnten diesen Text verwenden, um den Test zu umgehen und der Test könnte ebenso weggelassen werden, da er für menschliche *Seitenbesucher* eine Unbequemlichkeit darstellt. Als Alternative zu CAPTCHA-Bildern gibt es deshalb oft Audio-CAPTCHAs.

Abbildung 8: Ein CAPTCHA-Bild, welches die Zeichenkette “smwm” zeigt[18]

Viele CAPTCHAs sind zeitlich begrenzt und können besonders für langsame oder Menschen mit motorischen Schwierigkeiten zur *Barriere* werden.

[Cunningham (2012), 1.4 Creating Accessible Sites]

3.3 Ausgewählte Barrieren

In diesem Abschnitt werden verschiedene *Barrieren* näher beleuchtet. Hier wird erklärt, warum diese *Barrieren* problematisch sind und was es zu beachten gibt. Außerdem wird erklärt, wie man diese *Barrieren* erkennt.

Es werden die verschiedenen Regeln des Programms erklärt. Jede Regel hat einen bestimmten Identifikator, der auch im Analyse-Werkzeug verwendet und angezeigt wird.

Regeln können verschiedene Resultate liefern. Sie können bestehen (eng. *pass*), wenn für die Regel kein Problem festgestellt wurde, sie schlagen fehl (eng. *fail*), wenn die Regel verletzt wurde oder sie können eine Warnung (eng. *warn*) ausgeben. Falls es zu einem technischen Problem während der Überprüfung kommen sollte, wird ein Fehler (eng. *error*) signalisiert.

3.3.1 Nicht-textuelle Inhalte

Für alle Inhalte, die nicht in maschinenlesbarer Form[19] vorliegen, muss eine Text-Alternative vorliegen. Es ist auch möglich, dass der *Besucher* durch eine Beeinträchtigung, wie zum Beispiel Sehschwäche oder Blindheit, die Anzeige nicht identifizieren kann. Ferner kann das Anzeigen der Inhalte durch Verbindungsprobleme verhindert oder durch den *Seitenbesucher* selbst deaktiviert werden.

[18] Bild-Quelle: http://commons.wikimedia.org/w/index.php?title=File:Captcha-smwm.svg&oldid=65707386, Letzter Zugriff: 27. September 2013

[19] “maschinenlesbare Form” bedeutet hier, dass es sich um eine Sequenz von Zeichen handelt. Texte in Bilddateien, typografische Bilder (ASCII-Art) und “Geheimsprachen” (Leetspeak) zählen nicht dazu.

Der Verlust der nicht-textuellen Inhalte bedeutet auch einen Verlust an Informationen. Deshalb sollte die Text-Alternative als Ersatz ausreichen, falls solche Inhalte nicht angezeigt werden können.

[W3C (2008), Guideline 1.1 Text Alternatives[20]]

Bilder
Repräsentativ für alle nicht-textuellen Inhalte werden in dieser Arbeit Bilder verwendet.

Bilder werden mit dem `<img>`-Tag in eine *Webseite* eingebunden (siehe Listing 8). Nach dem HTML-Standard müssen sowohl das `src`- als auch das `img`-Attribut vorhanden sein. Das `src`-Attribut muss einen validen URI als Wert zugewiesen haben, welches die Bild-Resource angibt. Der Wert des `alt`-Attribut enthält den Alternativtext für das Bild. [W3C (1999), 13.2 Including an image: the `IMG` element[21]]

```
<img src="images/logo.png" alt="Logo der Uni Hamburg" />
```

Listing 8: HTML Beispiel: `<img>`-Tag

Es gibt informative und dekorative Bilder. Für dekorative Bilder kann der Alternativtext leer sein. Bilder mit leerem Alternativtext werden von *Screenreadern* nicht vorgelesen. Wie in Abschnitt 2.4.6 auf Seite 11 erläutert, ist es ratsam dekorative Bilder mithilfe von CSS einzubinden.

Das Analyse-Werkzeug bietet 2 Regeln zu diesem Thema an.

Bild-Resourcen können mit der Regel `IMG-SRC` überprüft werden. Der Test schlägt fehl, sollte das `src`-Attribut nicht vorhanden sein oder keinen validen URI als Wert zugewiesen haben.

Der Mehrwert dieser Regel liegt in der Identifikation der Bilder. Sollte der Test des Alternativtextes fehlschlagen, so kann man den `<img>`-Tag mithilfe des `src`-Attributs besser wiederfinden und das Problem somit schneller beheben.

Alternativtexte müssen zu jedem Bild vorhanden sein und sollten es angemessen ersetzen können. Die Regel `IMG-ALT` schlägt fehl, wenn das `alt`-Attribut nicht vorhanden ist. Falls ein `alt`-Attribut angegeben wird, aber der Wert leer ist, resultiert die Regel in einer Warnung. Die Regel schlägt in diesem Fall nicht fehl, da es sich um ein Bild mit dekorativem Charakter handeln könnte.

Die Regel prüft nur die Existenz des Alternativtextes. Es kann nicht technisch ermittelt werden, ob der Alternativtext das Bild im Kontext der *Webseite* angemessen ersetzt. Dies kann nur ein menschlicher Prüfer entscheiden. Dafür bietet das Analyse-Werkzeug eine unterstützende Funktion an, welche Bilder und dazugehörige Alternativtexte tabellarisch auflistet (siehe Abbildung 11, Seite 27). So wird es dem menschlichen Prüfer erleichtert zu überprüfen, ob Alternativtexte angemessen sind.

[20] http://www.w3.org/TR/2008/REC-WCAG20-20081211/#text-equiv, Letzter Zugriff: 27. September 2013
[21] http://www.w3.org/TR/1999/REC-html401-19991224/struct/objects.html#h-13.2, Letzter Zugriff: 27. September 2013

3.3.2 Verlinkung

Sehr wichtig für die Navigation auf *Webseiten* und in bzw. zwischen *Webpräsenzen* ist die Verlinkung. Mit einem Link, `<a>`-Tag, kann man andere *Seiten* aufrufen. Außerdem kann man an bestimmte Bereiche einer *Seite* springen, um schnell zu bestimmten Abschnitten zu gelangen oder Dateien herunterladen. [W3C (1999), 12.2 The `A` element[22]]

Mit dem `href`-Attribut spezifiziert man, mittels validem URI, das Ziel des Links. Das Ziel wird mit der Regel `A-HREF` überprüft. Diese Regel ist auch hauptsächlich dazu da, um den betrachteten Link besser identifizieren zu können.

Damit zum Beispiel ein blinder *Seitenbesucher* der *Seite* das Ziel identifizieren kann, ist eine Beschreibung des Ziels erforderlich. Diese Beschreibung kann dann von einem *Screenreader* anstelle des URIs vorgelesen werden. Das Identifizieren eines Linkziels hat nicht nur Vorteile beim Verwenden von *Screenreadern*, sondern kann auch für andere Zwecke von Vorteil sein. So kann *Opera* alle Links einer Seite anzeigen, damit man schnell zu gewünschten Zielen navigieren kann (siehe Abbildung 9).

