FAKULTÄT FÜR !NFORMATIK

Auswirkungen von modernen Softwareentwicklungstechniken auf die Barrierefreiheit von Web-Anwendungen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Roman Mauerhofer
Matrikelnummer 0325481

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuerin: o. Univ.-Prof. Dipl.-Ing. Mag. Dr. Gerti Kappel
Mitwirkung: Univ.-Ass. Mag. Dr. Manuel Wimmer

Wien, 01.09.2009

(Unterschrift Verfasser) (Unterschrift Betreuerin)

Erklärung zur Verfassung der Arbeit

Roman Mauerhofer
Wurmsergasse 44/16
1150 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit einschließlich Tabellen, Karten und Abbildungen, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, am 01.09.2009

_____

Roman Mauerhofer

Danksagung

Ich möchte mich bei Univ.-Ass. Mag. Dr. Manuel Wimmer und o. Univ.-Prof. Dipl.-Ing. Mag. Dr. Gerti Kappel für die Vergabe des Themas und die sehr gute Betreuung bedanken. Neben den vielen neuen Erkenntnissen ermöglichte mir dieses Thema ein Hinterfragen von modernen Technologien.

Besonders möchte ich meinen Eltern Herta und Helmut Mauerhofer für die große Unterstützung danken, ohne die mein Studium nicht so reibungslos verlaufen wäre. Meinem Bruder Mag. Andreas Mauerhofer danke ich für das Lektorat und die moralische Unterstützung.

Ich danke allen Freunden und Studienkollegen für die schöne Zeit in Wien, ich werde mich gerne daran erinnern.

Kurzfassung

Die Modellgetriebene Softwareentwicklung ist ein aktueller Trend in der Softwareentwicklung. Dabei ist die Grundidee, mit Hilfe von Code-Generatoren aus abstrakten Softwaremodellen ausführbaren Code automatisch zu generieren. Aktuelle Ansätze legen dabei hauptsächlich Wert auf funktionale Aspekte des zu generierenden Codes, z.B. wie aus UML-Diagrammen Java Code generiert werden kann. Jedoch werden Qualitätsaspekte, die nicht aus den technischen Anforderungen hervorgehen, nur selten berücksichtigt. Ein wichtiger Qualitätsaspekt stellt u.a. die Barrierefreiheit von Webseiten dar. Der Gesetzgeber in Österreich verlangt, dass der barrierefreie Zugang zu behördlichen Internetauftritten für Menschen mit besonderen Bedürfnissen bis 1.1.2008 umgesetzt hätte werden sollen. Um die Barrierefreiheit von Web-Anwendungen festzustellen, werden so genannte Web Accessibilty Guidelines (WAG) vorgegeben. Zum Beispiel gibt es von der W3C die WCAG (Web Content Accessibility Guidelines), welche sich bereits als ein Web Accessibility de facto Standard durchgesetzt haben. Um die Konformität mit den oben geforderten rechtlichen Bestimmungen zu erfüllen, müssen öffentliche Internetauftritte den jeweilig geforderten Richtlinien entsprechen. In dieser Diplomarbeit wird nun eruiert, in welcher Weise moderne Komponenten-Frameworks und Code-Generatoren die sich aus den Richtlinien ergebenden Anforderungen erfüllen. Dazu werden WAG (Web Accessibility Guidelines) konforme statische HTML-Seiten als Ausgangspunkt verwendet, welche dann mittels Komponenten-Frameworks und Code-Generatoren dynamisch weiter entwickelt und als Web-Anwendungen abgebildet werden. Das Ergebnis dieses Generierungsprozesses wird schließlich in Form von HTML-Seiten im Browser gerendert. Diese gerenderten Seiten werden mittels WAG-Validatoren (Prüfprogramme für die WAG-Konformität) hinsichtlich zuvor gewählter Richtlinien geprüft. Aus dem Ergebnis dieser Konformitätsprüfung ist ersichtlich, welche Verletzungen der gewählten WAG auftreten und wie gut die jeweiligen Entwicklungstechnologien für die Erstellung von WAG-konformen Web-Anwendungen geeignet sind. Es hat sich gezeigt, dass der generierte Code der evaluierten HTML-Seiten noch mit vielen Barrieren behaftet ist und noch großes Verbesserungspotential bei den eingesetzten Entwicklungswerkzeugen gegeben ist.

Abstract

Model engineering is a current trend in software engineering. Its aim is the automatic generation of executable code from abstract software models. Latest techniques focus mainly on functional aspects of the generated code, e.g., how Java code is generated from UML diagrams. However, aspects of quality which do not come from the technical requirements are rarely considered. An important aspect of quality represents among other things the accessibility of web pages. Legislation in Austria requires a barrier-free access to official internet appearances for humans with special needs. This should have been implemented until 1st of Jan. 2008. In order to determine the accessibility of web applications so called Web Accessibility Guidelines (WAG) have been developed. For example, there are the WCAG (Web Content Accessibility Guidelines) of the W3C, which already became generally accepted as a de facto web accessibility standard. In order to fulfil the conformity with the legal regulations demanded above, public internet appearances must correspond to the respectively demanded guidelines. In this diploma thesis it is evaluated, in which way modern component frameworks and code generators fulfil the requirements resulting from the given guidelines. Therefore, WAG conform static html pages are used as a starting point. These static pages are developed to dynamic web applications by means of chosen component frameworks and code generators. The result of the generation process of each chosen development technology is finally rendered in form of html pages in the browser. These rendered pages are evaluated by means of WAG evaluation tools (test programs for the WAG conformance) regarding guidelines selected before. The results of this conformance evaluation demonstrate, to which extent the selected Web accessibility guidelines are violated. This evaluation also highlights which of the used development environments are better in terms of WAG conformance.

Inhaltsverzeichnis

1 Einleitung

Die Nutzung des Internets dringt in immer mehr Bereiche des geschäftlichen und sozialen Lebens vor. Dabei werden oft Web-Anwendungen für Zwecke des e-*commerce* und *e-government* bereitgestellt. Damit stehen alternative Möglichkeiten zu Erledigung von Handel und Behördengängen zur Verfügung, bzw. können diese dadurch auch gänzlich substituiert werden. Obwohl diese Technologie für viele Menschen große Zeitvorteile mit sich bringt, gibt es auch den Nachteil der Ausschließung all jener, die diese Technologie nicht in der gleichen Weise, wie die Durchschnittsanwender nutzen können. Eine Dimension dieses *digital-divide*, stellen Barrieren bei der Nutzung von Web-Inhalten und Web-Anwendungen dar. Diese Barrieren kommen dadurch zustande, da Menschen mit besonderen Bedürfnissen das Web auf eine andere Art und Weise nutzen. Zum Beispiel können blinde Menschen ohne besondere technische Hilfsmittel keine Webseiten lesen. Auch Menschen mit Hörschwächen und motorischen Einschränkungen können bei ihrer Art der Nutzung von Web-Inhalten, auf Barrieren treffen. Diese Barrieren sind deshalb vorhanden, weil bei der Erstellung der Inhalte nicht auf diesen Umstand geachtet worden ist. Es ist beispielsweise so, als ob ein Architekt bei der Planung eines Bahnhofes den Einbau von Aufzügen vergisst. Aber im Gegensatz zu diesem realen Beispiel, fällt dem Standardnutzer die fehlende Barrierefreiheit in der virtuellen Welt des Internets nicht auf. Aber Menschen mit besonderen Bedürfnissen sind sehr wohl von diesen Barrieren betroffen und werden dadurch von der Nutzung von Web-Inhalten oder -Anwendungen ausgeschlossen. Diese Problematik wurde von Gesetzgebern und Standardisierungsgremien erkannt. In Folge entstanden Gesetze und Richtlinien, welche die Erstellung von barrierefreien Web-Inhalten und -Anwendungen gesetzlich vorschreiben, bzw. Leitlinien für die Entwicklung von Web-Inhalten, zur Verfügung stellen. In den USA ist die Barrierefreiheit von Web-Inhalten gesetzlich in der so genannten „Section 508“ geregelt. In Europa gibt es verschiedenste Regelungen, die sich aber oft an den Leitlinien einer Untergruppe der W3C orientieren. Es handelt sich dabei um die Leitlinien der WAI (Web Accessibility Initiative). Die Leitlinien der WAI haben sich in der Branche als ein de facto Standard etabliert. Beispielsweise gibt es die WCAG (Web Content Accessibility Guidelines), welche Richtlinien für die Erstellung von barrierefreien Web-Inhalten enthalten. Heute werden für die zeiteffiziente Entwicklung von Web-Anwendungen komponentenbasierte Frameworks oder Code-Generatoren auf Basis von modellgetriebener Entwicklung verwendet. Die Entwicklungstechniken nehmen

daher Methoden an, die auch bei der Entwicklung von Desktop-Anwendungen bereits üblich sind. Diese modernen Softwareentwicklungstechniken orientieren sich aber hauptsächlich an funktionalen Qualitätsaspekten. Die Qualität des generierten Codes hinsichtlich des Gesichtspunktes der Barrierefreiheit ist dabei nicht immer garantiert. Somit haben moderne Softwareentwicklungstechniken auch Auswirkungen auf die Barrierefreiheit der damit erstellten Web-Anwendungen.

1.1 Problemstellung

Die Barrierefreiheit von Web-Inhalten ist für Web-Auftritte des öffentlichen Sektors, in vielen Ländern bereits gesetzlich verankert. Da dies in Zukunft auch für die kommerziellen Web-Auftritte zu erwarten ist, bzw. die Geschäfts-Etikette dies sowieso verlangen würde, besteht für Web-Agenturen erhöhter Handlungsbedarf, jene Softwareentwicklungstechniken anzuwenden, die barrierefreie Web-Inhalte bzw. Web-Anwendungen generieren können. Beziehungsweise müssen die Auftraggeber von Entwicklungsprojekten den Qualitätsaspekt der Barrierefreiheit im Lastenheft einfordern. Zu dem stehen die Hersteller von modellbasierten Code-Generatoren vor der Herausforderung, geeignete Komponentenbibliotheken für ihre Werkzeuge auszuwählen, da sich ihre Werkzeuge auf einer höheren Abstraktionsebene befinden und daher die verwendeten Komponenten selbst schon möglichst barrierefrei gestaltet sein sollten. Die Frage ist nun, welche modernen Komponenten-Frameworks und modellbasierte Code-Generatoren, barrierefreien Code generieren können, bzw. die Entwicklung von barrierefreien Web-Anwendungen erleichtern?

1.2 Lösungsansatz

In dieser Diplomarbeit soll diese Frage mit Hilfe einer Evaluierung von modernen Softwareentwicklungstechniken geklärt werden. Dabei werden zunächst zwei komponentenbasierte Frameworks und zwei modellbasierte Code-Generatoren ausgewählt. Mit diesen vier Entwicklungstechniken wird jeweils die gleiche Web-Anwendung (Referenzanwendung) implementiert. Diese Referenzanwendung existiert bereits als statischer Prototyp und ist weitestgehend barrierefrei. Dieser statische Prototyp wird mit jeder der vier gewählten Entwicklungstechniken, dynamisch weiter entwickelt und als Web-Anwendung abgebildet. Für jede dieser generierten Web-Anwendungen, wird eine Stichprobe der sich im Browser ergebenden HTML-Seiten gespeichert und hinsichtlich Barrierefreiheit evaluiert. Als Evaluierungskriterien werden die WCAG1.0 der Stufe A und AA herangezogen. Da die Referenzanwendung

für alle vier Entwicklungstechniken gleich ist, aber der generierte Code der jeweiligen Web-Anwendung anders aussehen wird, sind unterschiedliche Ergebnisse, in Bezug auf die Barrierefreiheit zu erwarten. Auf Grund der Evaluierungsergebnisse kann dann geschlossen werden, wie gut die jeweilige Entwicklungstechnik für die Erstellung von barrierefreien Web-Anwendungen geeignet ist.

1.3 Aufbau

In Kapitel 2 wird die Bedeutung der Barrierefreiheit im Web erklärt und es werden gesetzliche Bestimmungen dazu erläutert. Einige wichtige Richtlinien und Empfehlungen zur Barrierefreiheit von Web-Inhalten, werden in weiterer Folge genauer vorgestellt. In diesem Kapitel wird auch ein Überblick über die verschiedenen Typen von Web-Inhalten gegeben, die von grafischen Web-Browsern dargestellt werden können. Dabei werden mögliche Barrieren aufgezeigt. Am Ende des Kapitels wird noch kurz auf assistierende Technologien eingegangen, die von Menschen mit besonderen Bedürfnissen zur Nutzung von Web-Inhalten verwendet werden können.

Das Kapitel 3 beschäftigt sich mit der Überprüfung von bestehenden Web-Seiten bzw. -Anwendungen. Es werden kommerzielle und frei verfügbare Prüfprogramme vorgestellt, die zur Evaluierung der Barrierefreiheit verwendet werden können. Es wird erklärt welche Typen von Prüfprogrammen es gibt und welche Verletzungen der Barrierefreiheit grundsätzlich automatisch festgestellt werden können. Am Ende des Kapitels wird auf Methoden zur Evaluierung der Barrierefreiheit eingegangen.

In Kapitel 4 wird die Problemstellung und der Lösungsansatz genauer erklärt, es werden die Auswirkungen von modernen Softwareentwicklungstechniken auf die Barrierefreiheit von Web-Anwendungen näher erläutert. Ausgehend davon werden die im Lösungsansatz aufgezeigten Komponenten-Frameworks und Code-Generatoren ausgewählt, welche für die Implementierung herangezogen werden. Es wird erklärt wie die Referenzanwendung aufgebaut ist und auf welche Weise diese Referenzanwendung mit den ausgewählten Entwicklungstechniken, jeweils als dynamische Web-Anwendung implementiert wird.

Das Kapitel 5 beschäftigt sich mit der Evaluierung der Barrierefreiheit. Es wurden vier, fast funktionsgleiche Web-Anwendungen erstellt. Deren generierter HTML-Code, dient als Basis für eine Evaluierung. In diesem Kapitel werden die Bedingungen und die Kriterien dieser Evaluierung genauer erklärt. Prüfwerkzeuge werden vorge-

stellt, der Ablauf der Evaluierung, sowie das Bewertungsschema und das Berichtformat, werden definiert.

Die Dokumentation der Evaluierungsergebnisse erfolgt in Kapitel 6 . Das Kapitel beginnt mit einer Zusammenfassung der numerischen Ergebnisse. Danach werden die gesammelten Detailergebnisse in Tabellen aufgelistet. Wie die Aufbereitung dieser Ergebnisse erfolgte und wie diese Ergebnisse interpretiert wurden, wird dann in Folge erklärt. Am Ende des Kapitels findet sich eine Gegenüberstellung der Entwicklungstechniken, in welcher zusammenfassend die Stärken und Schwächen der Entwicklungstechniken aufgezeigt werden.