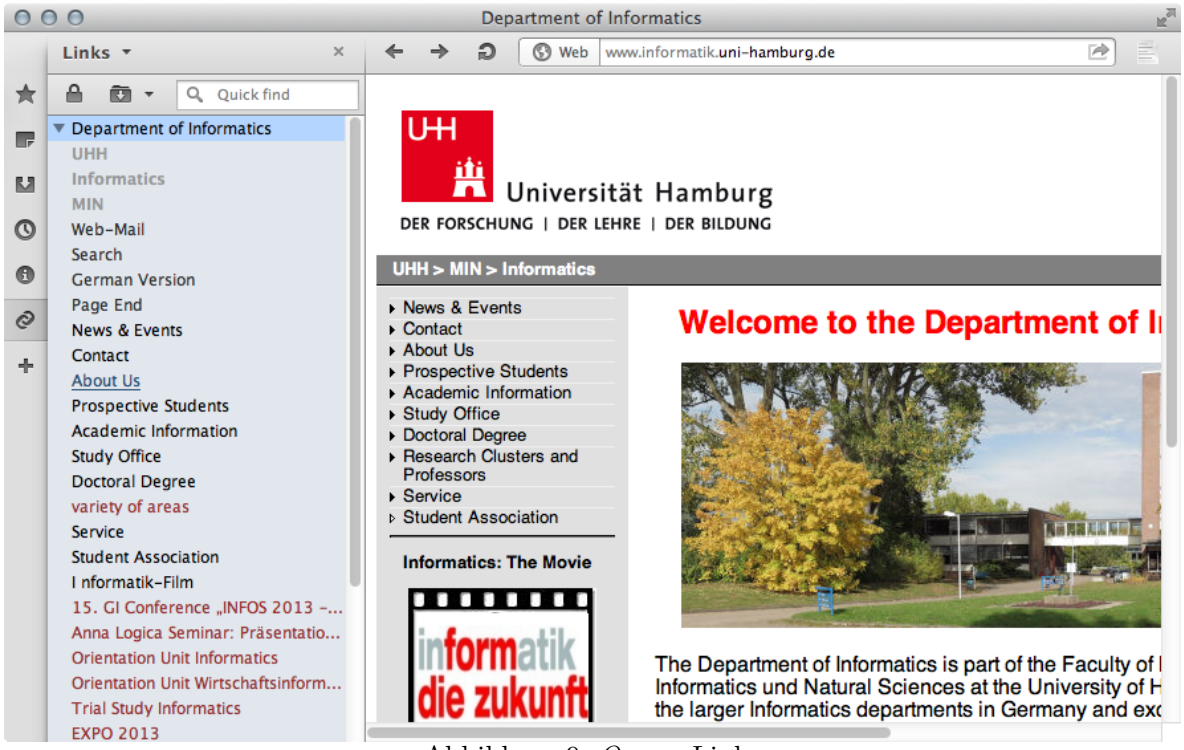

Abbildung 9: *Opera*: Links

Ein weiteres Beispiel ist die *Firefox* Erweiterung *DownThemAll!*[23]. Mit dieser Erweiterung kann man Links, insbesondere herunterladbare Dateien, inspizieren und schnell gewünschte Dateien finden, die man speichern möchte. Wie man in Abbildung 10 erkennen kann, ist ein Link oft nicht aussagekräftig bzw. spiegelt es den Inhalt nicht leicht erkennbar wider. Das

[22] http://www.w3.org/TR/1999/REC-html401-19991224/struct/links.html#h-12.2, Letzter Zugriff: 27. September 2013

[23] https://addons.mozilla.org/de/firefox/addon/downthemall/, Letzter Zugriff: 27. September 2013

oder ein ähnliches Werkzeug kann für Menschen mit Sehbeeinträchtigung oder motorischen Schwierigkeiten hilfreich sein, Dateien auf *Webseiten* zu finden und zu identifizieren.

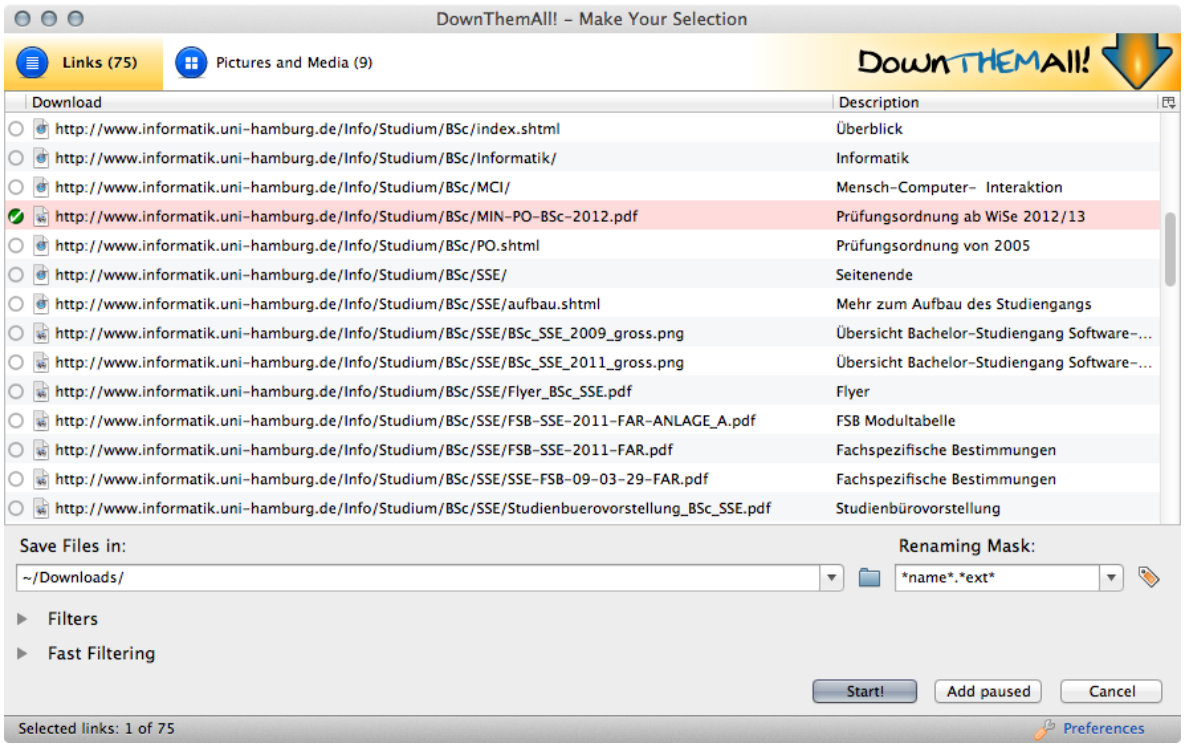

Abbildung 10: *Firefox*: *DownThemAll!*

Ein `<a>`-Tag kann Kindknoten im DOM-Baum erhalten. Dieser Kindknoten ist für die Beschreibung des Links relevant. Ist dieser Kindknoten ein reiner Textknoten, so ist die Beschreibung der Text. Ist der Kindknoten hingegen ein nicht-textuelles Element, so wird die Textalternative als Beschreibung verwendet (siehe Abschnitt 3.3.1, Seite 21). Die Beschreibung eines Links überprüft die Regel `A-TEXT`. Sie resultiert in einer Warnung, wenn keine Beschreibung vorhanden ist.

[W3C (2012), Understanding Success Criterion 2.4.9: Link Purpose[24]]

3.3.3 Seitentitel

Nicht nur Links sind für die Navigation wichtig – entscheidend sind auch Seitentitel. Seitentitel werden mit dem `<title>`-Tag angegeben. Pro Seite muss genau ein Seitentitel angegeben sein. Dieser muss sich im Kopf des HTML-Dokuments befinden (siehe Abbildung 1, Seite 9). Die Regel `TITLE-COUNT` überprüft, dass genau ein `<title>`-Tag vorhanden ist. [W3C (1999), 7.4.2 The `TITLE` element[25]]

[24] http://www.w3.org/TR/2012/NOTE-UNDERSTANDING-WCAG20-20120103/navigation-mechanisms-link.html, Letzter Zugriff: 27. September 2013

[25] http://www.w3.org/TR/1999/REC-html401-19991224/struct/global.html#h-7.4.2, Letzter Zugriff: 27. September 2013

Seitentitel identifizieren das besuchte Dokument. Der Titel wird bei den meisten Browsern in der Titelleiste oder im Tab angezeigt. Sehbeeinträchtigten *Nutzern* wird der Titel vorgelesen. Die Regel `TITLE-TEXT` überprüft, ob Text vorhanden ist.

Um das Dokument ohne Probleme identifizieren zu können, ist es wichtig, dass der Seitentitel Text enthält und über die *Webauftritt* hinweg eindeutig ist. Dies wird mit der Regel `TITLES-UNIQUE` überprüft.

[W3C (2012), Understanding Success Criterion 2.4.2: Page Titled[26]]

Seitentitel sollten möglichst konkret den Inhalt der *Seite* repräsentieren. Die Angemessenheit des Titels lässt sich, wie in Abschnitt 3.3.1 beschrieben, nicht automatisiert ermitteln. Auch hier können Prüfer durch Berichterstellung unterstützt werden, wie es exemplarisch für Alternativtexte gemacht wurde.