2 Barrierefreiheit von Web-Inhalten

Unter Barrierefreiheit von Web-Inhalten, in Englisch „Web Accessibility“, wird der barrierefreie Zugang zu Webinhalten für *alle* Nutzergruppen verstanden. Dies beinhaltet im speziellen Menschen mit besonderen Bedürfnissen bzw. ältere Menschen.

„Accessibility can be defined as the quality of a web site that makes it possible for people to use it- to find it navigable and understandable – even when they are working under limiting conditions or constraints.“ [Weis04]

„Barrierefreies Webdesign zielt dementsprechend darauf ab, Inhalte und Interaktionen im Netz für (möglichst) alle Nutzergruppen und Endgeräte zugänglich zu machen.“ [Radt06]

Barrierefreiheit im Web ist ein relativ neues Feld in der IT. Als das Internet entstand gab es noch kaum Unterstützung für Menschen mit besonderen Bedürfnissen [Weis04]. In 1997 etablierte sich eine der ersten Interessensgruppe für Menschen mit besonderen Bedürfnissen, die WAI (Web Accessibility Initiative) des W3C (World Wide Web Consortium). Das Ziel der WAI ist es, den barrierefreien Zugang für alle Internetnutzer zu ermöglichen.

2.1 Gesetzgebung zur Barrierefreiheit im Web

Heute gibt es weltweit eine Reihe von gesetzlichen Regelungen [Radt06], welche den gleichberechtigten Zugang zu Informationen im Internet garantieren sollen. Einen wichtigen Anstoß dazu lieferte 1993 die UNO Generalversammlung mit einer Resolution, in welcher der gleichberechtigte Zugang zu Information und Kommunikation für Menschen mit physischen Behinderungen gefordert wurde. 1994 wurde das W3C gegründet. Dieses Gremium verfolgt seitdem, neben der Standardisierung von Web-Technologien, auch die Entwicklung von Richtlinien für den barrierefreien Zugang zu Web-Inhalten. 1998 wurde in den USA die *Section 508 of the Rehabilitation Act Amandement* verabschiedet. Mit diesem Gesetz verpflichteten sich die US-Regierung und ihre Behörden, zur Einhaltung von bestimmten Zugänglichkeitsanforderungen. Neben den USA gab es auch in anderen Ländern Bemühungen, eigene Gesetze zur

Barrierefreiheit im Internet zu erlassen. Nachfolgend werden einige Beispiele für derartige Gesetze kurz beschrieben.

2.1.1 USA

In den USA wird die Barrierefreiheit von Webseiten über die als *Section 508* bekannte Richtlinie gesetzlich geregelt [Weis04]. Genau genommen handelt es sich dabei um die: *Section 508 of the U.S. Federal Government Rehabilitation Act Amendments of 1998*. Diese ergänzen den *Workforce Rehabilitation Act* aus dem Jahre 1973. Das Ziel dieses Gesetzes war, Anforderungen für Arbeitsplätze zu definieren, welche die Diskriminierung von Menschen mit besonderen Bedürfnissen beenden sollte. In den Erweiterungen der *Section 508* wurde das ursprüngliche Gesetz an den Stand der Technik angepasst und diese sind seit 2001 gültig.

In *Section 508* wird die Barrierefreiheit von Informationstechnologien gefordert, welche von den US-Bundesbehörden entwickelt oder verwendet werden. Das Gesetz gilt für:

- Behörden und Agenturen der Regierung
- Auftragnehmer, die für diese Behörden tätig sind
- Von der Regierung finanzierte Aktivitäten

In §1194.22 der *Section 508* [Sect08] werden 16 Anforderungen an Web-basierte Informationstechnologien aufgelistet. Die Anforderungen sind knapp formuliert, eindeutig und verpflichtend.

Die Signalwirkung an die Industrie kommt klar heraus [Weis04]. Wer Regierungsaufträge erhalten möchte, muss sich an diese Richtlinien halten. Davon können dann auch andere Sektoren profitieren.

2.1.2 Europäische Union

Die Europäische Kommission startete im Jahr 2000 mit der *eEurope* Initiative. Das Ziel war, den Übergang der EU in eine wissensbasierte Gesellschaft voranzubringen. Außerdem sollte das Programm die Grundlage für mehr Wachstum, mehr Arbeitsplätze und einen für alle Bürger besseren Zugang zu den neuen Informationsdiensten schaffen.

Die erste Phase der Initiative war der *eEurope 2002 Aktionsplan*. Dieser Plan hatte drei Hauptziele:

- Ein billigeres, schnelleres und mehr sicheres Internet
- Investitionen um die Fähigkeiten der Menschen zu steigern
- Die Nutzung des Internets zu verbreiten

Was den Bereich *Accessibility* angeht wird im Aktionsplan 2002 festgelegt [Thee05], dass die Mitgliedsländer bis zum Ende 2001 die Einhaltung der *Web Content Accessibility Guidelines* (WCAG) der WAI, für öffentliche Web-Seiten sicherstellen sollen. Darüber hinaus wird auch ausdrücklich festgelegt, dass öffentliche Web-Seiten auch für Menschen mit besonderen Bedürfnissen zugänglich sein müssen.

Im Jahr 2005 gab es eine Mitteilung [Komm05] der Europäischen Kommission über *eAccessibility*, KOM (2005)424. Die Kommission stellt fest, dass *eAccessibility* noch zu wenig in nationalen Gesetzen Berücksichtigung findet und macht auf den *digital divide* zwischen verschiedenen Nutzergruppen aufmerksam. Sie ruft zu einer freiwilligen Verbesserung der Situation auf. Die Forderungen sind:

- Sicherstellung von *eAccessibility* Anforderungen im öffentlichen Beschaffungswesen
- Zertifizierungsmechanismen für ICT-Produkte entwickeln
- Bestehende Gesetze für *eAccessibility* besser nutzen

Heute ist *eAccessibility* ein Teil der neuen: *European i2010 initiative on e-Inclusion*.

"To be part of the Information Society, includes an active strategy to improve accessibility to the Information Society for all potentially disadvantaged groups. In order to bridge the eAccessibility gap." [Thee08]

Das Ziel der *e-Inclusion* Initiative ist die Entwicklung eines Gesetzesvorschlages für die Verbesserung der *eAccessibility*. Konsultationen dazu wurden in 2008 abgehalten [Thee08].

Im Jahr 2008 ergab eine Studie [Euro09] im Auftrag der EU-Kommission ein ernüchterndes Ergebnis. Laut der Studie „Measuring progress of eAccessibility in Europe" (MeAC), erreichten nur 60% der behördlichen Web-Auftritte und 95% der Web-Auftritte des privaten Sektors, nicht einmal den Level A der WCAG1.0. Die Rechts-

lage bezüglich Barrierefreiheit im Web, ist in der EU nicht homogen. Es gibt zwar viele Initiativen, Empfehlungen und Entschließungen von diversen EU-Gremien aber die Mitgliedsstaaten haben diese Vorschläge unterschiedlich bzw. noch nicht vollständig in nationales Recht umgesetzt. Da es derzeit zur *eAccessibility* noch keine rechtlich verbindliche „EU-Richtlinie“ gibt, ist es nötig die jeweils nationalen Bestimmungen zu kennen.

2.1.3 Österreich

Die Bundesverfassung enthält mit dem Artikel 7 einen Gleichheitsgrundsatz welcher auch ein eigenes Diskriminierungsverbot für behinderte Menschen enthält. Bund, Länder und Gemeinden verpflichten sich die Gleichstellung aller Menschen in allen Bereichen des Lebens zu gewährleisten.

Im Behindertengleichstellungsgesetz (BGStG) werden die Bestimmungen aus Artikel 7 der Verfassung genauer definiert, außerdem wird hier bereits festgelegt, dass für Angebote im Internet die WAI-Leitlinien herangezogen werden. Seit 1.1.2006 sind diese Leitlinien für behördliche und generell auch für nicht- behördliche Webangebote anzuwenden. Wobei das Gesetz zulässt, dass die Rahmenbedingungen für eine Zumutbarkeitsprüfung im Einzelfall beurteilt werden. Bei einer Klage auf Schadenersatz ist aber ein Schlichtungsverfahren vorgelagert. [Digi08]

Im E-Government-Gesetz (E-GovG) wird in §1 Abs.3 gefordert:

„Bei der Umsetzung der Ziele dieses Bundesgesetzes ist Vorsorge dafür zu treffen, dass behördliche Internetauftritte, die Informationen anbieten oder Verfahren elektronisch unterstützen, so gestaltet sind, dass internationale Standards über die Web-Zugänglichkeit auch hinsichtlich des barrierefreien Zugangs für behinderte Menschen eingehalten werden.“ [Bund08]

Dieser Abs. 3 ist am 1.1.2008 in Kraft getreten. In § 3 Abs.1 *Zustelldiensteverordnung* wird auf die *Web Content Accessibility Guidelines 1.0* der Stufe A hingewiesen. Diese Verordnung sorgt für eine barrierefreie Umsetzung bei der elektronischen Zustellung. [Digi08]

2.1.4 Deutschland

Im Jahr 2002 wurde in Deutschland das *Behindertengleichstellungsgesetz (BGG)* verabschiedet. In diesem Gesetz wird in §11, die Barrierefreiheit von Internetauftrit-

ten der öffentlichen Verwaltungsstellen gefordert. Wie in einem Auszug aus §11 zu lesen ist. [Bund02]

„Träger öffentlicher Gewalt im Sinne des § 7 Abs. 1 Satz 1 gestalten ihre Internetauftritte und –angebote sowie die von ihnen zur Verfügung gestellten grafischen Programmoberflächen, die mit Mitteln der Informationstechnik dargestellt werden, nach Maßgabe der nach Satz 2 zu erlassenden Verordnung schrittweise technisch so, dass sie von behinderten Menschen grundsätzlich uneingeschränkt genutzt werden können.“ [Bund02]

Die Bedingungen der uneingeschränkten Nutzung werden dabei in einer separaten Verordnung definiert. Diese Verordnung nennt sich *Barrierefreie Informationstechnik-Verordnung* (BITV) und regelt die Umsetzung von §11 des BGG. Die BITV besteht aus drei Teilen:

- Verordnungstext; Enthält Allgemeine Bestimmungen zu Geltungsbereich, Zielgruppen und Fristen
- Anlage 1; Enthält konkrete Anforderungen bezüglich Barrierefreiheit
- Anlage 2; Enthält ein Glossar

Der §4 der Verordnung legt die Umsetzungsfristen fest, seit Anfang 2006 müssen alle Bundesverwaltungsstellen diese Verordnung erfüllen. Nach §5 wird die Verordnung alle drei Jahre darauf geprüft, ob sie noch dem Stand der Technik entspricht.

Die Bedingungen und Anforderungen an die Barrierefreiheit in der BITV, orientieren sich an den WCAG1.0 der WAI. [Barr06]

2.2 Empfehlungen und Richtlinien zur Barrierefreiheit

Die Barrierefreiheit im Web ist abhängig von verschiedenen Komponenten. Es gibt verschiedene Typen von Richtlinien, die jeweils auf eine bestimmte Komponente ausgelegt sind. Welche Komponenten dies sind, welche Typen von Richtlinien es gibt und was einige wichtige Richtlinien beinhalten wird nachfolgend beschrieben.

2.2.1 Abhängigkeit der Barrierefreiheit

Damit das Web für Menschen mit besonderen Bedürfnissen barrierefrei ist, müssen nach [Chis05] verschiedene Komponenten zur Erstellung und Interaktion von Web-Inhalten zusammenarbeiten. Es sind dies folgende Komponenten:

- *Inhalte:* Das sind die Informationen die in Web-Seiten und Web-Anwendungen enthalten sind. Zum Beispiel: Text, Bilder, Audio. Der Code und das Markup für die Beschreibung der Struktur und der Präsentation,..
- *Benutzeragenten:* Werden von den Benutzern verwendet, um mit den Inhalten zu interagieren. Beispiele: Web-Browser, Medienwiedergabe, Assistierende Technologien,..
- *Assistierende Technologien:* Sind Geräte oder Software, welche in Verbindung mit einem Browser benutzt werden können. Zum Beispiel: Screen Reader, alternative Keyboards und Zeigergeräte,..
- *Benutzer:* Sind Menschen die Web-Inhalte mit Benutzeragenten abrufen. Die Benutzer haben verschiedene Erfahrungen und Fähigkeiten.
- *Autoren, Entwickler:* Menschen die Web-Inhalte erstellen oder Web-Anwendungen entwickeln.
- *Evaluierungswerkzeuge*: Programme die zur Überprüfung der Barrierefreiheit von Webinhalten verwendet werden.
- *Autorenwerkzeuge:* Software die für die Erstellung bzw. Änderung von Web-Inhalten oder Web-Anwendungen verwendet wird. Zum Beispiel: WYSIWYG-Werkzeuge, CMS, Blogging-Werkzeuge,...

Von der WAI wurden Richtlinien entwickelt, welche diese Komponenten berücksichtigen. Für die Erstellung von barrierefreien Inhalten wurden die WCAG entwickelt. Für die Evaluierung der Barrierefreiheit von Autorenwerkzeugen wurden die *Authoring Tool Accessibility Guidelines* (ATAG) entwickelt. Da auch Menschen mit besonderen Bedürfnissen Web-Inhalte erstellen, müssen die Autorenwerkzeuge selbst auch barrierefrei sein. Für die Evaluierung der Barrierefreiheit von Benutzeragenten wurden die *User Agent Accessibility Guidelines* (UAAG) entwickelt. Sie richten sich hauptsächlich an die Entwickler von Web-Browsern, assistierenden Technologien und anderen Benutzeragenten.

Die Richtlinien der WAI ermöglichen ein einwandfreies zusammenarbeiten dieser Komponenten. In Abbildung 2.1 ist das zusammenwirken der verschiedenen Komponenten und deren Einbettung in die verschiedenen Richtlinien der WAI dargestellt.