[26] http://www.w3.org/TR/2013/NOTE-UNDERSTANDING-WCAG20-20130905/navigation-mechanisms-title.html, Letzter Zugriff: 27. September 2013

4 Das Analyse-Werkzeug

In diesem Abschnitt wird das entwickelte Analyse-Werkzeug vorgestellt. Zunächst wird ein Überblick über die Funktionalität gegeben, danach folgen Details über die konkrete Umsetzung.

4.1 Funktionen

4.1.1 Regeln

Die erkannten *Barrieren*, sowie deren Bedeutung für die *Seitenbesucher* und die potentiellen Probleme sind in Abschnitt 3.3 ab Seite 21 erläutert. Sie werden hier daher nicht erneut aufgeführt.

4.1.2 Ergebnisübersicht

Das Analyse-Werkzeug zeigt die Ergebnisse der angewendeten Regeln an. Diese Ergebnisse kann man sortieren und filtern, sodass man beispielsweise Ergebnisse ausblenden kann, die man nicht betrachten möchte. So kann man sich auf bestimmte *Barrieren* konzentrieren.

4.1.3 Berichterstattung

Da *Barrierefreiheit* nur mithilfe von Automatismen nicht gewährleistet werden kann, kommt man nicht umhin, menschliche Prüfer einzubinden. Das Analyse-Werkzeug versucht Prüfer zu unterstützen. Dafür bietet es eine erweiterbare Berichterstattung an. Stellvertretend wurde ein Typ von Berichten umgesetzt.

Wie beispielsweise in Abschnitt 3.3.1 auf Seite 22 bereits erwähnt, kann nicht ermittelt werden, ob ein Alternativtext nicht-textuellen Inhalt angemessen ersetzt. Diese Aufgabe muss ein menschlicher Prüfer übernehmen. Damit der Prüfer nicht manuell den *Webauftritt* nach Bildern und deren Alternativtexten durchsuchen muss, kann er sich einen *Images Alternatives* Bericht erzeugen lassen. Das manuelle Heraussuchen eines Alternativtextes erfordert in vielen Web-Browsern oft mehrere Schritte. Außerdem werden eventuell Bilder übersehen oder Daten fehlerhaft zusammengetragen.

Der Bericht kann direkt im Programm angesehen werden, wie man in Abbildung 11 sehen kann. Er kann außerdem auch als HTML-Datei gespeichert werden, damit man den Bericht unabhängig vom Analyse-Werkzeug verwenden kann.

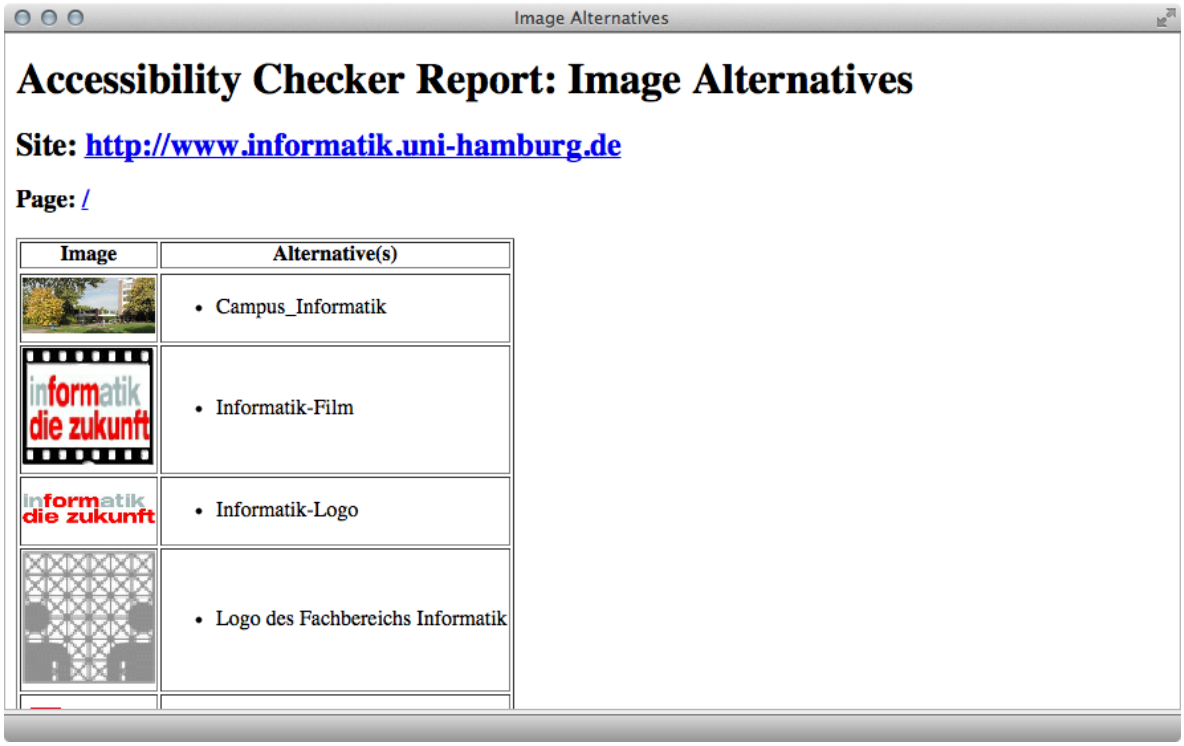

Abbildung 11: Bericht: *Image Alternatives*

4.1.4 Ergebnisvisualisierung

Nachdem eine *Webseite* auf *Barrieren* überprüft wurde, kann man sich die Ergebnisse ansehen. Um *Barrieren* zu beheben, muss man das betroffene Element im Quelltext der *Seite* ändern. Die betroffene Stelle muss zunächst aufgespürt werden.

Die Identifizierung von Elementen über das `id`-Attribut des jeweiligen Elements ist für die Benutzer des Analyse-Werkzeuges nicht geeignet. Nicht jedes Element muss ein `id`-Attribut besitzen. Falls `id`-Attribute vergeben sind, müssen diese eindeutig sein. Dies ist allerdings erst bei validem HTML gewährleistet. [W3C (1999), 7.5.2 Element identifiers: the `id` and `class` attributes[27]]

Intuitiver kann man `<img>`-Tags über das `src`-Attribut und `<a>`-Tags über das `href`-Attribut identifizieren. Selbst bei validem HTML ist keine Eindeutigkeit gewährleistet. Außerdem ist das `src`-Attribut von `<a>`-Tags optional. In der Ergebnisbeschreibung zu den Regeln `IMG-ALT` und `A-TEXT` werden diese Informationen trotzdem für die bessere Identifizierung mitgegeben.

[27] http://www.w3.org/TR/1999/REC-html401-19991224/struct/global.html#h-7.5.2, Letzter Zugriff: 27. September 2013