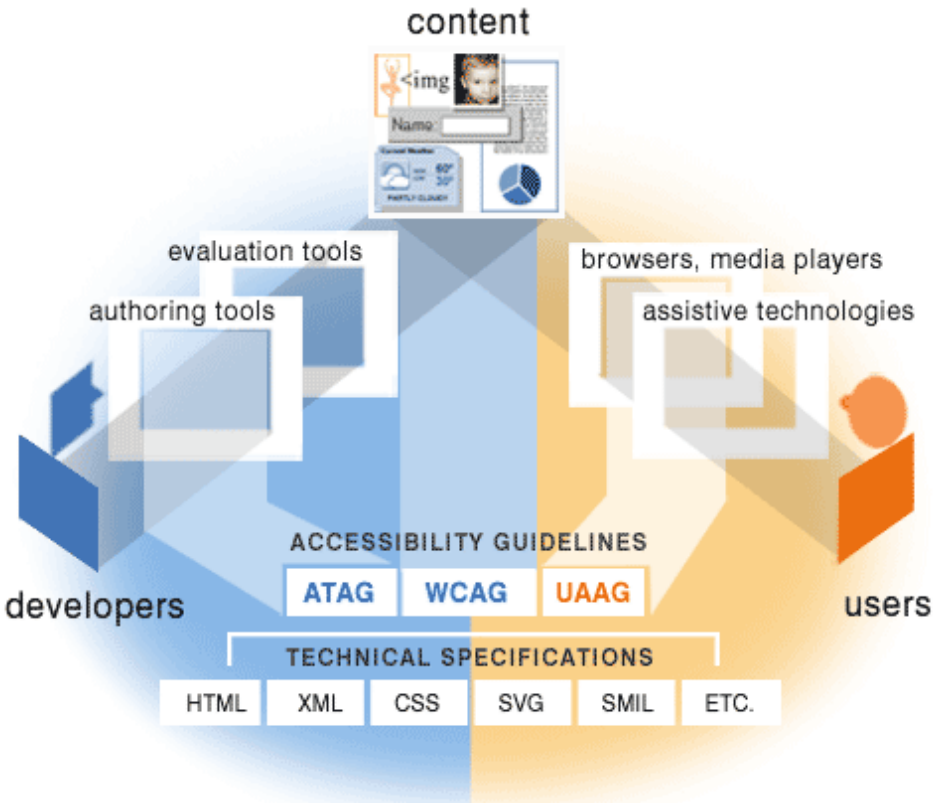

Abbildung 2.1 Komponenten der Barrierefreiheit im Web, [Henr05]

Im Folgenden werden die WCAG der WAI genauer vorgestellt, da sie für die spätere Evaluierung von besonderer Bedeutung sind. Die Richtlinien für barrierefreie Web-Inhalte der *Section 508* werden auch kurz erörtert. Zum Schluss wird noch eine der neueren Richtlinien der WAI, die WAI-ARIA erörtert. Dabei geht es um die Barrierefreiheit von *Rich Internet Applications* (RIAs).

2.2.2 WAI – Web Content Accessibility Guidelines 1.0

Im Jahr 1999 wurden die *Web Content Accessibility Guidelines 1.0 (WCAG1.0)* von der W3C veröffentlicht [W3cw99]. Diese Richtlinien bieten Betreibern und Entwicklern bzw. Designern von Webseiten eine Hilfestellung, wie Webseiten für Menschen mit Einschränkungen besser zugänglich gemacht werden können. Die WCAG1.0 beinhalten 14 Richtlinien welche wiederum mehrere Prüfpunkte beinhalten. Diese Prüfpunkte sind in Prioritätsklassen eingeteilt. Klasse A beschreibt die Prüfpunkte der Kategorie: „Muss erfüllt sein“, Klasse AA beschreibt die Prüfpunkte der Kategorie: „Sollte erfüllt sein“ und Klasse AAA beschreibt die Prüfpunkte der Kategorie: „Ist

sinnvoll“. Die WCAG1.0 haben sich als ein Accessibility de facto Standard etabliert, die meisten Prüfprogramme unterstützen die WCAG1.0, außerdem werden diese Richtlinien in vielen Rechtsnormen als Maßstab herangezogen. Entweder wird in den Gesetzgebungen direkt darauf verwiesen oder es wird eine abgewandelte Form definiert. Zum Beispiel gibt es Ähnlichkeiten zwischen *Section 508*, BITV und den WCAG1.0. Aufgrund ihrer besonderen Bedeutung werden die 14 Richtlinien der WCAG1.0 anschließend kurz beschrieben. [Weis04]

1. Bereitstellung von Äquivalenten für audiovisuelle Inhalte

„Provide content that, when presented to the user, conveys essentially the same function or purpose as auditory or visual content.“ [W3cw99]

Spezielle Inhalte wie: Filme, Bilder, Applets usw. können von manchen Benutzern nicht direkt wahrgenommen werden. Diese Nutzer können aber mitunter eine äquivalente Form des Inhaltes wahrnehmen. Diese äquivalente Form muss dabei denselben Zweck erfüllen wie der Originalinhalt. Beispielsweise sind Text-Äquivalente für Bilder, Diagramme, Videos usw. eine gute Alternative. Text-Äquivalente haben den Vorteil, dass sie unter Verwendung der verschiedensten Technologien, unterschiedlichen Gruppen von Menschen mit Einschränkungen zugänglich gemacht werden können. Alternativer Text für audiovisuelle Objekte erlaubt zudem die Erfassung der Inhalte durch Suchmaschinen. [W3cw99]

2. Beachten sie die Wirkung von Farbe

„Ensure that text and graphics are understandable when viewed without color.“ [W3cw99]

Menschen die bestimmte Farben nicht voneinander unterscheiden können, haben Probleme bei der Wahrnehmung von Inhalten, die nur Farbe als Informationsträger nützen. Geräte die ohne Farbe arbeiten, oder keinen visuellen Output liefern können die Information dann nicht darstellen. Wenn Hintergrund- und Vordergrundfarbe kaum Kontrast aufweisen, kann dies auch zu Problemen bei der Wahrnehmung führen. [W3cw99]

3. Markup und Stylesheets korrekt verwenden

„Mark up documents with the proper structural elements. Control presentation with style sheets rather than with presentation elements and attributes.“ [W3cw99]

Markup sollte nicht für Layout- und Präsentationseffekte genutzt werden (z.B. Tabellen für Layoutzwecke), da dies für die Benutzer von Screenreadern oder anderen Werkzeugen Probleme bereitet. Es ist dann für die Benutzer schwer die Seite zu verstehen bzw. in der Seite zu navigieren. Konstrukte die in älteren Browsern für Formatierungszwecke verwendet bzw. missbraucht wurden, sollen nicht mehr verwendet werden. Es wird empfohlen CSS für Layout und Präsentationszwecke zu verwenden. Es wird empfohlen Dokumenttypen zu verwenden, die veröffentlichten, formalen Grammatiken entsprechen. [W3cw99]

4. Machen Sie Änderungen der natürlichen Sprache kenntlich

„Use markup that facilitates pronunciation or interpretation of abbreviated or foreign text.“ [W3cw99]

Die Änderung der natürlichen Sprache soll von den Entwicklern durch Markup kenntlich gemacht werden (z.B. mit dem *lang* oder *xml:lang* Attribut). Dann können Sprachgeneratoren oder Blindenschrift-Geräte auf die neue Sprache wechseln. Die dominierende Sprache soll angegeben werden (z.B. im HTTP-Header). Abkürzungen und Akronyme sind auch mittels Markup zu kennzeichnen bzw. sollen ausformuliert werden. Dafür kann in HTML das *title-attribute* der Elemente ABBR und ACRONYM verwendet werden. [W3cw99]

5. Gewährleisten sie das geschmeidige transformieren von Tabellen

„Ensure that tables have necessary markup to be transformed by accessible browsers and other user agents.“ [W3cw99]

Tabellen sollten nur für Daten verwendet werden und nicht für Layoutzwecke, da Layouttabellen in linearisierter Form (Tabellenzellen werden sequentiell in Textform ausgegeben), oft keinen Sinn ergeben. Für Benutzer von Screenreadern bringen Tabellen spezielle Probleme mit sich, da die Linearisierung des Tabelleninhaltes unterschiedlich erfolgen kann. Bei Datentabellen sollen Zeilen- und Spaltenüberschriften gekennzeichnet werden. Bei komplexeren Datentabellen (z.B. mit mehreren Ebenen) soll Markup dazu verwendet werden um Daten- von Überschriftenzellen unterscheiden zu können. [W3cw99]

6. Es ist sicherzustellen, dass Seiten mit neuartigen Technologien geschmeidig transformieren

„Ensure that pages are accessible even when newer technologies are not supported or are turned off.“ [W3cw99]

Wenn neue Technologien für Inhalte verwendet werden, so ist Sorge zu tragen, dass die Inhalte auch bei Abschaltung dieser Technologien noch zur Verfügung stehen. Auch bei der Verwendung von älteren Browsern, welche diese Technologien vielleicht nicht unterstützen, sollten die Inhalte noch erfassbar sein. Z.B. sollen HTML-Dokumente auch bei Abschaltung der Stylesheets noch lesbar sein. Oder bei der Änderung von dynamischen Inhalten, soll auch der Äquivalente Inhalt mit geändert werden. Bei Verwendung von Scripts, Applets oder anderen dynamischen Objekten, ist dafür zu sorgen, dass auch bei Abschaltung dieser Objekte, die Webseite immer noch verwendbar ist. [W3cw99]

7. Die Benutzer sollen die Kontrolle über sich zeitlich ändernde Inhaltselemente erhalten

„Ensure that moving, blinking, scrolling, or auto-updating objects or pages may be paused or stopped.“ [W3cw99]

Bewegter oder blinkender Text kann von Menschen mit bestimmten visuellen oder kognitiven Einschränkungen nicht richtig wahrgenommen werden. Auch Screenreader können bewegten Text nicht lesen. Bildschirmflackern ist zu vermeiden, da Menschen mit photosensitiver Epilepsie davon Anfälle bekommen können. Blinkende oder bewegte Objekte sollen vermieden werden, solange die Benutzer keine Möglichkeit haben, solche Objekte abzuschalten. [W3cw99]

8. Eingebettete Benutzerschnittstellen sollen direkt zugänglich gemacht werden.

„Ensure that the user interface follows principles of accessible design: device-independent access to functionality, keyboard operability, self-voicing, etc.“ [W3cw99]

Eingebettete Objekte (z.B. Scripts, Applets oder Flash-Inhalte) können eigene Schnittstellen haben. Diese Schnittstellen müssen zugänglich gemacht werden. Ist es nicht möglich diese Schnittstellen zugänglich zu machen, muss eine Alternative angeboten werden. Wenn eingebettete Objekte wichtige Inhalte enthalten, dann sollen diese auch direkt zugänglich sein, bzw. Assistierende Technologien sollen dann auch darauf zugreifen können (z.B. ein Applet welches nur über die Maus gesteuert werden kann, sollte auch mit der Tastatur steuerbar sein). [W3cw99]

9. Das Design soll geräteunabhängig sein

„Use features that enable activation of page elements via a variety of input devices.“ [W3cw99]

Die Benutzer sollen die Möglichkeit haben, mit den bevorzugten Eingabegeräten oder Ausgabegeräten auf die Dokumente zugreifen zu können. Z.B. können bei Formularfeldern Probleme auftreten, wenn diese nicht mit der Tastatur aktiviert werden können. Web-Seiten die über die Tastatur bedient werden, können normalerweise auch über Spracheingabe oder andere Schnittstellen leichter zugänglich gemacht werden. Es wird empfohlen Client-Seitige, statt Server-Seitige *image maps* (das sind Regionen in Bildern, die aktiv auf Benutzereingabe reagieren) zu verwenden. Da Client-Seitige *image maps* unabhängig vom Zeigergerät sind. Für *image maps* wird auch die Verwendung von alternativen Text-Links empfohlen. [W3cw99]

10. Erfüllung der Abwärtskompatibilität mit vorläufigen Lösungen

„Use interim accessibility solutions so that assistive technologies and older browsers will operate correctly.“ [W3cw99]

Bei älteren Browsern ist es z.B. nicht möglich zu leeren Textboxen zu navigieren oder ältere Screenreader haben das Problem, dass Listen von Links als ein einziger Link gelesen werden. Probleme entstehen wenn der Zugriff auf aktive Elemente aufgrund der Verwendung von alten Benutzeragenten nicht möglich ist. Das erscheinen von Pop-Up-Fenstern oder anderen Fenstern, ohne das der Benutzer davon Kenntnis hat, soll vermieden werden. Es ist auch dafür zu sorgen, dass allen Kontrollelementen von Formularen explizite Beschriftungen (z.B. ein Label) zugeordnet werden.

Die Arbeitsgruppe der WCAG erachtet diese Richtlinie als vorläufig noch notwendig. Zumindest solange bis moderne Web-Technologien zur Verfügung stehen, welche die Punkte unter dieser Richtlinie bereits berücksichtigen. [W3cw99]

11. W3C-Richtlinien und –Technologien sollen verwendet werden

„Use W3C technologies (according to specification) and follow accessibility guidelines. Where it is not possible to use a W3C technology, or doing so results in material that does not transform gracefully, provide an alternative version of the content that is accessible.“ [W3cw99]

Die Verwendung von W3C-Technologien wie HTML, CSS etc. wird empfohlen, weil diese Technologien bereits Zugänglichkeits-Features beinhalten. Weil diese Technologien schon in der Designphase auf Zugänglichkeit geprüft werden und weil diese Technologien in einem offenen Prozess entwickelt werden. Bei anderen Technologien wie PDF, Flash, usw. werden Plug-Ins oder eigene Anwendungen benötigt. Diese Technologien haben den Nachteil, dass sie nicht mit Standardbenutzeragenten bzw. assistierenden Technologien genutzt werden können. Wenn derartige Technologien verwendet werden, müssen die Inhalte auf eine alternative Art und Weise zugänglich gemacht werden. Z.B. können Formate wie: PDF, DOC usw. nach HTML oder XML konvertiert werden. [W3cw99]

12. Kontext und Orientierung

„Provide context and orientation information to help users understand complex pages or elements.“ [W3cw99]

Zusammengehörende Elemente sind übersichtlich und in Gruppen anzuordnen. Kontextinformation für Elemente soll zur Verfügung gestellt werden. *Frames* sind zu betiteln bzw. ihr Zweck und ihre Beziehung untereinander soll beschrieben werden. [W3cw99]

13. Verwenden Sie übersichtliche Navigationsmechanismen

„Provide clear and consistent navigation mechanisms – orientation information, navigation bars, a site map, etc. – to increase the likelihood that a person will find what they are looking for at a site.“ [W3cw99]

Menschen mit kognitiven Behinderungen oder Blindheit benötigen klare Navigationsmechanismen, diese kommen aber auch anderen Benutzern zugute. Z.B. sollten die Ziele von Links klar formuliert werden. Es wird empfohlen Metadaten über die Seite zur Verfügung zu stellen (z.B. mit RDF). Auch Informationen über das allgemeine Layout, wie *site maps* oder Inhaltsverzeichnisse können eine gute Orientierung bieten. [W3cw99]

14. Dokumente sollen einfach verständlich und übersichtlich sein

„Ensure that documents are clear and simple so they may be more easily understood.“ [W3cw99]

Das Seitenlayout soll konsistent sein und die Sprache soll eindeutig und klar sein. Für Menschen mit kognitiven Behinderungen, die Probleme beim Lesen haben, ist

dies besonders wichtig. Daher soll die einfachste Formulierung gewählt werden, die noch angemessen erscheint. Bei Verwendung von Grafiken und Bildern sind Text-Äquivalente zur Verfügung zu stellen. Der Präsentationsstil soll über die ganze Seite hin konsistent sein. [W3cw99]

Die hier ausgeführte Beschreibung der einzelnen Richtlinien, ermöglicht einen ersten Einblick in die WCAG1.0. Für genauere Angaben zu den Prüfpunkten, wird auf das Originaldokument, WCAG1.0 [W3cw99] verwiesen. Weiters sind die Unterpunkte der einzelnen Richtlinien tabellarisch im Dokument [W3cc00] zusammengefasst. Basierend auf den WCAG1.0 wurden die Richtlinien weiterentwickelt, es gibt seit 2001 eine Version 2, die allerdings erst kürzlich ihren endgültigen Stand erreicht hat und seit dem zwölf mal überarbeitet wurde. Die Version WCAG2.0 [W3cw08a] vom April 2008 hat einen Stand erreicht, der laut W3C erste Testimplementierungen zulässt. Während der Erstellung dieser Diplomarbeit wurden die WCAG2.0 [W3ct08] von der WAI finalisiert, der Stand vom 11.12.2008 gilt nun als Web-Standard und soll die WCAG1.0 ablösen.

2.2.3 WAI – Web Content Accessibility Guidelines 2.0

Die WCAG2.0 [W3ct08] sind zwar eine Erweiterung der WCAG1.0 aber das Prinzip hat sich geändert. Die WCAG2.0 sind technologieunabhängiger und abstrakter in ihrer Ausprägung. Sie bestehen aus Richtlinien und Prinzipien die vier Hauptprinzipien sind:

- Wahrnehmbarkeit
    - Stellen Sie Textalternativen für Nicht-Text Inhalt zur Verfügung.
    - Stellen Sie Untertitel und Alternativen für Audio- und Videoinhalt zur Verfügung.
    - Machen Sie Inhalte adaptierbar und zugänglich für assistierende Technologien.
    - Verwenden Sie genügenden Kontrast, um Inhalte besser zu sehen und zu hören.
- Funktionalität
    - Machen Sie alle Inhalte zugänglich für die Tastatur.

  - Geben Sie Benutzern genügend Zeit um Inhalte zu lesen und zu benutzen.
  - Vermeiden Sie Inhalte, die Krämpfe verursachen (z.B. Epilepsie).
  - Helfen Sie den Benutzern bei der Navigation und Suche von Inhalten.

- Verständlichkeit
  - Machen Sie Text lesbar und verständlich.
  - Lassen Sie Inhalte auf vorhersagbare Arten erscheinen und anwenden.
  - Helfen Sie Benutzern beim Vermeiden und Beheben von Fehlern.

- Robustheit
  - Maximieren Sie die Kompatibilität mit den gegenwärtigen und zukünftigen Technologien.

Die Prüfpunkte wurden durch Erfolgskriterien der Stufe A, AA und AAA ersetzt. Insgesamt sind die WCAG2.0 komplexer aufgebaut und schwieriger anzuwenden als die WCAG1.0. In dieser Diplomarbeit werden für eine Evaluierung der Barrierefreiheit von Web-Anwendungen, die WCAG1.0 herangezogen. Da sie in den gängigen Prüfprogrammen bereits implementiert sind.

2.2.4 Richtlinien nach Section 508

Wie bereits in Abschnitt 2.1.1 erwähnt sind die gesetzlichen Bestimmungen zur Barrierefreiheit von Informationstechnologie in den USA im US-Bundes-Gesetz *Section 508* geregelt. Genau genommen sind all jene Belange, die Web-Inhalte und Web-Anwendungen betreffen, im „Subpart B, Technical Standards“ [Sect08] geregelt. Diese Bestimmungen lauten folgendermaßen:

(a) Für jedes Nicht-Text Element soll ein Textäquivalent zur Verfügung gestellt werden.

(b) Äquivalente Alternativen für Multimedia-Inhalte sollen geändert werden, wenn sich die Multimedia-Inhalte ändern.

(c) Webseiten sollten so gestaltet werden, dass alle Inhalte in Farbe auch ohne Farbe verfügbar sind.

(d) Dokumente sollen auch ohne Stylesheets lesbar bleiben.

(e) Für jede aktive Region einer serverseitigen *image map* sollen redundante Text-Links zur Verfügung gestellt werden.

(f) *Image maps* auf der Client-Seite sollen statt Serverseitigen *image maps* verwendet werden. Es sei denn, die Regionen können in einer vorhandenen geometrischen Form nicht definiert werden.

(g) Bei Datentabellen sollen Zeilen- und Spaltenbezeichnungen verwendet werden.

(h) Um in komplexen Tabellen mit mehreren logischen Ebenen von Zeilen- und Spalten-Bezeichnungen, die Datenzellen den entsprechenden Kopfzellen zuordnen zu können, soll entsprechendes Markup verwendet werden.

(i) Frames sollen betitelt werden um die Identifikation und Navigation sicherzustellen.

(j) Das Bildschirmflackern soll verhindert werden, es dürfen keine Elemente vorhanden sein, die im Bereich von 2Hz-55Hz ein flackern verursachen.

(k) Falls es nicht möglich ist die vorliegenden Richtlinien auf einer Seite einzuhalten, soll eine äquivalente Text-Seite zur Verfügung gestellt werden, deren Inhalte bei Änderungen der Hauptseite auch aktualisiert werden.

(l) Wenn Seiten Skript-Sprachen verwenden, um Inhalte darzustellen oder für Interaktionselemente, soll die Art der Informationen die das Skript liefert, mit Text beschrieben werden, welcher von Assistierenden Technologien gelesen werden kann.

(m) Wenn eine Seite ein Applet, Plug-In oder eine andere Applikation auf dem Client-System benötigt um den Seiteninhalt darzustellen, muss die Seite einen Link zu einem entsprechenden Applet bzw. Plug-In zur Verfügung stellen. Dieses Applet bzw. Plug-In muss mit den Punkten (a) bis (l) konform sein.

(n) Wenn Formulare verwendet werden, die online ausgefüllt werden sollen, müssen diese Formulare auch Assistierende Technologien unterstützen.

(o) Es soll möglich sein Navigationsbereiche zu überspringen.

(p) Wenn eine Antwort in einer limitierten Zeit benötigt wird, sollen die Benutzer darüber informiert werden und es soll genug Zeit zur Verfügung stehen um eine Fristverlängerung zu erlangen. [Sect08]

2.3 Rich Internet Applications und WAI-ARIA

Heute werden so genannte *Rich Internet Applications* (RIAs) immer öfter eingesetzt. Dieser Typ von Web-Anwendungen ermöglicht die Nutzung von erweiterten Funktionen, wie sie aus herkömmlichen GUI-Anwendungen bekannt sind. Dabei kommen Internet-Technologien wie JavaScript, Ajax, Flash, udg. zum Einsatz. Ajax bedeutet „Asynchronous JavaScript and XML“ [Stey08]. Die Technologie dahinter gibt es schon seit 1998 aber die Bezeichnung Ajax wurde erst 2005 etabliert. Das Prinzip von Ajax ist, dass der Client Daten vom Server *asynchron* zur Benutzerinteraktion anfordert. Dies geschieht mit Hilfe des „XMLHttpRequest-Object“. Diese angeforderten Daten werden dann mit Hilfe von JavaScript in das DOM (Document Object Model) der Web-Seite eingebaut.

RIAs ermöglichen daher eine Aktualisierung der Benutzerschnittstelle, ohne die ganze Seite neu laden zu müssen [Coop07]. Mit RIAs können dynamische Inhalte und selbst entwickelte Komponenten realisiert werden. Diese neuen Möglichkeiten im Web schaffen auch neue Barrieren. Zum Beispiel gibt es Probleme mit der Interoperabilität von RIAs und assistierenden Technologien. HTML alleine ist nicht mächtig genug, um diese Lücke in der Interoperabilität schließen zu können. Aus diesem Grund wurde von der WAI die *WAI-ARIA* [W3ca08] *WAI – Accessible Rich Internet Applications Suite* entwickelt.

“WAI-ARIA, the Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. It especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.” [W3cw08b]

WAI-ARIA hilft Entwicklern von Web-Anwendungen Rollen, Zustände und Eigenschaften von selbst entwickelten Komponenten zu definieren, damit diese Komponenten von assistierenden Technologien genutzt werden können. Dazu werden zusätzliche *semantische* Daten in HTML bzw. XHTML untergebracht. Es werden auch Mittel zur Verfügung gestellt um die Benutzer auf die Aktualisierung von Inhalten hinzuweisen, auch wenn eine Seite nicht komplett neu geladen wurde. WAI-ARIA ist kein Richtliniensatz für die Überprüfung der Barrierefreiheit sondern eine Erweiterung der Markup-Sprachen HTML und XHTML, mit deren Hilfe die Erstellung von barrierefreien RIAs erleichtert wird.

2.3.1 Grundlegende Probleme von HTML bezüglich RIAs

Klassisches HTML erschwert die Zugänglichkeit von dynamischen Inhalten. Zum Beispiel verwenden Autoren von JavaScript Inhalten oft das DIV-Tag, um eine Benutzerschnittstelle zu definieren. HTML DIV-Tags stellen aber keine semantischen Informationen zur Verfügung. Beispielsweise wird mit einem DIV-Tag der Start eines Pop-Up-Menüs definiert, aber es existiert kein Mechanismus in HTML um:

- Die Rolle des DIV-Tags als Pop-Up-Menü zu identifizieren
- Assistierende Technologien darauf hinzuweisen, dass das Menü aktiv ist
- Statusinformationen über das Menü zu erlangen (geöffnet oder geschlossen)
- Informationen über die möglichen Aktionen des Menüs zu erlangen

JavaScript benötigt also eine Zugänglichkeits-Architektur, die eine Zuweisung von semantischen Informationen, von Dokumentelementen zu den assistierenden Technologien auf der Plattform des Benutzeragenten ermöglicht. Bei Desktop-Anwendungen stellen so genannte *Accessibility API's*[1] eine Brücke zwischen klassischen GUI-Komponenten und assistierenden Technologien her.

Dabei werden Datenfelder definiert, die für die Zugänglichkeit relevant sind und von beiden Seiten verwendet werden. Dieser Vertrag ist in Abbildung 2.2 als Box zwischen assistierender Technologie und einem Benutzeragenten dargestellt. In diesem Beispiel fungiert der Benutzeragent als Controller. Was passiert nun wenn JavaScript als Controller eingesetzt wird?

[1] Betriebssysteme stellen Schnittstellen zur Verfügung, um Informationen und Events von Objekten an Assistierende Technologien weiterzuleiten. Die Assistierende Technologie interagiert mit den Objekten über diese Schnittstelle. Beispiele: Java Accessibility API [JAPI], Microsoft Active Accessibility [MSAA], Apple Accessibility for COCOA [AAC]

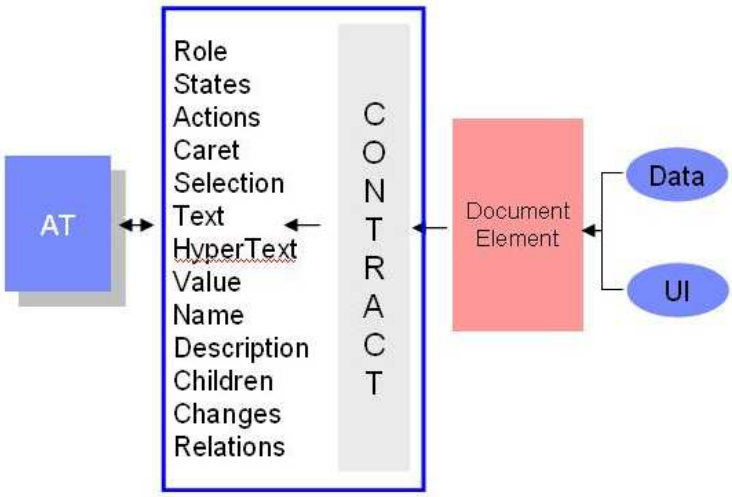

Abbildung 2.2 Interoperabilität ohne JavaScript, [W3cw08c]

JavaScript ändert das standardmäßige Verhalten des Benutzeragenten, da damit neue Komponenten erstellt werden können, die den oben angeführten Vertrag verletzen. Die Informationen für die Zugänglichkeit sind daher nicht mehr vollständig oder korrekt. Dadurch können Lücken und Fehler in der Zugänglichkeit der Anwendung entstehen. Diese potentiellen Fehler und Lücken im Markup sind in Abbildung 2.3 mit einem Stern gekennzeichnet.

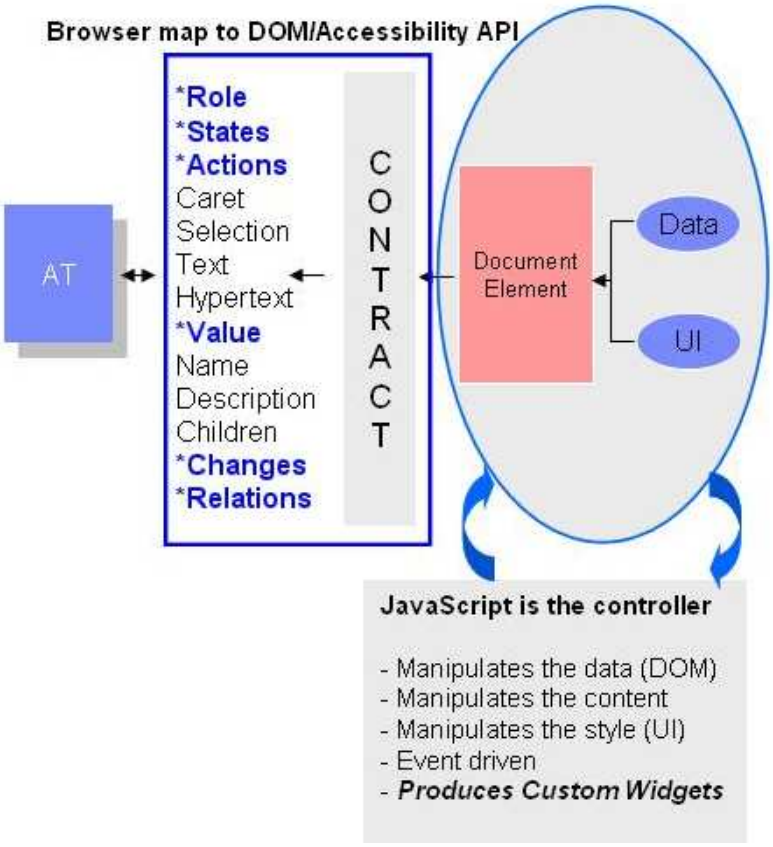

Abbildung 2.3 Interoperabilität mit JavaScript, [W3cw08c]

Die AutorInnen von Web-Anwendungen haben unter diesen Umständen nicht die Möglichkeit, die nötigen Informationen für die Zugänglichkeit im Markup zur Verfügung zu stellen. WAI-ARIA stellt Markup-Erweiterungen für HTML 4.01 und XHTML 1.x zur Verfügung. Bei XHTML können die Erweiterungen in zusätzlichen Namespaces untergebracht werden. Bei HTML 4 gibt es eine Übergangslösung aber künftig sollen die Neuerungen in HTML 5 berücksichtigt werden. [W3cw08c]

2.3.2 Neue Semantik mit WAI-ARIA

- Rollenbeschreibung mit dem „Role-Attribut“

  Für die Beschreibung der Rolle eines Objektes wird das *role-attribute* eingeführt. Damit kann das zuvor beschriebene Problem mit dem JavaScript Menü gelöst werden, da innerhalb des DIV-Blockes ein *role-attribute* eingefügt wird, z.B. (role = „menu“).