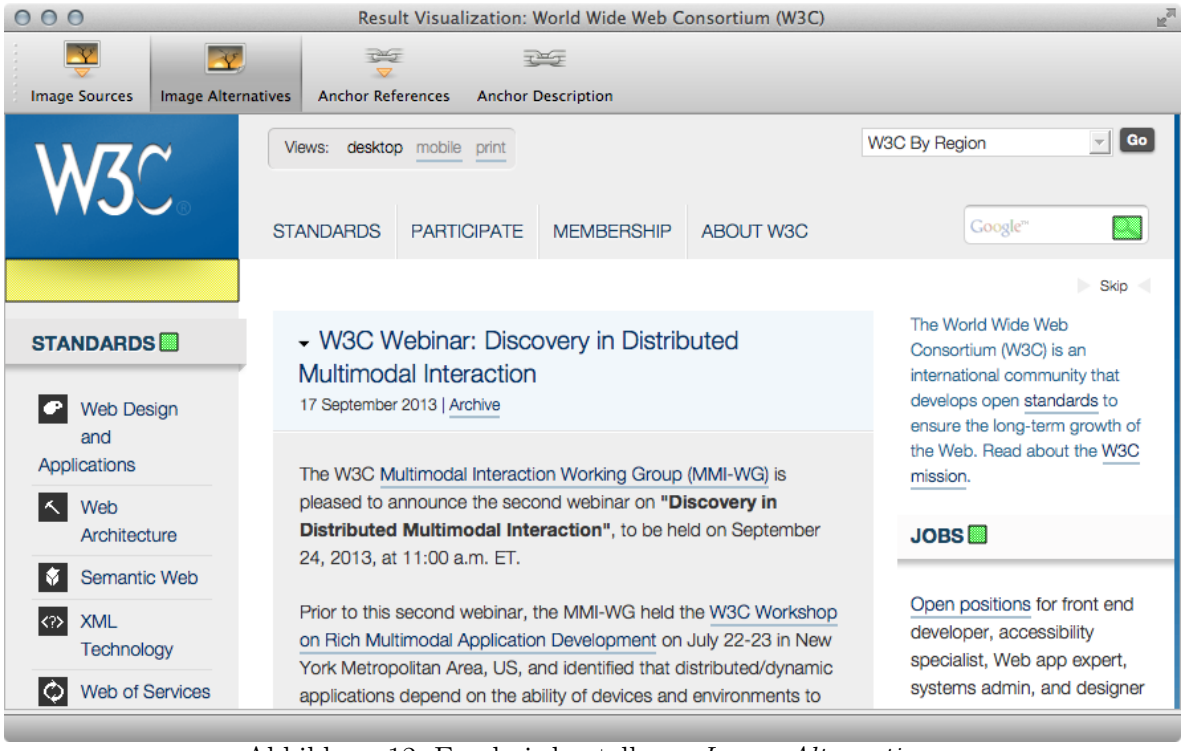

Abbildung 12: Ergebnisdarstellung: *Image Alternatives*

Um die Identifizierung problematischer Elemente zu erleichtern, wird eine Ergebnisvisualisierung angeboten. Zunächst wird die *Webseite* wie in einem Web-Browser dargestellt. Dann kann man die Visualisierung von ausgewählten Regeln aktivieren. Die überprüften Elemente werden dann, je nach Resultat, eingefärbt. Elemente, bei denen keine Probleme entdeckt wurden, werden grün eingefärbt, bei Warnungen, gelb und rot, wenn die Regel fehl schlug. Anker, die keine Referenzen haben, werden blau eingefärbt.

In Abbildung 12 sind die Ergebnisse der `IMG-ALT` Regel hervorgehoben. Unter dem W3C-Logo ist ein Bild gelb eingefärbt. Das heißt, dass zwar ein `alt`-Attribut zu dem `<img>`-Tag gibt, dieser jedoch keinen Wert hat. Hier kann man anhand der Visualisierung schnell erkennen, dass die Warnung an dieser Stelle ignoriert werden kann, da sie unproblematisch ist. Es handelt sich um ein dekoratives Element, welches Schattierung unter dem Logo darstellt.

Die Ergebnisvisualisierung hat zwei Probleme, die in zukünftigen Versionen ausgebessert werden sollten. Zum Einen ist eine einfache Einfärbung der Ergebnisse nicht zugänglich. Farbenblinde Nutzer können eingefärbten Elemente womöglich nicht erkennen oder unterscheiden. Die Darstellung der *Webseite* wird ebenfalls nicht berücksichtigt. Zum Anderen kann es zu Darstellungsproblemen kommen, wenn auf den *Webseiten JavaScript* verwendet wird.

4.2 Implementation

In diesem Abschnitt geht es um die konkrete Realisierung des Analyse-Werkzeuges. Das Analyse-Werkzeug besteht aus mehreren Komponenten. Zuerst werden Komponenten, deren Abhängigkeiten und technische Details beschrieben. Anschließend wird die objektorientierte

Modellierung näher beleuchtet. Dann werden weitere Design-Entscheidungen über den Umgang mit Web-Techniken und die Umsetzung hinsichtlich der Überprüfung auf *Barrierefreiheit* erläutert. Es werden nur die Komponenten beschrieben, die für diese Arbeit relevant sind.

Im Folgenden wird anstelle von Programmbibliotheken nur von Bibliotheken gesprochen.

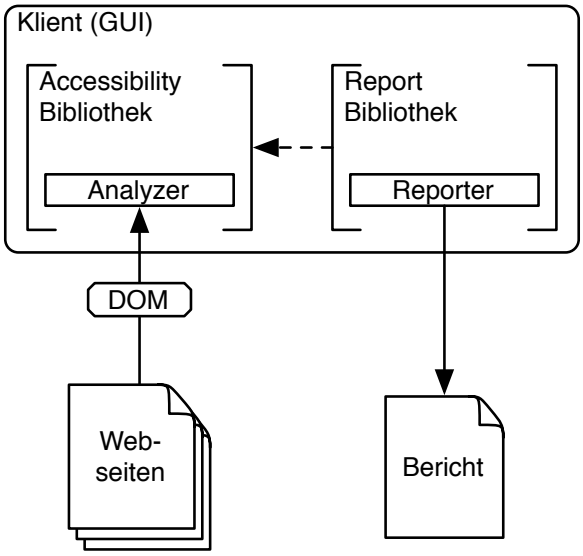

Abbildung 13: Architektur des Analyse-Werkzeuges

4.2.1 Architektur

Die Architektur des Analyse-Werkzeuges wurde so entworfen, dass die Funktionalität in Bibliotheken liegt. Auf diese Weise kann man beliebige Programme erstellen, die mit den Daten arbeiten, welche die Bibliotheken bereitstellen. Für das im Rahmen dieser Arbeit erstellte Analyse-Werkzeug ist es eine grafische Benutzeroberfläche (GUI). Kommandozeilenwerkzeuge (CLIs) oder Web-Interfaces wären ebenso denkbar (siehe Abschnitt 7, Seite 38).

Die Struktur ist in Abbildung 13 dargestellt. Aus einer *Webseite*, welches ein HTML-Dokument sein sollte, wird zunächst das DOM ermittelt. Diese Aufgabe übernimmt der Klient. Der DOM-Baum wird dann mit Regeln überprüft und pro *Webseite* werden die Resultate gespeichert. Mithilfe dieser Ergebnisse kann man dann Berichte erzeugen (siehe Abschnitt 4.1.3, Seite 26) oder den Klient weitere Aufgaben erledigen lassen, wie beispielsweise die, in der GUI vorhandenen, Ergebnisvisualisierung (siehe Abschnitt 4.1.4, Seite 27).

Accessibility-Bibliothek
Die wichtigste Komponente ist die *Accessibility*-Bibliothek. Die Bibliothek analysiert den DOM-Baum eines HTML-Dokumentes mithilfe von Regeln auf mögliche *Barrieren*. Das Dokument kann eine *Webseite* sein.

Report-Bibliothek
Die *Report*-Bibliothek ist für die Berichterstattung zuständig. Berichte lassen sich direkt in

eine Datei oder in einem Buffer speichern. Der Buffer kann dazu verwendet werden, dass man die Berichte, wie es in der GUI-Anwendung umgesetzt wurde, ansehen kann (siehe Abschnitt 4.1.3, Seite 26).

Es gibt momentan nur einen Bericht, der Bilder und deren Alternativtexte gegenüber stellt. Die Bibliothek ist so aufgebaut, dass man weitere Berichte hinzufügen kann.

Die *Report*-Bibliothek verwendet die *Accessibility*-Bibliothek.

GUI-Anwendung
Die GUI-Anwendung stellt die Ergebnisse, textuell oder grafisch, dar, die von der *Accessibility*-Bibliothek bereitgestellt werden. Ferner kann man mit der Anwendung Berichte ansehen oder speichern, die durch die *Report*-Bibliothek generiert werden.

4.2.2 Objektorientierte Modellierung

In diesem Abschnitt wird der objektorientierte Entwurf des im Rahmen der Arbeit erstellten Programms vorgestellt. Dabei werden Entwurfsmetaphern im Sinne des Werkzeug & Material Ansatzes (WAM) verwendet.

Die Hauptklasse der *Accessibility*-Bibliothek ist der `Analyzer`. Der Nutzer des Analyse-Werkzeuges gibt die zu analysierenden *Webseiten* ein und startet dann den Analyse-Vorgang. Zunächst wird vom Klienten das zu analysierende DOM erstellt. Dann wendet der `Analyzer` die implementierten Regeln darauf an, ohne dass weiteres Zutun des Nutzers erforderlich ist. Somit ist der `Analyzer` ein Automat. [Züllighoven (1998), Seite 88 f.]