  Code-Beispiel:

```
<?xml version=“1.1“ encoding=“us-ascii“?>
<!DOCTYPE html PUBLIC “Accessible Adaptive Applica-
tions//EN” http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd”>

<html xmlns=”http://www.w3.org/1999/xhtml”>
    <body>
        <div role=”menu”>
            File
        </div>
    </body>
</html>
```

- Zustandsbeschreibung von Objekten mit dem „State-Attribut“

  Bei dynamischen Inhalten kann sich der Status eines Objektes ändern, daher werden State-Attribute eingefügt, die den Status eines Objektes als Markup repräsentieren. Zu vorigem Beispiel kommt nun das Attribut: aria-haspopup = “true“ hinzu. Die Autoren von JavaScript müssen dafür sorgen, dass dieses Attribut zur Laufzeit ständig aktualisiert wird, was durch DOM-Calls ermöglicht wird. Bei einer Änderung des Status, sendet der Benutzeragent ein *EVENT_OJECT_STATECHANGE* Event an die Assistierende Technologie.

  Code-Beispiel:

```
<?xml version=“1.1“ encoding=“us-ascii“?>
<!DOCTYPE html PUBLIC “Accessible Adaptive Applica-
tions//EN” http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd”>

<html xmlns=”http://www.w3.org/1999/xhtml”>
```

```
    <body>
        <div role="menu" aria-haspopup="true">
            File
        </div>
    </body>
</html>
```

- Fokus für alle Elemente mit dem neuen „Tabindex-Attribut“

  Screen-Reader und andere Assistierende Technologien müssen wissen welches Element gerade aktiv ist, also den Fokus hat. Jedoch ist es heute mit HTML 4.01 und XHTML 1.x nicht möglich jedem Element einen Fokus zu geben, sondern nur Formular- und Link-Elemente können Events empfangen. Auch ist es mühsam mit der Tabulator Taste über alle Elemente zu klicken bis das gewünschte Element den Fokus hat. Um diese Probleme zu lösen wird die Bedeutung des *tabindex-attribut* verändert. In Tabelle 2.1 wird die neue Bedeutung des Attributes erklärt.

Tabindex-attribute	**Fokusierbar mit Maus oder JavaScript via element.-focus()**	**Tabulator navigierbar**
Nicht vorhanden	Standardverhalten des Elements (Ja für Formulare und Links)	Standardverhalten des Elements
tabindex = “-1“	Ja	Nein, Das Element muss mit *element.focus()* aktiviert werden, abhängig von der Keyboardeingabe
tabindex = “0“	Ja	Ja, enthalten in der Tab-Reihenfolge, je nach relativer Position im Dokument
tabindex = “22“	Ja	Ja, direkte Angabe der Position in der Tab-Reihenfolge

Tabelle 2.1 WAI-ARIA Tabindex-Attribut und Element-Fokus, [W3cw08c]

Das Code-Beispiel wird nun um das *tabindex-attribute* erweitert. Somit ist es möglich, dass auch das JavaScript-Menü über die Tastatur direkt angesteuert werden kann.

Code-Beispiel:

```
<body>
     <div role=”menu” aria-haspopup=”true” tabindex=-1>
          File
     </div>
</body>
```

- Live-Regionen „Live Regions“

Mit Ajax ist es möglich bestimmte Teile einer Seite zu aktualisieren, ohne das die ganze Seite neu geladen wird. Dies kommt z.B. bei einer Chat-Anwendung vor, wenn Nachrichten in einer Text-Box neu erscheinen. Um diese Änderungen assistierenden Technologien mitzuteilen gibt es in WAI-ARIA das Konzept der Live-Regionen. Das Markup für die Live-Regionen umfasst mehrere Attribute, die drei gebräuchlichsten werden hier vorgestellt:

Live: gibt die Priorität an, mit der eine assistierende Technologie die Änderungen in einer Live-Region verfolgen soll. Das Attribut kann mit folgenden Werten belegt werden:

Wert	**Beschreibung**
Live=“off“	This is the default. Any updates made to it should not be announced to the user. Aria-live=”off” would be a sensible setting for things that update very frequently such as timers that change every second.
Live=“polite“	The region is live, but updates made to it should only be announced if the user is not currently doing anything. Aria-live=”polite” should be used in most situations involving live regions that present new information to users, such as updating news headlines.
Live=“assertive“	The region is live. Updates made to it are important enough to be announced to the user as soon as possible, but it is not necessary to immediately interrupt the user. Aria-live=”assertive” should be used if there is information that a user should know about right away, for example, warning messages in a form that does validation on the fly.

Wert	Beschreibung
Live=“rude“	The region is live. Updates to it are extremely important. In fact, the updates are so important that the user must be interrupted immediately. Aria-live=”rude” should be used sparingly and only with great consideration as it can be very annoying to users.

Tabelle 2.2 Mögliche Werte des Live-Attributes, [Thie07]

Atomic: kann die Werte „true“ und „false“ annehmen. Dabei ist „false“ die Standardeinstellung. Die Assistierende Technologie muss dann nicht die gesamte Live-Region darstellen, sondern nur jene Inhalte die sich geändert haben. Dies verhindert die Wiederholte Darstellung von bereits gelesenen Inhalten.

Relevant: gibt an welche Typen von Änderungen für die Live-Region relevant sind. Das Attribut kann einen oder mehrere der folgenden Werte: „additions“, „removals“, „text“ und „all“ annehmen. Bei „additions“ werden nur die neu hinzugekommenen Knoten in der Live-Region beachtet, bei „removals“ nur jene, die gelöscht wurden. Die Standardeinstellung ist „text“, dabei werden die Änderungen von Text, bei bestehenden Knoten beachtet. [Thie07]

Die genauen Definitionen der Semantik von WAI-ARIA sind in [W3ca08] ersichtlich, zusätzlich gibt es von der WAI noch weitere Dokumente. Für Entwickler von RIAs ist das Dokument *WAI-ARIA Best Practices* [W3cw08d] zu empfehlen.

WAI-ARIA erleichtert es *Rich Internet Applications* barrierefrei zu gestalten [Coop07], für eine effiziente Entwicklung dieser Anwendungen ist jedoch die Integration von WAI-ARIA in Web-Frameworks, Entwicklungsumgebungen und Prüfprogrammen nötig.

2.4 Web-Inhalte und deren Einfluss auf die Barrierefreiheit

Moderne grafische Browser müssen eine Vielzahl von Formaten unterstützen. Früher (Anfang der 90er Jahre) gab es nur HTML, doch mittlerweile drängen immer mehr neue Web-Formate und -Technologien auf den Markt. Für Entwickler von Web-Inhalten und -Anwendungen ist es schon schwierig, die Standard Web-Formate wie HTML, XHTML und CSS richtig einzusetzen, um einen barrierefreien Zugang zu ermöglichen. Noch schwieriger wird es wenn proprietäre Formate verwendet werden

müssen, bzw. wenn komplexe Anwendungen (Stichwort: Web 2.0 bzw. *Rich Internet Applications*) erstellt werden sollen. Im Folgenden werden einige Web-Inhalte mit mögliche Barrieren und deren Vermeidung beschrieben.

2.4.1 HTML-, XHTML- und CSS-Inhalte

Bilder und Schriftgrafiken

Bilder und Grafiken können von Screenreadern bzw. Web-Readern nicht interpretiert werden [Hell06, 10]. Ein Screenreader gibt bei jedem Bild ohne nähere Beschreibung das Wort „Grafik“ aus. Es können nach [Kann06, 79] drei Arten von Bildern und Grafiken unterschieden werden:

- *Informationsgrafiken:* Symbole (Icons) oder grafischer Text, z.B. für ein Druckersymbol oder für ein Navigationselement
- *Darstellungsbilder:* Fotos, Zeichnungen, etc..
- *Layoutgrafiken:* Sind Bilder die als Stilmittel dienen

Bilder können mit alternativen Texten „alt-Texte“ barrierefrei gemacht werden. Z.B. in XHTML mit dem Befehl: . Der alt-Text sollte aussagekräftig sein. Informationsgrafiken und Darstellungsbilder können mit alt-Texten beschrieben werden. Für Layoutgrafiken reicht ein leerer alt-Text (alt=““) aus. Eine zusätzliche Möglichkeit bietet das „title-Attribut“ der Text des Attributes erscheint als *Tooltip* wenn der Mauszeiger über das Bild fährt. Besser ist aber das „longdesc-Attribut“, weil damit eine Beschreibung des Bildes als Textdatei eingebunden werden kann …longdesc=“beschreibung.txt“/>. Da der Internet-Explorer das longdesc-Attribut nicht anzeigt, kann ein „D-Link“ *description Link* verwendet werden. Der D-Link wird neben der Grafik platziert und verweist auf eine Datei (.txt oder .xhtml). Ein Beispiel dafür:

Dlink

Schriftgrafiken sind Texte die als Bild dargestellt werden, z.B. ein Firmenlogo oder ein eingescannter Text. Schriftgrafiken sind zu vermeiden, längere Texte sollen mit einer Markup Sprache wie z.B. XHTML dargestellt werden. [Kann06,79]

Eine Sonderstellung bei Schriftgrafiken nimmt CAPTCHA *Completely Automated Public Turing Test to tell Computers and Humans Apart* ein [Holm07]. Dieser Test

wird von diversen Web-Seiten für Sicherheitsabfragen verwendet. Stellt aber eine hohe Barriere für Menschen mit Sehschwächen dar.

Beispiel für ein CAPTCHA-Formular:

Enter the letters shown in the box above:

Bei diesem Test wird oft die Form der Schriftgrafik verwendet, es gib aber auch schon barrierefreie Formen die eine audiovisuelle Wiedergabe erlauben. Auch ein Logiktest in Form einer mathematischen Aufgabe, ist als Alternative in Textform denkbar.

Image-Maps

Image maps sind Grafiken, die zu Navigation verwendet werden [Kann06, 86]. Die Grafik ist in mehrere Bereiche aufgeteilt, beim Anklicken wird jeweils eine neue Seite aufgerufen. Zum Beispiel werden oft Landkarten als *image maps* verwendet, um Niederlassungen von Firmen anzuwählen, siehe Abbildung 2.4. Es gibt zwei Arten von *image-maps*:

Serverseitige image maps: (ismap-Befehl) Diese können nur per Mausklick verwendet werden und sind nicht barrierefrei.

Clientseitige image maps: (usemap-Befehl) Diese können auch mit der Tastatur bedient werden. Die Links sollten zusätzlich auch als Text-Link vorhanden sein, da sonst bei ausgeschalteten Grafiken kein Zugriff möglich ist.

Abbildung 2.4 Beispiel für eine image map[1]

[1] http://www.volksbank.at/m101/volksbank/de/modul/vbsuche/vbsuche.jsp (Stand: 10.04.2009)

Links und Navigation

Eine barrierefreie Navigation zeichnet sich dadurch aus, dass die Navigationselemente überhaupt bedient werden können [Hell06, 30]. Die Navigationselemente sollen geräteunabhängig sein. Die Schrift der Navigationselemente soll groß genug sein, damit sie von Menschen mit Sehschwächen gelesen werden kann. Unter Navigationselementen versteht man z.B. Haupt- und Neben-Navigationsleiste, Fußleiste und *site map* (Inhaltsverzeichnis). Die Anordnung dieser Elemente soll über die ganze Web-Seite konsistent sein. Die aktuelle Position kann mittels Navigationspfadangaben, auch *breadcrumbs* genannt, dargestellt werden. Auch der Einbau einer Suchfunktion kann die Navigation erleichtern. Generell soll die Navigation mit Markup Sprachen realisiert werden, andere Techniken wie Javascript, Flash, etc. erzeugen Barrieren. Navigationselemente sollen im linken und oberen Bereich der Seite angebracht werden. Da andernfalls die Navigationselemente bei Verwendung einer kleineren Auflösung oder von Vergrößerungslupen im nicht sichtbaren Bereich liegen würden.

Links müssen vom übrigen Text unterschieden werden können [Kann06, 88]. Z.B. mit anderer Farbe und unterstrichen. Link-Texte sollen eindeutig sein und sprechende Bezeichnungen haben. Für Benutzer von Screenreadern ist es dann einfacher möglich, die gewünschten Ziele zu finden. Für die Kennzeichnung von aktuellen Menüpunkten muss der Menüpunkt anders formatiert werden (Farbe, Symbol oder Leuchtbalken). Wenn Links direkt aufeinander folgen, muss ein Abstand eingefügt werden. Screenreader interpretieren die beiden Links sonst als einzelnen Link. Die Formatierung der Links soll zentral über CSS definiert werden. Egal ob ein Link mit der Maus oder mit der Tastatur (Tab-Taste) angesteuert wird, die Darstellung soll konsistent sein. In CSS gibt es folgende Situationen für Links:

- *a:link* noch nicht besuchter Link
- *a:visited* bereits besuchter Link
- *a:hover* “mouseover” Nur bei Anwahl mit Mauszeiger
- *a:activ* gerade ausgewählt, beim Klick
- *a:focus* Link hat Fokus, (für Tastaturnutzung)

a:hover ist nur für die Maus relevant und a:focus ist für die Tastaturnutzer. Für beide Situationen, soll im Stylesheet das gleiche Format zugeordnet werden.

Text und Sprache

Die verwendete Schrift soll eine serifenlose Schriftart sein [Kann06, 99]. Die Schrift muss skalierbar sein, dies wird mit der Verwendung der relativen Einheit „em“ erreicht. In CSS z.B.: *p{font-size: 9.9em;}*. Bei Nutzung von Screenreadern ist es von Vorteil, wenn die Überschriften mit dem Überschriften Markup: h1-h6 definiert worden sind. Sie können dann leichter herausgefiltert werden.

Akronyme und Abkürzungen sollen kenntlich gemacht werden [Radt06, 185]. In XHTML gibt es dafür das *acronym*- und *abbr-Attribut*. Ein Beispiel dazu:

Wir verwenden das <acronym title=“Secure Copy Protocol“> SCP
</acronym>

<abbr title=“zum Beispiel“> z.B.</abbr>

Im Browser werden dann die Werte der „title-Attribute“ als Tooltip ausgegeben. Screenreader können auch so konfiguriert werden, dass sie nur das „title-Attribut“ verwenden.

Die vorherrschende Sprache muss ausgezeichnet werden. Screenreader beziehen sich darauf. Die Standardsprache wird im Kopf des Dokuments definiert. Zum Beispiel bei XHTML:

<html lang=“de“ xmlns=“http://www.w3.org/1999/xhtml“ xml:lang=“de“>

Um Screenreadern den Wechsel der Sprache mitzuteilen müssen entsprechende Textstellen gekennzeichnet werden, dies funktioniert mit dem „span-Element“ und den Attributen *lang* bzw. *xml:lang*. Zum Beispiel:

Newsletter

Strukturelemente

Alternative Benutzeragenten sind auf die korrekte Verwendung der Strukturelemente angewiesen [Hell06, 50]. Bei der Verwendung dieser Elemente ist auf folgendes zu achten:

- *Überschriften:* Die spezielle Formatierung von Text als Überschrift ist nicht ausreichend. Für Überschriften sollen die Elemente H1-H6 verwendet werden, damit sie auch für alternative Benutzeragenten als Überschriften erkannt werden. Eine hierarchische Gliederung der Überschriften ist vorteilhaft.

- *Listen:* Für Aufzählungen und Listen stehen die Elemente UL *unordered list*, OL *ordered list*, DL *definition list* zur Verfügung. Für Layoutzwecke sollen die Listen nicht verwendet werden, hierfür gibt es CSS. Listen vereinfachen auch das erstellen von Navigationsleisten[1], diese können dann mit CSS auch beliebig formatiert werden.

- *Absätze:* Grundsätzlich sollen Absätze mit dem <p> Element gekennzeichnet werden. Die Verwendung von
 Elementen zur Erstellung von Absätzen soll vermieden werden. Das <p> Element kann von Screenreadern erkannt werden und somit ist ein springen von Absatz zu Absatz möglich.

Tabellen

Tabellen sollen zur strukturierten Darstellung von Daten verwendet werden und nicht für Layoutzwecke. Screenreader lesen den Inhalt von Tabellen zeilenweise aus. Wenn eine Web-Seite Daten- und Layout-Tabellen enthält, ist es schwierig die Datentabellen zu finden. Daher sind Layouttabellen nicht barrierefrei. Damit Tabellen richtig skalieren [Kann06, 101], sollen die Größenangaben mit relativen Einheiten erfolgen, z.B. *width=25%*. Tabellen sollen eine Überschrift „caption-Element“ und eine Zusammenfassung „summary-Attribut“ enthalten. Spalten- und Zeilenüberschriften sollen auch vorhanden sein.

Problematisch sind *komplexe Tabellen* mit mehreren logischen Ebenen von Zeilen- und Spalten-Überschriften [Hell06, 50], [Radt06, 170]. Hierfür gibt es die Attribute *header* und *scope*. Das „headers-Attribut“ ermöglicht die Zuordnung einer Tabellenzelle zu einer Zeilen- und Spalten-Überschrift. Damit kann der Screenreader in jeder Zelle die Zugehörigkeit zu Zeile und Spalte ermitteln. Mit dem „scope-Attribut“ kann dasselbe Ergebnis erzielt werden. Jedoch ist die Syntax etwas anders gestaltet, als beim „header-Attribut“. Das „scope-Attribut“ muss einen der Werte *col* oder *row* annehmen. *Scope* ermöglicht eine elegantere Kodierung als das „header-Attribut“ und wird von Screenreadern besser unterstützt. Ein Beispiel dafür:

```
<table align=”center” border=”1” cellspacing=”0” cellpad-
ding=”5”>

<tr>
<td rowspan=”2”> </td>
<th colspan=”2” scope=”colgroup”>Winter</th>
<th colspan=”2” scope=”colgroup”>Summer</th>
</tr>
```

[1] http://css.maxdesign.com.au/ (Stand: 15.4.2009)

```
<tr>
<th scope=″col″>Morning</th>
<th scope=″col″>Afternoon</th>
<th scope=″col″>Morning</th>
<th scope=″col″>Afternoon</th>
</tr>

<tr>
<td scope=″row″>Wilma</td>
<td>9-11</td>
<td>12-6</td>
<td>7-11</td>
<td>12-3</td>
</tr>
…
</table>
```

Dieser Code erzeugt folgende HTML-Tabelle. Allerdings wurde die zweite Tabellenzeile im Code-Beispiel weggelassen.

	Winter		**Summer**	
	Morning	**Afternoon**	**Morning**	**Afternoon**
Wilma	9-11	12-6	7-11	12-3
Fred	10-11	12-6	9-11	12-5

Tabelle 2.3 Beispiel für eine komplexe HTML-Tabelle[1]

Frames

Frames sind teilweise problematisch für die Barrierefreiheit [Kann06, 103]. Ältere Browser und Screenreader unterstützen Frames nicht. Jedoch haben moderne Screenreader keine Probleme mit Frames. In Text-Browsern werden Frames als Links angezeigt. Zudem ist der Verweis von Suchmaschinen auf einzelne Seiten eines „Frame-Sets“ problematisch, da der Rest des „Frame-Sets“ mangels Navigation, möglicherweise verborgen bleibt. Wichtig ist, dass „Frame-Sets“ beschrieben werden und einzelne Frames über das „title-Attribut“ benannt werden. Falls der Browser keine Frames unterstützt, gibt es noch die Möglichkeit mittels des „Noframes-Elementes“ einen alternativen Inhalt zu präsentieren. [Hell06, 12]

[1] http://www.yourhtmlsource.com/examples/accessibletable.html, (Stand: 20.4.2009)

Formulare

Formulare sind ein wichtiges Interaktionsmittel für Web-Anwendungen [Hell06, 45]. Bei unsachgemäßer Implementierung treten eine Reihe von Barrieren auf. Formularfelder müssen einwandfrei erkannt werden können und geräteunabhängig bedient werden können. Die Bedienung mit der Tastatur ist oft umständlich. Zwischen den Formularfeldern wird dabei mit der Tab-Taste „navigiert“. Die Formularfelder sollen in einer logischen Reihenfolge angeordnet sein. Die Reihenfolge für die Formularfelder kann alternativ auch mit dem „tabindex-Attribut“ spezifiziert werden. Für Screenreader ist es entscheidend, dass die Eingabefelder im Quellcode direkt mit einem vorangestellten Label benannt werden. Z.B.:

```
<label> Vorname: </label> <input type=“text“ name=“textfeld1“>
```

So ist die Bedeutung des Eingabefeldes klar. Besser ist jedoch die explizite Zuordnung mit dem „for-Attribut“ z.B.:

```
<label for=“vorname“> Vorname: </label>
<input type=“text“ id=“vorname“/>
```

Das Label-Element kann angeklickt werden, wenn mit dem Mauszeiger auf den Label-Text geklickt wird, dann springt der Cursor in das Eingabefeld.

Wenn aufwändige Formulare erstellt werden, ist es sinnvoll logische Gruppen von Eingabefeldern und Beschriftung zu erstellen [Radt06, 174]. Es ist dann für die Nutzer von Screenreadern einfacher sich zu orientieren. Dies kann mit den FIELDSET- und LEGEND-Elementen bewerkstelligt werden. Zum Beispiel kann dies so aussehen:

```
<fieldset>
     <legend><strong>1: Personal Information</strong></legend>
     Name <input type=”text” name=”name”>
     Email <input type=”text” name=”email”>
</fieldset><br>
<fieldset>
     <legend><strong>2: Preferences</strong></legend>
     Favourite Colour: <input type=”radio” name=”test”
     value=”blue”> Blue <input type=”radio” name=”test”
     value=”scarlet”> Scarlet <input type=”radio” name=”test”
     value=”puce”> Puce
</fieldset><br>
<fieldset>
     <legend><strong>3: Submit Info</strong></legend>
     <div align=”center”>
          <input type=”submit” value=”Submit” onclick=”return
          false;”>   <input type=”reset”>
     </div>
</fieldset>
```

Abbildung 2.5 Beispiel für eine Gruppierung mit dem Fieldset-Element[1]

Neben den Aspekten der Barrierefreiheit gibt es bei den Formularen noch zahlreiche Aspekte der Usability, die hier aber nicht erläutert werden.

2.4.2 Andere Web-Formate und Inhalte

Neben den von der W3C empfohlenen Web-Formaten gib es noch eine Reihe von proprietären Formaten, die eine Stellung als Quasi-Standard inne haben. Diese Formate und andere Typen von Inhalten, die nicht konform mit Standard Markup-Sprachen sind, werden hier vorgestellt.

Multimedia-Inhalte

Der Einsatz von Multimediaformaten (Video, Animation, Sound, Präsentationen,..) im Web nimmt zu. Grundsätzlich treten Probleme mit Multimedia-Objekten auf, wenn sie nur mit einer Bild- bzw. Ton-Spur wiedergegeben werden. Ähnlich wie bei den Bildern sind Multimediainhalte mit alternativen Texten zu benennen [Kann06, 108]. Probleme gibt es mit selbst startenden Multimediainhalten, der Zeitpunkt der Wiedergabe soll vom Benutzer bestimmt werden können. Für Gehörlose kann es hilfreich sein wenn Videos einen Untertitel enthalten bzw. mit Gebärdensprache begleitet werden. Für blinde Menschen, die mit einem Screenreader arbeiten, kann eine Audiodeskription hilfreich sein. Eine Audiodeskription ist eine Audiodatei, die gesprochene Szenenerklärungen enthält. Die Organisation von Untertiteln und Audiodeskriptionen wird mit der Sprache SMIL (Synchronized Multimedia Integration Language) erreicht. Dies ist ein „XML-Dokumenttyp“ der W3C. In einem SMIL[2] Dokument wird definiert, zu welchem Zeitpunkt, welche Medien bzw. Dateien aufgerufen werden.

[1] http://www.yourhtmlsource.com/forms/formsaccessibility.html (Stand: 10.04.2009)

[2] http://www.w3.org/TR/2008/REC-SMIL3-20081201/ (Stand: 20.5.2009)

Untertitel werden z.B. mit Textdateien realisiert, die sekundengenaue Zeitangaben enthalten. [Hell06, 54]

Flash

Flash ist ein proprietäres Format bzw. ein Werkzeug zur Erstellung von Web-Anwendungen der Firma Adobe (seit 2005, vorher Macromedia). Es wird häufig für eingebettete Videos verwendet bzw. für Flash-Spiele oder –Präsentationen. Flash benötigt ein Browser-Plugin. Flash ist generell nicht als barrierefrei einzustufen [Kann06, 109]. Es ergeben sich u.a. folgende Nachteile [Radt06, 223] in Bezug auf die Zugänglichkeit:

- Flash kann von Textbrowsern nicht gelesen werden.
- Erhält ein Flash-Bereich den Fokus (nur mit Maus möglich), kommt man mit der Tab-Taste nicht mehr heraus.
- Nicht alle Screenreader können Flash richtig interpretieren
- Schriftgrößen und Farbschemata können mit den üblichen Methoden nicht verändert werden

„Das Hauptproblem bei Flash ist, dass es für viele Content- und kommerzielle Sites nicht angemessen ist. Doch Programmierer setzen es in diesen unangemessenen Situationen ein, weil mit Flash todschicke Präsentationen möglich sind, bei denen die Kunden den Eindruck bekommen, sie kriegen mehr fürs Geld..“ [Zeld06, 103]

PDF

PDF (Portable Document Format) ist ein proprietäres Format der Firma Adobe. Allerdings sind die Spezifikationen offen. Für Menschen mit besonderen Bedürfnissen, sind PDF-Dokumente im Allgemeinen schwer zugänglich. Zu dem benötigt PDF ein Browser-Plugin (PDF-Reader). Bezüglich Barrierefreiheit besteht das Problem, dass Screenreader und PDF-Reader nicht gut zusammen arbeiten. Prinzipiell gibt es die Möglichkeiten mit den Werkzeugen von Adobe (z.B. Acrobat Pro), barrierefreie PDF-Dokumente zu erstellen. In der Praxis werden aber viele PDF-Dokumente über den Druckertreiber generiert, damit scheitert dieses Konzept. Für Menschen mit besonderen Bedürfnissen wäre das RTF-Format oft geeigneter als PDF, da es aus reinem Text und Markup aufgebaut ist [Hell06, 57], [Radt06, 203].

JavaScript

JavaScript ist eine clientseitige Skript-Sprache [Wenz08]. Javascript wird oft eingesetzt um dynamische und optische Effekte zu erzielen. Zum Beispiel für „Event-Handler“ und Links. Mit dem aufkommen von Ajax wurde JavaScript wieder sehr populär. Grundsätzlich sollen Web-Seiten nach WCAG1.0 auch bei Abschaltung von Javascript noch zugänglich sein [Hell06, 26]. Das heißt für jede Javascript-Funktion, muss ein Ersatz in Form von HTML oder CSS vorhanden sein bzw. sind serverseitige Alternativen zu verwenden. Dies wird aber in den WCAG2.0 relativiert, da mit WAI-ARIA ein Framework zur Verfügung steht, dass auch eine barrierefreie Verwendung von JavaScript möglich macht. Für Browser die kein JavaScript unterstützen, kann das „Noscript-Element“ verwendet werden um eine Fehlermeldung auszugeben.