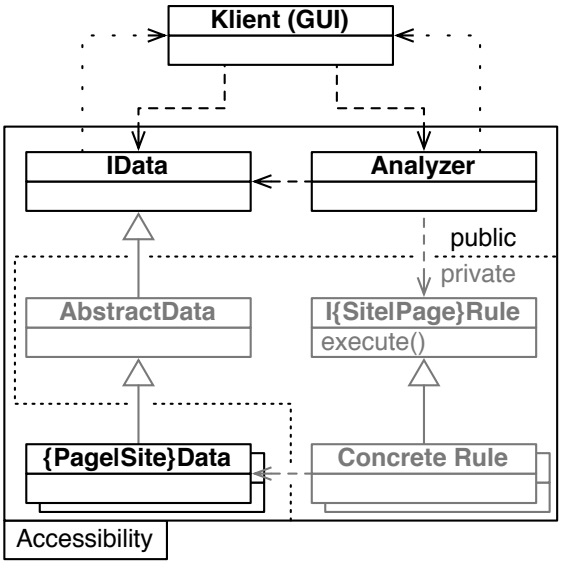

Abbildung 14: Implementierung: Sichtbarkeit & Abhängigkeit der Klassen

Der `Analyzer`-Automat besteht aus Werkzeugen, die nicht Teil der Schnittstelle der *Accessibility*-Bibliothek sind – die Regeln. Mithilfe der Regel-Werkzeuge bearbeitet der Automat ein DOM. Ein DOM ist demzufolge Material, welches die *Webseite* und die Ergebnisse der Analyse dieser repräsentiert. [Züllighoven (1998), Seite 85 f.] Die Knoten dieses DOMs werden mit Informationen angereichert (siehe Abschnitt 4.2.3, Seite 32).

Die Regeln, sowohl für *Seiten* als auch für *Webauftritte*, bearbeiten das DOM als Material. Die Implementation der Regeln orientiert sich an dem Entwurfsmuster Befehl (siehe Abbildung 14). Das erlaubt es, später die Regeln beliebig zu kombinieren und einzelne Regeln zu testen. [Gamma u. a. (2009), Seite 273 ff.]

Die `Rule`-Klassen enthalten teilweise mehrere Regeln. So sind beispielsweise die Regeln `A-TEXT`

und `A-HREF` in der Klasse `PageAnchorRule` implementiert, da eine Beschreibung nur erforderlich ist, falls der `<a>`-Tag ein Ziel definiert hat.

Wie in Abschnitt 4.2.1 beschrieben, bearbeitet der Automat in seinem Arbeitsprozess pro *Webseite* aus dem HTML-Quelltext ein DOM.

Die Berichterstattung sowie die Ergebnisvisualisierung sind Darstellungswerkzeuge für die ermittelten Ergebnisse. Ein Darstellungswerkzeug erlaubt es, das Material, also das DOM oder die Ergebnisse selbst, darzustellen, aber nicht zu bearbeiten. [Züllighoven (1998), Seite 83 ff. & 166]

Ferner wurden, für im Programm verwendete URIs, die Entwurfsmuster Singleton und Fliegengewicht verwendet. [Gamma u. a. (2009), Seite 157 ff. & 223 ff.]

4.2.3 Umsetzung

In diesem Abschnitt werden einige Entscheidungen hinsichtlich der Umsetzung genauer betrachtet.

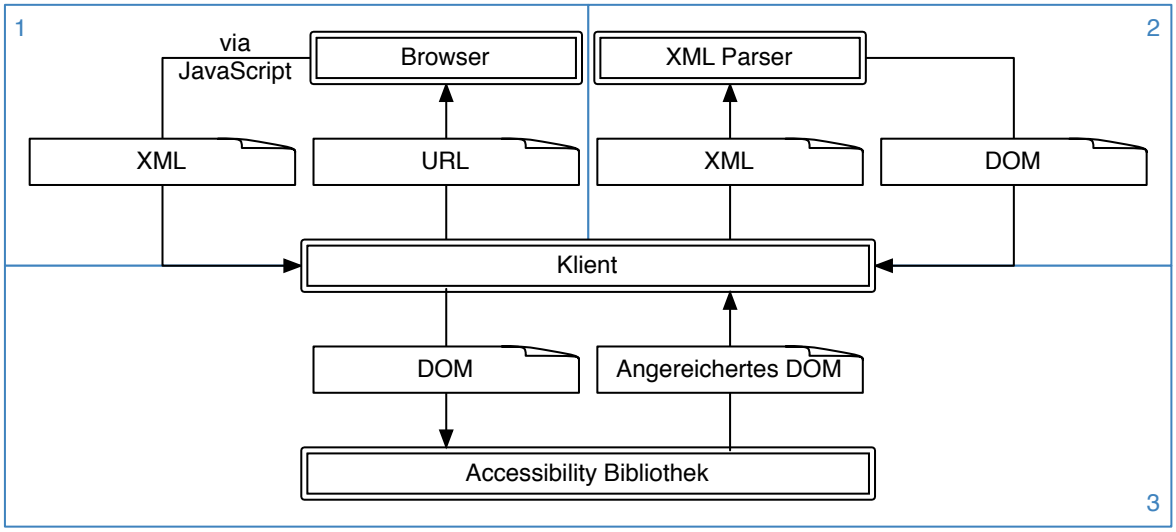

Abbildung 15: Arbeitsablauf: Erzeugung & Analyse des DOM

Abbildung 15 stellt den grundsätzlichen Arbeitsablauf des Analyse-Werkzeuges dar:

1. Zunächst wird von dem Klienten ein Uniform Resource Locator (URL) zu einer *Webseite* an einen Browser übergeben. Dieser ruft dann die entsprechende *Seite* auf und erstellt XML-Quelltext, welcher das DOM repräsentiert und gibt diesen XML-Quelltext zurück an den Klienten.
2. Der Klient übergibt dann diesen XML-Quelltext an den XML-Parser, der daraus ein DOM erstellt.
3. Wenn das DOM erfolgreich erstellt werden konnte, wird dieser an die `Accessibility`-Bibliothek übergeben. Dort wird das DOM mit der Anwendung der Regeln angereichert.

Strikter Parser
Zunächst muss das DOM erstellt werden. Dies wird mithilfe eines XML-Parsers gemacht. Da valider HTML-Quellcode allerdings nicht valides XML sein muss, ist ein reiner XML-Parser zu restriktiv (siehe Abschnitt 3.2.1, Seite 18). Diese Restriktion ist jedoch erwünscht.

Hilfsmittel, wie zum Beispiel *Screenreader* oder ein Werkzeug zur Linearisierung von *Webseiten*, sind Erweiterungen der Funktionalität für Web-Browser. Der Browser führt unter Anderem *JavaScript* aus und somit kann das DOM verändert werden, wobei hingegen der HTML-Quelltext statisch ist. Es ist daher wichtig, dass diese Hilfsmittel den DOM des Browsers verwenden. Sie können über, vom Browser angebotene Schnittstellen, dann mit dem DOM interagieren. Falls sich aufgrund der Verwendung von beispielsweise *JavaScript* der Inhalt der *Webseite* ändert, müssen die Hilfsmittel in der Lage sein, darauf zu reagieren. So informieren zum Beispiel *Screenreader* einen *Seitenbesucher* darüber, wenn sich der Inhalt oder Aufbau der *Seite* verändert hat.

Da aus einem DOM wieder XML-Quellcode erzeugt werden kann und das statische HTML einer *Webseite* für die Analyse keine große Rolle spielt, wurde auf einen HTML-Parser verzichtet. Zusätzlich müssen HTML-Parser mehr beachten und stellen daher eine potentielle Fehlerquelle dar. Im Analyse-Werkzeug wird ein sogenannter "*headless browser*", also ein Browser, der im Hintergrund als Teil des Programms läuft, verwendet, um den XML-Quelltext zu erzeugen. Dies geschieht mittels *JavaScript*. Dieses Vorgehen kann später für andere Browser übernommen werden (siehe Abschnitt 7, Seite 38).