2.4.3 Assistierende Technologien

Assistierende Technologien [Mies02] sind technische Hilfsmittel also Geräte bzw. Software, die sich wie ein Benutzeragent verhalten bzw. mit einem üblichen Benutzeragenten interagieren. Für Menschen mit besonderen Bedürfnissen ist es dann möglich, die für sie fehlenden Interface-Funktionen der üblichen Benutzeragenten, mit Hilfe der assistierenden Technologie zu kompensieren. Assistierende Technologien kommunizieren meist über „Accessibility APIs“ mit den Benutzeragenten. Beispiele für Assistierende Technologien sind:

- *Bildschirmlupen:* Diese Programme vergrößern Teile der Bildschirmausgabe. Betriebssysteme bieten diese Funktion meist optional an, es gibt aber auch eigenständige Programme, die leistungsfähiger sind. Unter Windows ist die Funktion unter „Zubehör/Eingabehilfen/Bildschirmlupe“ zu finden [Radt06].

- *Bildschirmtastaturen:* Eine einfache Bildschirmtastatur ist z.B. in Windows unter „Zubehör/Eingabehilfen/Bildschirmtastatur“ verfügbar. Bildschirmtastaturen können auch mit einer einzigen Taste (Leertaste) bedient werden. Dabei läuft ein farbiger Balken von oben nach unten über die Tastatur, beim Drücken der Taste wird die Zeile ausgewählt. Dann läuft ein farbiger Punkt von links nach rechts über die Tasten, wenn die gewünschte Taste erreicht ist, wird wiederum die Leertaste gedrückt [Radt06].

Abbildung 2.6 Windows XP, Bildschirmtastatur

- *Braillezeilen:* Sind Geräte, die den Bildschirminhalt in Form von Blindenpunktschrift „Braille“ ausgeben. Dabei kann jedes Zeichen des Alphabets mit 8 höhenveränderbaren Stiften dargestellt werden. Gelesen wird mit den Fingerkuppen, welche die Position der Stifte ertasten. 8 Stifte ergeben ein „Braillemodul“, eine Braillezeile besteht aus mehreren Braillemodulen. Für die Nutzung der Braillezeile ist ein Screenreader notwendig [Radt06].

Abbildung 2.7 Beispiel für eine Braillezeile[1]

- *Screenreader:* Sind Programme die zur nichtvisuellen Ausgabe von Bildschirminhalten verwendet werden. Die Ausgabe erfolgt über Sprachausgabe und/oder über Braillezeilen. Es wird unterschieden zwischen *Screenreader* und *Web-Reader* bzw. *Voice-Browser*. Die Screenreader lesen den Inhalt des Bildschirms unabhängig vom laufenden Programm aus. Die anderen Programme sind spezielle Web-Browser. Beispiele[2] für kommerzielle Screenreader sind: JAWS, Virgo, Blindows, HAL, COBRA,.. Beispiele für Freeware Screenreader sind: NVDA, Orca (nur für Linux), Thunder,.. [Radt06].

[1] http://www.incobs.de/produktinfos/braillezeilen/index.php, (Stand: 10.4.2009)

[2] http://www.incobs.de/produktinfos/screenreader/index.php, (Stand: 10.4.2009)

- *Alternative Zeigergeräte und Tastaturen:* Es gibt eine Vielzahl von alternativen Eingabegeräten. Großfeldtastaturen sind für Menschen mit Koordinationsproblemen hilfreich, Kleinfeldtastaturen sind für Menschen mit eingeschränktem Aktionsradius gedacht. Es gibt Geräte für die Steuerung des Mauszeigers mit dem Kopf oder über die Pupillenbewegung [Kann06].

- *Spracherkennung:* Spezielle Software zur Spracherkennung ermöglicht die Navigation in Dokumenten und die Ausführung von Kommandos.

3 Überprüfung der Barrierefreiheit

Im vorigen Kapitel wurden mögliche Barrieren von typischen Web-Inhalten aufgezeigt. Sowie die Bemühungen von Regierungen und Standardisierungsgremien, diesen Barrieren mit Gesetzen und Richtlinien entgegenzuwirken. Es stellt sich nun die Frage, wie die Barrierefreiheit einer Web-Seite überprüft werden kann, bzw. welche Methoden es dazu gibt. Mittlerweile gibt es eine Reihe von Software-Werkzeugen, die EntwicklerInnen dabei unterstützen, die Barrierefreiheit von Web-Seiten und Web-Anwendungen zu überprüfen. Diese Software-Werkzeuge werden in der Literatur auch als w*eb accessibility evaluation tools* bezeichnet, im Weiteren werden sie hier als Prüfprogramme bezeichnet. Natürlich sind diese Prüfprogramme nicht perfekt und ein vollautomatisches Prüfverfahren das alle Zugänglichkeitsbarrieren aufdeckt, gibt es noch nicht. Im Folgenden werden einige Prüfprogramme aufgezählt. Es werden auch Vorgehensweisen vorgestellt, die zur Evaluierung der Barrierefreiheit von Web-Inhalten verwendet werden können.

3.1 Prüfprogramme

Bei den Prüfprogrammen gibt es verschiedene Typen. Zum einen gibt es Onlinedienste die mittels Eingabe einer URL, die entsprechende Webseite überprüfen und das Ergebnis gleich im Browser anzeigen. Manche dieser Dienste erlauben auch einen Datei-Upload. Dann gibt es Desktop-Anwendungen die meist einen umfangreicheren Funktionsumfang haben, als die Online-Dienste. Der Vorteil der Desktop-Anwendungen ist, dass sie auch auf einen lokal laufenden Webserver einfach zugreifen können. Dies ist interessant für EntwicklerInnen. Dann gibt es noch Prüfprogramme die als Browser-Plugin funktionieren. Diese Programme sind sehr zu empfehlen, da sie direkt auf das DOM (Document Object Model) des Browsers zugreifen [Gund06] und nicht nur die HTML-Datei einlesen. Was die Unterstützung der verschiedenen Richtlinien betrifft, sind die Programme natürlich unterschiedlich ausgeprägt. WCAG1.0 und *Section 508* werden aber von sehr vielen Prüfprogrammen unterstützt. Es gibt Prüfprogramme die sich auf die WCAG-Konformitätsprüfung von Web-Seiten spezialisiert haben. Andere haben sich auf ein bestimmtes Teilgebiet der Barrierefreiheit spezialisiert. Zum Beispiel Analyse von Farbkontrasten oder Validierung von Grammatik. Die Prüfprogramme unterscheiden sich auch in der Art, wie sie

die Ergebnisse aufbereiten. Es gibt Programme, welche die Richtlinienverletzungen direkt auf der getesteten Webseite anzeigen (z.B. Wave). Dabei wird an der betreffenden Stelle, ein Symbol oder eine Fehlermeldung eingeblendet. Dann gibt es Programme, die HTML-Berichte erstellen. In diesen Berichten werden die Verletzungen der Richtlinien und die entsprechende Zeile im Quellcode dokumentiert. Das richtige Werkzeug zu finden ist eine Aufgabe für sich. Oft ist es nötig mehrere Werkzeuge zu verwenden. Eine gute Auswahlmöglichkeit von Prüfprogrammen aller Art bietet die WAI auf ihrer Homepage [W3cw06]. Einige freie und kommerzielle Prüfprogramme werden nun aufgelistet.

3.1.1 Freie Prüfprogramme

Name	Typ
TAW[1]	WCAG1.0-Prüfprogramm (Desktop-Anwendung)
Web Accessibility Toolbar[2]	Browser-Toolbar für IE (diverse Test's)
Firefox Accessibility Extension[3]	Browser-Toolbar für Firefox (diverse Test's)
Web Developer[4]	Browser-Toolbar für Firefox (diverse Test's)
FAE[5]	Online-Service (FAE-Regeln)
A-Checker[6]	Online-Service (WCAG, Section508,..)
Wave[7]	Online-Service und Browser-Toolbar (WCAG, Section 508)
aDesigner[8]	Barrieren-Simulator

1 http://www.tawdis.net/taw3/cms/en, (Stand: 20.05.2009)

2 http://www.visionaustralia.org.au/ais/toolbar/, (Stand: 20.05.2009)

3 http://firefox.cita.uiuc.edu/index.php, (Stand: 20.05.2009)

4 http://chrispederick.com/work/web-developer/, (Stand: 20.05.2009)

5 http://fae.cita.uiuc.edu/, (Stand: 20.05.2009)

6 http://checker.atrc.utoronto.ca/index.html, (Stand: 20.05.2009)

7 http://wave.webaim.org/, (Stand: 20.05.2009)

8 http://www.eclipse.org/actf/downloads/tools/aDesigner/index.php, (Stand: 20.05.2009)

Fortsetzung, Freie Prüfprogramme:

Name	Typ
Web Accessibility Self-Evaluation Tool[1]	Online Evaluierungsanleitung
Flicker Rate Test for Gif Images[2]	Online-Service (Flicker-Test für Gif's)
Juicy Studio Accessibility Toolbar[3]	Browser-Toolbar für Firefox (Farbkontrast-analyse und WAI-ARIA)
W3C Markup Validation Service[4]	Online-Service (HTML-Validierung)
W3C CSS Validation Service[5]	Online-Service (CSS-Validierung)
Fangs[6]	Screenreader-Emulator, Browser-Plugin (Firefox)
WebIE[7]	Textbrowser für Screenreader

Tabelle 3.1 Freie Prüfprogramme

3.1.2 Kommerzielle Prüfprogramme

Name	Typ
AccVerify[8]	komplette Prüfumgebung
(Bobby) IBM Rational Policy Tester[9]	komplette Prüfumgebung

1 http://www.techdis.ac.uk/index.php?p=3_6_20051905120529, (Stand: 20.05.2009)

2 http://tools.webaccessibile.org/test/check.aspx, (Stand: 20.05.2009)

3 https://addons.mozilla.org/de/firefox/addon/9108, (Stand: 20.05.2009)

4 http://validator.w3.org/, (Stand: 20.05.2009)

5 http://jigsaw.w3.org/css-validator/, (Stand: 20.05.2009)

6 http://www.standards-schmandards.com/projects/fangs/, (Stand: 20.05.2009)

7 http://www.webbie.org.uk/, (Stand: 20.05.2009)

8 http://www.hisoftware.com/products/accverify.html, (Stand: 20.05.2009)

9 http://www-01.ibm.com/software/awdtools/tester/policy/accessibility/, (Stand: 20.05.2009)

Fortsetzung, Kommerzielle Prüfprogramme:

Name	Typ
Imergo[1]	komplette Prüfumgebung
Infocus[2]	komplette Prüfumgebung
AccessValet[3]	Online-Service

Tabelle 3.2 Kommerzielle Prüfprogramme

In [Kirc02] wird eine Evaluierung von verschiedenen Prüfprogrammen vorgenommen. Es hat sich gezeigt, dass einige Prüfpunkte der WAI- und *Section 508*-Richtlinien nicht automatisch geprüft werden konnten. Sondern noch eine manuelle Prüfung dieser Punkte notwendig war. Eine weitere Erkenntnis ist, dass EntwicklerInnen für die richtige Erstellung von barrierefreien Inhalten geschult werden müssen. Ist das Entwicklungspersonal einmal geschult, entstehen später kaum noch Zusatzkosten. Da die produzierten Web-Seiten, schon beim ersten Entwurf viele Aspekte der Barrierefreiheit berücksichtigen. Unzugängliche Web-Seiten zu reparieren erzeugt hingegen viel höhere Kosten, als die Schulung des Entwicklungspersonals vor dem Projektbeginn. Das Ergebnis der Evaluierung nach [Kirc02] ist in Tabelle 3.3 ersichtlich.

Name and reference	**on-line version**	**download version**	**classification**	**scripts**	**declared coverage**	**notes**
AccessEnable [20]	whole website (e-mail report)	for purchase	Evaluation (wrong tag correction)	Wrongly "corrected"	Section 508	careful supervision needed
AccVerify [21]	Web pages	for purchase	evaluation / repair	ignored	WAI guidelines, Section 508	didactically valid
CAST – Bobby [22]	One page at a time	For purchase	Evaluation	Ignored	WAI guidelines	Complete tool (hints for needed manual checks)

[1] http://imergo.com/home, (Stand: 20.05.2009)

[2] https://www.ssbbartgroup.com/amp/infocus.php, (Stand: 20.05.2009)

[3] http://valet.webthing.com/access/, (Stand: 20.05.2009)

Fortsetzung der Evaluierung von Prüfprogrammen nach [Kirc02]:

Name and reference	**on-line version**	**download version**	**classification**	**scripts**	**declared coverage**	**notes**
DoctorHTML [24]	One page at a time (max 5)	For purchase for Linux, sun Solaris, windows 2K	Evaluation	ignored	HTML 4.0, IE, Netscape	Accurate, complete report; didactically valid
InSight – AskAlice [25]	Whole website	Not available	Evaluation	unknown	Section 508	Not really useful
Tidy [27]	Not available	Free (open source)	repair	HTML tags in it corrected	Any of W3C standards	Strictly follows the HTML guidelines
UsableNet – Lift [28]	One page at a time, report of all frames	Only for MM DreamWeaver HTML editor	Evaluation	Ignored	Section 508	didactically valid; good report of accessibility errors
WAVE [29]	You „navigate“ and comments are displayed	Not available	Evaluation	Ignored, but executed in browser	Wai-guidelines	Easy usage; only main errors reported
W3C HTML validation service [30]	One page at a time	Not available	Evaluation	ignored	Any of W3C standards	Strictly follows the HTML guidelines; needs precise HTML version

Tabelle 3.3 Evaluierung von Prüfprogrammen, [Kirc02]

3.1.3 Was kann automatisch geprüft werden?

Eine automatische Prüfung erfolgt ohne Benutzerinteraktion. Bei den WCAG1.0, können z.B. die beschriebenen Aspekte der folgenden Prüfpunkte vollautomatisch geprüft werden:

- *Prüfpunkt 1.1:* Ist ein Alt-Attribut bei Nicht-Text Inhalten vorhanden?
- *Prüfpunkt 2.2:* Wie groß ist der Kontrast zwischen Vordergrund und Hintergrundfarbe?
- *Prüfpunkt 3.2:* Validiert das Dokument gegen veröffentlichte Grammatiken?

- *Prüfpunkt 3.4:* Werden absolute Einheiten verwendet?
- *Prüfpunkt 3.5:* Sind Header-Elemente vorhanden?
- *Prüfpunkt 3.6:* Stimmt die Hierarchie von Listen-Elementen?
- *Prüfpunkt 4.2:* Stimmt die Syntax von *Acronym*- und *Abbreviation*-Elementen?
- *Prüfpunkt 4.3:* Gibt es eine Identifikation der primären Sprache im Header?
- *Prüfpunkt 5.1:* Sind Zeilen- und Spalten-Überschriften bei Datentabellen ausgezeichnet?
- *Prüfpunkt 7.2:* Ist ein verbotenes *Blink*-Element vorhanden?
- *Prüfpunkt 7.4:* Gibt es eine periodischer Aktualisierung des Inhaltes?
- *Prüfpunkt 10.1:* Gibt es Pop-Up-Fenster?
- *Prüfpunkt 10.5:* Gibt es Link-Elemente in der gleichen Zeile, die nicht durch Trennzeichen getrennt sind?
- *Prüfpunkt 12.4:* Gibt es immer eine explizite Zuordnung von Label-Elementen zu den dazugehörigen Formular-Elementen?
- *Prüfpunkt 13.1:* Gibt es Links mit gleichem Link-Text und unterschiedlichen Zielen?
- *Prüfpunkt 13.2:* Sind Meta-Elemente enthalten?