Anreicherung des DOM
Damit man die Ergebnisse bestimmter Elemente der *Webseite* zuordnen kann, wird diese Information im Knoten im DOM-Baum festgehalten. Momentan wird dem Element mithilfe des `class`-Attributs nur die Information über den Ausgang der angewendeten Regeln festgehalten. Bestehende Informationen gehen dabei nicht verloren. Dies kann dann vom Klienten genutzt werden. Das Analyse-Werkzeug kann mit dieser Technik die Ergebnisse visuell darstellen (siehe Abschnitt 4.1.4, Seite 27). Eine Darstellung mittels CSS oder eine automatische Reparatur mit *JavaScript*, sofern möglich, wären ebenfalls denkbare Szenarien.

4.3 Qualitätssicherung

Es wurde eine Vielzahl von *Webseiten* und *Webpräsenzen* für manuelle Tests verwendet. Besonders hilfreich war der *Mother Effing Tool Confuser*[28]

Um die Qualität des Analyse-Werkzeuges zu verbessern, wurden Unit-Tests erstellt. Die meisten dieser Unit-Tests sind datengesteuert (eng. *data-driven*). Die Tests wurden mit dem *Qt-Test*-Framework erstellt.

4.3.1 Testabdeckung

Mithilfe von Testabdeckung des Quelltextes (eng. *code coverage*) kann man ermitteln, welche Teile des Quelltextes durch Tests ausgeführt wurden.

[28] *Mother Effing Tool Confuser*: http://mothereffingtoolconfuser.com, Letzter Zugriff: 6. Oktober 2013

Zunächst wird das Analyse-Werkzeug in einer speziellen Konfiguration übersetzt, die in der Dokumentation des Programms beschrieben ist. Danach werden alle Unit-Tests ausgeführt und der Quelltext transparent instrumentiert. Diese Instrumentationen enthalten Informationen darüber, welche Teile des Quelltextes ausgeführt wurden.

Für die Ermittlung der Testabdeckung wurde *Squish Coco* verwendet. Andere Werkzeuge können sich sowohl im Vorgang der Ermittlung, als auch in den Resultaten unterscheiden[29].

In den Anlagen dieser Arbeit befindet sich ein Bericht über die Testabdeckung des Analyse-Werkzeuges im HTML-Format, damit man sich diesen auch ohne *Squish Coco* ansehen kann.

[29] *Squish Coco*: Code Coverage Overview: http://doc.froglogic.com/squish-coco/2.1/codecoverage.html, Letzter Zugriff: 6. Oktober 2013

5 Evaluation

In diesem Abschnitt werden *Webseiten* und *Webpräsenzen* mithilfe des Analyse-Werkzeuges untersucht.

Da das Analyse-Werkzeug nur wenige Regeln besitzt, sind diese Untersuchungen sehr oberflächlich. Es werden daher lediglich erkannte Probleme hervorgehoben und keine Bewertung über die generelle *Barrierefreiheit* der *Auftritte* abgegeben.

5.1 Ausgewählte Seiten

5.1.1 Bundesministerium für Arbeit und Soziales

Das Bundesministerium für Arbeit und Soziales ist als Behörde gesetzlich dazu verpflichtet, auf *Barrierefreiheit* des offiziellen *Webauftritts* zu achten. Es wurden zufällig 5 *Seiten* ausgewählt[30], die möglicherweise für bestimmte *Besucher* besonders interessant sein könnten.

- http://www.bmas.de/DE/Startseite/start.html
- http://www.bmas.de/DE/Leichte-Sprache/Leichte-Sprache/inhalt.html
- http://www.bmas.de/DE/Themen/Aus-und-Weiterbildung/inhalt.html
- http://www.bmas.de/DE/Service/Gesetze/beschv.html
- http://www.bmas.de/DE/Service/Benutzerhinweise/inhalt.html

Es gibt einige Bilder ohne Alternativtexte aus dem Menü, welche dekorativen Charakter haben. Ein Bild auf der Startseite besitzt keinen Alternativtext, löst allerdings durch Anklicken eine Funktion aus: Pausieren der Animation. Ob es sich hierbei um eine *Barriere* handelt, ist aber nicht so einfach zu sagen. Einerseits wird blinden *Besuchern* die Information über die Funktionalität vorenthalten – andererseits werden sie diese Funktionalität nicht benötigen. Da es sehr viele verschiedene Arten von kognitiven oder visuellen Beeinträchtigungen gibt, sollte man auch hier einen Alternativtext zur Verfügung stellen. Blinde *Besucher* werden diese Funktionalität vielleicht nicht benötigen, aber *Seitenbesucher* mit stark eingeschränktem Sehvermögen, die eventuell durch die Animation zu stark abgelenkt werden und größtenteils mit der Unterstützung eines *Screenreaders* arbeiten, könnten an dieser Stelle ein Problem haben, die Animation zu deaktivieren. Symbole, wie das Pause-Symbol auf dem Bild, nicht zu erkennen, wäre eine denkbare Problematik.

Einige Links besitzen keine Beschreibung. Sie befinden sich außerhalb des sichtbaren Bereiches der *Seite* und werden in der Regel von Hilfsmitteln speziell behandelt. [Cunningham (2012)] Diese Links sollten von dem Analyse-Werkzeug in zukünftigen Versionen ausgeschlossen oder anders behandelt werden.

[30] Letzter Zugriff der *Seiten* des Bundesministeriums für Arbeit uns Soziales: 6. Oktober 2013

5.1.2 Google

Als Suchmaschine wird *Google* häufig zum Auffinden von *Webseiten* verwendet. Es wurde zum Einen die Startseite und eine *Seite* mit angezeigten Ergebnissen gewählt[31]:

- https://www.google.de/
- https://www.google.de/search?q=Barrierefreiheit

Auf der Startseite hat das Logo von *Google Chrome* keinen Alternativtext. An dieser Stelle kann man sich streiten, ob es sich hierbei um dekorativen Charakter handelt oder nicht. Die Empfehlung *Google Chrome* zu verwenden ist allerdings auch ohne das Logo erkennbar.

Auf der *Seite* mit den Ergebnissen zu dem exemplarisch gewähltem Suchbegriff "*Barrierefreiheit*" wird für den Link, der zurück auf die Startseite führt, keine Beschreibung erkannt. Diese Beschreibung befindet sich jedoch im `title`-Attribut des Links (siehe Listing 9) und sollte von dem Analyse-Werkzeug in zukünftigen Versionen erkannt werden.

```
<a class="gb_bb␣gb_ba" href="/webhp?hl=de&[...]" title="Weiter␣zur␣Google-Star
tseite">
  <span class="gb_c"></span>
</a>
```

Listing 9: HTML: *Google*: Zurück zur Startseite[32]

5.1.3 Universität Hamburg

Der offizielle *Webauftritt* der Universität Hamburg bietet wichtige Informationen rund um das Studium an der Universität Hamburg an. Es wurden 5 *Seiten* ausgewählt[33], die für einige *Besucher* von besonderem Interesse sein könnten:

- http://www.uni-hamburg.de
- http://www.uni-hamburg.de/campuscenter/studienangebot.html
- http://www.uni-hamburg.de/campuscenter/bewerbung/bachelor-staatsexamen/fristen-termine.html
- http://www.uni-hamburg.de/campuscenter/bewerbung.html
- http://www.uni-hamburg.de/campuscenter/beratung/kontakt.html

Auf der *Seite*, die das Studienangebot präsentiert, sind ein paar Download-Links zu PDF-Dateien ohne Beschreibung (siehe Listing 10). Sie sind im Browser ebenfalls nicht sichtbar, werden jedoch in Link-Übersichten oder Download-Werkzeugen angezeigt. Es gibt jeweils weitere Download-Links, die, aufgrund des gleichen Dateinamens die gleichen Dateien sein könnten. Zwei ausgewählte Dateien mit gleichem Dateinamen haben dementgegen unterschiedliche Hash-Werte und sind somit nicht identisch. Ein Dokument bleibt bestimmten Nutzern somit verborgen.

[31] Letzter Zugriff auf *Google*: 6. Oktober 2013
[32] Das `href`-Attribut in Listing 9 wurde verkürzt.
[33] Letzter Zugriff die ausgewählten *Seiten* der Universität Hamburg: 6. Oktober 2013

```
<li>
  <a target="_blank" href="/campuscenter/download/lehramt-gymnasium.pdf"></a>
  <a target="_blank" href="/campuscenter/studienorganisation/formulare-informa
tionsmerkblaetter/lehramt-gymnasium.pdf">
    Lehramt am Gymnasium (PDF)
  </a>
  <br>
</li>
```

Listing 10: HTML: Universität Hamburg: Dateidownload Studienangebot

5.1.4 Wikipedia

Wikipedia ist eine Online-Enzyklopädie, bei der *Seitenbesucher* Informationen beisteuern können. Dieser *Webauftritt* wurde ausgewählt, da er eine weit verbreitete und sehr häufig verwendete Quelle für Informationen ist. Es wurden 5 *Seiten* durch die auf *Wikipedia* vorhandene "*Zufallsartikel anzeigen*"-Funktion[34] ausgewählt:

- http://de.wikipedia.org/wiki/American_Blimp_Corporation
- http://de.wikipedia.org/wiki/Biegenbrück
- http://de.wikipedia.org/wiki/Bistum_Perpignan-Elne
- http://de.wikipedia.org/wiki/Jüdischer_Friedhof_(Boizenburg/Elbe)
- http://de.wikipedia.org/wiki/Stephen_Russell_Mallory_senior

Es gibt auf jeder *Seite* Bilder ohne Alternativtexte, die nicht nur dekorativen Charakter haben. Zwei der *Seiten* enthalten Informationen über Orte. Dort wird jeweils angeboten, sich die Position des Ortes auf einer Karte anzeigen zu lassen. Diese Funktion ist durch Klicken auf einen Link aktivierbar. Der Link enthält ein Bild, welches keinen Alternativtext enthält (siehe Listing 11). Dies ist nach aktuellem HTML-Standard nicht gestattet. Direkte Probleme mit der *Barrierefreiheit* sollte es an dieser Stelle allerdings nicht geben, da die Funktion durch das `title`-Attribut hervorgeht. Es wäre ratsam, eine Kopie des Wertes aus dem `title`-Attribut des Links als Alternativtext für das Bild zu verwenden. Alternativ kann man dem Bild auch ein leeres `alt`-Attribut zuweisen, damit der dekorative Charakter des Bildes hervorgehoben wird – da der Link an dieser Stelle das informationstragende Element ist und dieser durch das Bild nur dargestellt wird.

```
<a href="#" title="Zeige␣Koordinaten␣auf␣einer␣Karte␣von␣OpenStreetMap" class=
"noprint␣osm-icon-coordinates">
  <img src="//upload.wikimedia.org/wikipedia/commons/thumb/c/c9/OpenStreetMapL
ogo.png/17px-OpenStreetMapLogo.png" width="17px" height="17px"/>
</a>
```

Listing 11: HTML: Karte anzeigen auf *Wikipedia*

Zu einigen Links konnte keine Beschreibung gefunden werden, weil die darin eingebetteten Bilder keinen oder einen leeren Alternativtext haben.

[34] *Wikipedia*: Zufälliger Artikel: http://de.wikipedia.org/wiki/Spezial:Zufällige_Seite, Letzter Zugriff: 6. Oktober 2013

6 Zusammenfassung

In dieser Arbeit wurde gezeigt, dass es eine sehr große Schnittmenge zwischen *Barrierefreiheit* und *Benutzerfreundlichkeit* gibt. Zu den betrachteten *Barrieren* wurden sowohl Konsequenzen für Menschen mit Beeinträchtigungen als auch der Mehrwert für die *Benutzerfreundlichkeit* aufgezeigt. Der Mehrwert an *Benutzerfreundlichkeit* wurde aufgezeigt, da sie die Umsetzung von *Barrierefreiheit* attraktiver macht.

Das erstellte Analyse-Werkzeug wurde vorgestellt. Dabei wurden besonders die Funktionalität hinsichtlich der Analyse auf *Barrierefreiheit* und Unterstützung der Prüfer, sowie die softwaretechnischen Aspekte in den Vordergrund gestellt.

Ferner wurde auch aufgezeigt, dass es nicht möglich ist ein Programm zu erstellen, welches entscheiden kann, ob eine *Webseite barrierefrei* ist. Man wird nicht alle *Barrieren* finden oder beheben können. Wichtig für die *Barrierefreiheit* ist, dass man Werkzeuge einsetzt, welche potentielle *Barrieren* automatisiert ermitteln. Diese lassen sich meist schnell beheben und die Werkzeuge können jedes Mal verwendet werden, wenn sich der Inhalt der *Webseite* verändert. Allerdings ist es ebenso wichtig, dass man menschliche Prüfer und Testpersonen mit Beeinträchtigungen am Testprozess beteiligt. Sofern Werkzeuge, wie das im Rahmen dieser Arbeit entwickelte, diese Funktionalität bieten, können sowohl Prüfer, als auch Testperson dabei unterstützt werden.

Der Vollständigkeit halber sei hier noch erwähnt, dass das Thema *Barrierefreiheit* hier nur abgesteckt und keinesfalls vollständig oder ausführlich betrachtet wurde.

"*Barrierefreiheit ist ein Ziel und kein Zustand.*" [Hellbusch u. Probiesch (2011), Seite 11]

7 Ausblick

Barrierefreiheit und *Benutzerfreundlichkeit* werden immer eine Rolle spielen. Der Fortschritt der Technik wird sowohl positiven als auch negativen Einfluss auf diese Gebiete haben. Sowie immer neue technische Möglichkeiten entstehen, *Barrierefreiheit* zu fördern, so werden auch immer neue mobile internetfähige Geräte entwickelt, welche neue *Barrieren* mit sich bringen.

Das Analyse-Werkzeug erkennt momentan nur exemplarisch einen Bruchteil an *Barrieren*, die sich automatisiert erkennen lassen. Um die Implementierung weiterer Regeln führt für die Verwendbarkeit des Analyse-Werkzeuges kein Weg herum. Das Erstellen weiterer Berichte, Speichern, Importieren und Exportieren von Resultaten sollte umgesetzt werden. Außerdem funktioniert die Ergebnisvisualisierung nicht für alle *Seiten* und sollte ausgebessert werden.

Im Folgenden werden Ideen aufgeführt, welche Funktionalität das Analyse-Werkzeug zukünftig noch unterstützen sollte. [35]

Barrierefreiheit des Analyse-Werkzeuges
Damit das Analyse-Werkzeug Prüfer und Testpersonen mit Beeinträchtigungen sinnvoll unterstützen kann, muss es natürlich selbst möglichst zugänglich gestaltet sein.

DOM vom Browser
Momentan verwendet das Analyse-Werkzeug einen "*headless Browser*", um sich das DOM zu erzeugen (siehe Abschnitt 4.2.3, Seite 32). Da die meisten Hilfsmittel Erweiterung der Funktionalität von Browsern sind und teilweise Inhalte auf bestimmte Browser angepasst werden, sollte man auch das DOM eines Browsers zum Testen verwenden können. Damit kann man auch Probleme feststellen, die vielleicht in einem anderen Browser oder einer anderen Version gar nicht vorhanden sind.

Dynamische Regeln
Es wäre gut, wenn Nutzer des Analyse-Werkzeuges in der Lage wären, Regeln spezifisch zu gestalten. Dafür könnte man eine Skript-Sprache oder eine Domain-Specific Language (DSL) verwenden. Dies erlaubt auch das dynamische Einbinden, Verbessern und Erweitern von Regeln.

Test-Integration
Es sollte möglich sein, das Analyse-Werkzeug in Verwaltungssysteme zu integrieren, sodass, wenn eine Änderung stattfindet, Tests durchgeführt werden können. Sollten *Barrieren* entdeckt werden, sollte das Werkzeug automatisch ein Fehlerbericht eintragen können.

[35] Vgl. http://www.karlgroves.com/2013/06/28/choosing-an-automated-accessibility-testing-tool-13-questions-you-should-ask/, Letzter Zugriff: 29. September 2013

Anlagen

Folgende Anlagen sind Bestandteil dieser Arbeit:

- PDF-Version der Arbeit
- LaTeX-Quelltext der Arbeit
- Übersetzte *Windows* & *OS X* Release Version des *AccessibilityCheckers*
- Quelltext des *AccessibilityCheckers*
- HTML-Dokumenation des Quelltextes
- Ein HTML-Bericht über die Testabdeckung des Quelltextes

Danksagung

Ich bedanke mich bei Charline Bernatek und Alin Golbs für hilfreiche Kritik und Anregungen.

Ich bedanke mich hiermit außerdem bei der *froglogic GmbH* für Anregungen und dafür, dass mir *Squish Coco* zur Verfügung gestellt wurde.

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die Bachelorarbeit im Studiengang Software-System-Entwicklung selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten (Internet-)Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht. [36]

Hiermit bestätige ich außerdem, dass ich damit einverstanden bin, dass diese Arbeit in den Bibliotheksbestand der Informatik-Bibliothek der Universität Hamburg aufgenommen werden darf.

Ort, Datum

Unterschrift

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[Brewer u. a. 2006] Brewer, Judy ; Henry, Shawn L. ; Education & Outreach Working Group (EOWG): *Policies Relating to Web Accessibility*. http://www.w3.org/WAI/Policy/. Version: 25. August 2006. – Letzter Zugriff: 27. September 2013

[Cunningham 2012] Cunningham, Katie: *Accessibility Handbook*. O'Reilly Media, Inc., 2012. – ISBN 978–1–4493–2285–4

[Duden 2013] *"barrierefrei"*. http://www.duden.de/node/666578/revisions/1158377/view. Version: 15. Januar 2013. – Letzter Zugriff: 27. September 2013

[Gamma u. a. 2009] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Entwurfsmuster*. Addison-Wesley, 2009. – ISBN 987–3–8273–2824–3

[Behindertengleichstellungsgesetz 2002] *Gesetz zur Gleichstellung behinderter Menschen (Behindertengleichstellungsgesetz – BGG)*. BGBl. I S. 1467, 1468; zuletzt geändert durch Artikel 12 G. v. 19. Dezember 2007 BGBl. I S. 3024. 27. April 2002

[Barrierefreie-Informationstechnik-Verordnung 2012] *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung – BITV 2.0)*. BGBl. I S. 1843. 12. September 2012

[Grundgesetz 1949] *Grundgesetz für die Bundesrepublik Deutschland (GG)*. BGBl. I S. 1; zuletzt geändert durch Artikel 1 G. v. 11. Juli 2012 BGBl. S. 1478. 23. Mai 1949

[Harper u. Yesilada 2008] Harper, Simon ; Yesilada, Yeliz: *Web Accessibility*. Springer London, 2008 (Human-Computer Interaction Series). – ISBN 978–1–84800–049–0

[Hellbusch u. Probiesch 2011] Hellbusch, Jan E. ; Probiesch, Kerstin: *Barrierefreiheit verstehen und umsetzen*. 1. Auflage. dpunkt.verlag GmbH, 2011. – ISBN 978–3–89864–520–1

[Kent u. Bäckmann 2006] Kent, Peter ; Bäckmann, Marcus: *Suchmaschinenoptimierung für Dummies*. 1. Auflage. Wiley, 2006. – ISBN 978–3–527–70317–3

[Langridge 2005] Langridge, Stuart: *DHTML Utopia - Modern Web Design: Using JavaScript and DOM*. SitePoint Pty, Limited, 2005 (Sitepoint Series). – ISBN 978–0–9579218–9–4

[Qt Project 2013] *Qt 5.1 Documentation*. http://qt-project.org/doc/qt-5.1/. Version: 2013. – Letzter Zugriff: 22. September 2013

[Universität Hamburg, Studienbüro Informatik 2013] *Merkblatt zur Anfertigung von Bachelorarbeiten im Studiengang B.Sc Software-System-Entwicklung - Wissenswertes rund um die Bachelorarbeit*. http://www.informatik.uni-hamburg.de/StB/Formulare/BSc_SSE_Anmeldung-BA.pdf. Version: 22. April 2013. – Letzter Zugriff: 27. September 2013

[W3C 1999] W3C: *HTML 4.01 Specification*. http://www.w3.org/TR/1999/REC-html401-19991224/. Version: 24. Dezember 1999. – Letzter Zugriff: 27. September 2013

[W3C 2008] W3C: *Web Content Accessibility Guidelines (WCAG) 2.0*. http://www.w3.org/TR/2008/REC-WCAG20-20081211/. Version: 11. Dezember 2008. – Letzter Zugriff: 27. September 2013

[W3C 2012] W3C: *Understanding WCAG 2.0*. http://www.w3.org/TR/2012/NOTE-UNDERSTANDING-WCAG20-20120103/. Version: 3. Januar 2012. – Letzter Zugriff: 27. September 2013

[Züllighoven 1998] Züllighoven, Heinz: *Das objektorientierte Konstruktionshandbuch*. 1. Auflage. dpunkt.verlag GmbH, 1998. – ISBN 978–3–932588–05–1

Glossar

Barriere

Etwas, was einem an der Erreichung eines Ziels hindert. 5–7, 10, 14–16, 18, 20, 21, 26, 27, 29, 34, 37, 38

Barrierefreiheit (eng. *accessibility*)
Die Möglichkeit sein Ziel selbstständig zu erreichen. 5–7, 10, 11, 14, 15, 17, 18, 20, 26, 29, 34, 36–38

Benutzerfreundlichkeit (eng. *usability*)
Das Empfinden des Nutzers, ob sein Ziel "angemessen" erreicht werden kann. 5, 7, 15, 37, 38

Heuristik
Eine Methode mit Annahmen, beziehungsweise Mutmaßungen, Lösungen zu erschließen. 7, 20

JavaScript
Eine Skriptsprache, die besonders im Web-Kontext prominent ist. 10, 11, 28, 32

Screenreader
Ein Programm, welches den Inhalt einer Webseite vorliest. 8, 14, 18–20, 22, 23, 32, 34

Seitenbesucher *Besucher*, *Nutzer*
Der Besucher einer *Webseite*. 6, 7, 10, 11, 15, 16, 19, 21, 23, 25, 26, 32, 34–36

Webauftritt *Webpräsenz* (eng. *website*)
Eine Sammlung von *Webseiten*, die hinter einer Adresse stehen. 5, 6, 10, 14–17, 23, 25, 26, 30, 32, 34–36

Webseite *Seite* (eng. *webpage*)
Ein Dokument, welches Teil eines *Webauftritts* ist. 5–7, 10, 11, 14–20, 22–25, 27–32, 34–38

Abkürzungsverzeichnis

ARIA Accessible Rich Internet Applications 10

ASCII American Standard Code for Information Interchange 21

BITV 2.0 Barrierefreie-Informationstechnik-Verordnung 6

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart 20, 21

CLI Command Line Interface 29

CSS Cascading Style Sheets 9–11, 14, 20, 22, 32

DOM Document Object Model 9, 10, 18, 19, 24, 29–32, 38

DSL Domain-Specific Language 38

GUI Graphical User Interface 29, 30

HTML Hypertext Markup Language 8–11, 14, 16, 18–20, 22, 24, 26, 27, 29, 31–33, 35, 36, 39

OWL Web Ontology Language 8

PDF Portable Document Format 6, 35, 39

SEO Search Engine Optimization 16

URI Uniform Resource Identifier 8, 22, 23, 31

URL Uniform Resource Locator 31

W3C World Wide Web Consortium 14, 15, 28

WAI Web Accessibility Initiative 15

WAM Werkzeug & Material-Ansatz 30

WCAG Web Content Accessibility Guidelines 6, 7, 14

XHTML Extensible Hyptertext Markup Language 18

XML Extensible Markup Language 7–9, 18, 19, 31, 32

Regelverzeichnis

A-HREF *Anchor Reference*
Wenn zu einem `<a>`-Tag ein `href`-Attribut existiert, muss der Wert ein valider URI sein. 23, 31

A-TEXT *Anchor Description*
Zu jedem `<a>`-Tag sollte eine Beschreibung des Zieles existieren. 24, 27, 30

IMG-ALT *Image Alternative*
Zu jedem `<img>`-Tag muss ein `alt`-Attribut existieren. 22, 26–28

IMG-SRC *Image Source*
Zu jedem `<img>`-Tag muss ein `src`-Attribut mit validem URI als Wert existieren. 22

TITLE-COUNT *Title Count*
Jede *Webseite* muss genau einen Titel haben. 24

TITLE-TEXT *Title Text*
Titel müssen Text enthalten. 25

TITLES-UNIQUE *Unique Titles*

Jeder Titel einer *Webseite* muss einzigartig in einer *Webauftritt* sein. 25