Es kommen jene Aspekte der Prüfpunkte vor, bei denen auf das Vorhandensein, oder nicht Vorhandensein von bestimmten Elementen oder Attributen geachtet wird. Die Ausbeute bei der automatischen Prüfung, ist damit auf derartige Aspekte reduziert. Für die Entwicklung von Prüfalgorithmen ist es eine Voraussetzung, die einzelnen Techniken pro Prüfpunkt zu definieren. Dies kann mit speziellen Methodologien erreicht werden. Ein Beispiel dafür ist die UWEM (Unified Web Evaluation Methodology) nach [Wabc07].

3.2 Evaluierungsmethoden

Um die Barrierefreiheit von Web-Inhalten bzw. –Anwendungen feststellen zu können, ist es nötig sich für einen Evaluierungsprozess zu entscheiden. Die WAI stellt mehrere Ansätze bereit, wie so ein Prozess aussehen kann. Bei der Wahl des Prozesses ist darauf zu achten wie das Umfeld und die Evaluierungsobjekte beschaffen sind. Es kann unterschieden werden zwischen einer Evaluierung während der Entwicklung und einer Evaluierung von bestehenden Web-Seiten. Es ist auch zu klären ob die Konformität, z.B. hinsichtlich WCAG evaluiert werden soll oder ob die Evaluierung sich an selbst gewählten Kriterien orientiert. Für eine Konformitätsprüfung gegen die WCAG, wird von der WAI nach [W3cc05], folgender Prozess vorgeschlagen:

- Definition der Rahmenbedingungen
    - Wahl der Konformitätsstufe
    - Auswahl einer Stichprobe von Web-Seiten mit unterschiedlichen Funktionen, Inhalten bzw. Layout
- Auswahl von Prüfprogrammen
    - Programme für Validierung der Syntax bzw. Grammatik
    - Verwendung von mindestens zwei Accessibility-Prüfprogrammen
- Manuelle Evaluierung der ausgewählten Stichprobe von Web-Seiten
    - Verwendung der „WCAG-Checkliste“ [W3cc00] zur manuellen Prüfung jener Prüfpunkte die nicht automatisch geprüft werden können
    - Verwendung von graphischen Browsern und „Accessibility-Toolbars“
    - Verwendung eines Text-Browsers
    - Verwendung eines Screenreaders bzw. „Voice-Browser“
    - Lesen der Text-Inhalte und Prüfung auf korrekten Sprachstil
- Zusammenfassung der Ergebnisse

Zu beachten sind spezielle Kontexte, wie z.B. dynamisch generierte Inhalte. Bei diesen Inhalten werden z.B. Daten aus einer Datenbank in einer HTML-Tabelle geren-

dert. Damit solche Inhalte überhaupt erscheinen, ist oft eine Benutzerinteraktion nötig (z.B. drücken eines Buttons). Vollautomatische Prüfprogramme können diese Inhalte also nicht erfassen bzw. nur unter Verwendung von aufwändigen Makrofunktionen. Eine Lösung ist, dass diese generierten Seiten lokal gespeichert werden, damit sie anschließend überprüft werden können. Dies hat auch den Vorteil, dass jedes Prüfprogramm, die gleichen Zeilennummern angibt. Für die Prüfung von dynamischen Inhalten sind jene Prüfprogramme sehr geeignet, die einen Browser emulieren bzw. als Browser-Plugin zur Verfügung stehen. Deren Verwendung erfordert aber eine Benutzerinteraktion und das Prüfverfahren wird zeitaufwendiger. Dafür sind die Ergebnisse genauer.

Ein etwas anderer Evaluierungsprozess, genannt SERPA *Streamlined Evaluation and Reporting Process for Accessibility* wird in [Lawc05] beschrieben. Dieser Ansatz richtet sich nach den Bedürfnissen von Web-EntwicklerInnen. Da diese Gruppe die eigentlichen Empfänger der Evaluierungs-Berichte sind. Das besondere an diesem Prozess ist die Aufbereitung der Ergebnisse. Statt der Bereitstellung eines unlesbaren Berichtes mit unzähligen Fehler-Instanzen, werden die Ergebnisse extra aufbereitet. Wichtige Aspekte dabei sind:

- *500 Instanzen eines Problemtyps = 1 Problem:* Es ist also nicht nötig jede Instanz extra aufzuzählen. Für die EntwicklerInnen ist nur wichtig zu wissen, dass dieser Problemtyp zumindest einmal vorgekommen ist.

- *Numerische Darstellung der Ergebnisse:* Ausgehend davon, dass nur jeweils eine Instanz einer Richtlinienverletzung gezählt wird, ergibt sich folgende Unterscheidung:

    a. Wie viele W3C Prüfpunkte wurden eingehalten? (d.h. für die ProgrammiererInnen ist bei diesen Punkten nichts zu tun)

    b. Wie viele der verletzten Prüfpunkte, benötigen nur geringen Aufwand um behoben zu werden? (d.h. nicht viel zu tun für die ProgrammiererInnen)

    c. Wie viele von den Prüfpunkten sind nicht anwendbar *not applicable*? (d.h. nichts zu tun für die ProgrammiererInnen)

    d. Wie viele Prüfpunkte wurden nicht eingehalten und müssen behoben werden? (d.h. echter Aufwand für die ProgrammiererInnen)

Aus den Punkten a, b und c wird in [Lawc05] eine Formel definiert, die in Prozent den Fortschritt der Barrierefreiheit, für eine Web-Seite angibt. Diese APPF *Accessibility Programming Progress Formula* ist so definiert:

$P = [(a + b) / (T – c)] *100\%$

P… Fortschritt der Barrierefreiheit in %

T… Anzahl der Prüfpunkte in der Checkliste

Es wird also der Quotient aus der Anzahl der „relativ gut eingehaltenen" Prüfpunkte und der Anzahl der relevanten Prüfpunkte gebildet. Zum Beispiel: T= 19, a=10, b=2, c=3 ergibt dann für P=75%.

Der zusätzliche Aufwand liegt allerdings darin, festzustellen welche der nicht eingehaltenen Prüfpunkte zur Gruppe b gehören. Das elegante an dieser Darstellung ist aber, dass sie die Ergebnisse der Evaluierung rein qualitativ beschreibt. Damit können z.B. Seiten verglichen werden, die mit unterschiedlichen Web-Technologien erstellt worden sind aber die gleichen Funktionen abbilden. Dies kommt in Kapitel 6 auch zur Anwendung.

Der vollständige SERPA Prozess nach [Lawc05], umfasst folgende Schritte:

1. Diskussion der Anforderungen und Ziele des Entwicklungsprojektes
2. Konsensfindung über den Grad der Barrierefreiheit
3. Erstellung einer entwicklerorientierten Berichtsvorlage
4. Evaluierung der Barrierefreiheit
5. Berechnung der *Accessibility Programming Progress Formula*
6. Nur notwendige Informationen an die Entwicklung weitergeben
7. Erneute Überprüfung der angepassten Web-Seiten

4 Generierung von Web-Anwendungen und Barrierefreiheit

Die Entwicklung von Web-Anwendungen hat sich verändert, die manuelle Kodierung wurde großteils durch Codegenerierung abgelöst. Die modernen Ansätze orientieren sich aber hauptsächlich an funktionalen Aspekten. Aspekte wie die Barrierefreiheit im Web werden noch wenig berücksichtigt.

4.1 Softwareentwicklungstechniken und ihre Auswirkungen

Moderne Softwareentwicklungstechniken erlauben es, die Entwicklungszeit drastisch zu reduzieren. In den Anfängen der Informatik wurde noch viel von Hand programmiert, inzwischen gibt es moderne Entwicklungsumgebungen und zahlreiche Technologien, welche die Entwicklung beschleunigen. Bei den Web-Anwendungen erfolgte eine ähnliche Evolution der Entwicklungstechniken wie bei den Desktop-Anwendungen. Auch für die Entwicklung von Web-Anwendungen stehen heute verschiedenste Technologien zur Verfügung, die den EntwicklerInnen großteils die Kodierung abnehmen. Es können zwei Gruppen von modernen Entwicklungstechniken für Web-Anwendungen unterschieden werden:

- *Komponentenbasierte Frameworks:* Diese Frameworks erlauben es, ähnlich wie bei der Entwicklung von Desktop-Anwendungen, die Anwendung aus vordefinierten Komponenten aufzubauen. Dazu wird meist eine MVC-Architektur zur Verfügung gestellt. Dabei kann die Benutzerschnittstelle oft mit einem graphischen Designer erstellt werden und der Quellcode für die Programmlogik bzw. das Datenmodell kann separat kodiert werden. Diese Frameworks generieren dann eine Web-Anwendung, welche den generierten Code (je nach Zielplattform z.B. html, jsp, asp, php…) und zusätzliche Konfigurationsdateien für den Webserver enthält. Beispiele für komponentenbasierte Frameworks sind: ASP.Net, Silverlight, JSF, Rails, PHP, GWT, Wicket,..

- *Auf MDA basierende Generatoren:* Dies sind Code-Generatoren, welche auf einer MDA „Model driven Architecture“ basieren. Diese Generatoren erlauben eine Erstellung der Web-Anwendung aus abstrakten Modellen (z.B. aus UML, Webml,..). Das Datenmodell und die Programmlogik bzw. Benutzer-

schnittstelle werden graphisch in einem Modell-Editor erstellt. Aus den abstrakten Modellen wird dann der Code für die Web-Anwendung generiert. Dieser Ansatz ermöglicht es zum Teil, dass gar kein Code manuell implementiert werden muss. Beispiele für modellbasierte Code-Generatoren sind: Arcstyler, Webratio, Andromeda, Blue Age, Ca Plex,..

Was diese Ansätze gemeinsam haben ist, dass sie einen großen Teil des Codes für die Web-Anwendung generieren. Dies hat aber folgen für die Qualität des Codes. Beide Ansätze verfolgen bei der Codegenerierung hauptsächlich funktionale Aspekte. Qualitätsaspekte wie die Barrierefreiheit im Web werden noch zu wenig berücksichtigt.

Aufgrund der in Kapitel 1 beschriebenen Problemstellung wird der dazu gewählte Lösungsansatz umgesetzt. Dabei wird eine fiktive Web-Anwendung (Referenzanwendung), mit Hilfe von vier Softwareentwicklungstechniken, jeweils als dynamische Web-Anwendung abgebildet bzw. implementiert. Die Referenzanwendung existiert bereits als statischer Prototyp, welcher größtenteils barrierefrei ist. Die generierten HTML-Seiten dieser Web-Anwendungen werden dann als Basis für eine Evaluierung der Barrierefreiheit nach den WCAG1.0 herangezogen. Welche Entwicklungstechniken hierfür ausgewählt werden, wie die Referenzanwendung spezifiziert ist und wie diese dynamisch abgebildet wird, wird im weiteren Verlauf dieses Kapitels beschrieben.

4.2 Auswahl von vier Softwareentwicklungstechniken

In der Praxis werden die komponentenbasierten Frameworks *ASP.Net* (Microsoft) und *Java Server Faces* (Sun) häufig eingesetzt, sie werden daher für die Implementierung der Referenzanwendung verwendet. Für die auf MDA basierenden Generatoren fällt die Wahl auf *Arcstyler* (Interactive Objects) und *Webratio* (Web Models). Eine genauere Beschreibung der verwendeten Plattformen, wird dann in Abschnitt 4.4 erfolgen. Eine Referenzanwendung, die in Form eines statischen Prototyps (Studenten-Informationssystem) existiert, soll mit diesen vier Entwicklungsansätzen dynamisch weiter entwickelt werden. Diese Referenzanwendung wird im folgenden Abschnitt beschrieben.

4.3 Die Referenzanwendung

Die Referenzanwendung ist bereits als statischer Prototyp vorhanden. Sie stellt ein Studenteninformationssystem dar, bei welchem nach Lehrveranstaltungen und ProfessorInnen gesucht werden kann. Weiters ist ein *CSS-Stylesheet* vorhanden, welches das Layout der Web-Seiten definiert. Der Quellcode der Referenzanwendung ist in Anhang A referenziert. Die verwendete Referenzanwendung ist eher einfach gehalten und enthält grundlegende *Markup-Elemente*, welche oft in Informationssystemen vorkommen. Die wichtigsten Elemente sind: Überschriften, Listen, Links, Text, Tabellen, Formulare. Es gibt einen *Trade-off* zwischen der Anzahl der Inhaltstypen und der Komplexität bei der Implementierung der Referenzanwendung. Natürlich wäre es wünschenswert, für den Vergleich der Entwicklungstechniken, so viele Komponententypen und Inhaltstypen wie nur möglich zu evaluieren. Allerdings steht der Zeitaufwand bei der Implementierung, dann in keinem Verhältnis zum erzielten Ergebnis. Da angenommen wird, dass auch die begrenzte Anzahl von Element-Typen und Komponententypen in der Referenzanwendung, eine differenzierte Aussage bezüglich der Barrierefreiheit des generierten Codes zulässt.

Der statische Prototyp besteht aus fünf XHTML-Seiten, die dem Dokumenttyp „XHTML 1.1 strict“ entsprechen. Die Inhalte dieser fünf statischen Seiten werden im Folgenden als Screenshots dargestellt und jeweils kurz beschrieben:

In Abbildung 4.1 ist die Einstiegsseite der Anwendung dargestellt. Sie hat im Anhang A den Dateinamen: Index.html. Sie enthält Text-Elemente, Überschriften und Links. Das Basis-Layout, welches für alle Seiten gleich ist, besteht aus einer *headerbar*, einem Navigationsbereich mit den Links (Vorlesungen, ProfessorInnen und Home) und einem Bereich für die Inhalte *container*. Das Layout für den Inhalts-Container ist in Regionen aufgeteilt. Es gibt darin einen zweiten Navigationsbereich *sidebar*, welcher auf der linken Seite ersichtlich ist, einen Bereich für die Inhalte *main* in der Mitte und eine Fußzeile *footer*, welche unten als grüne Zeile zu erkennen ist.

Seite 1: Home

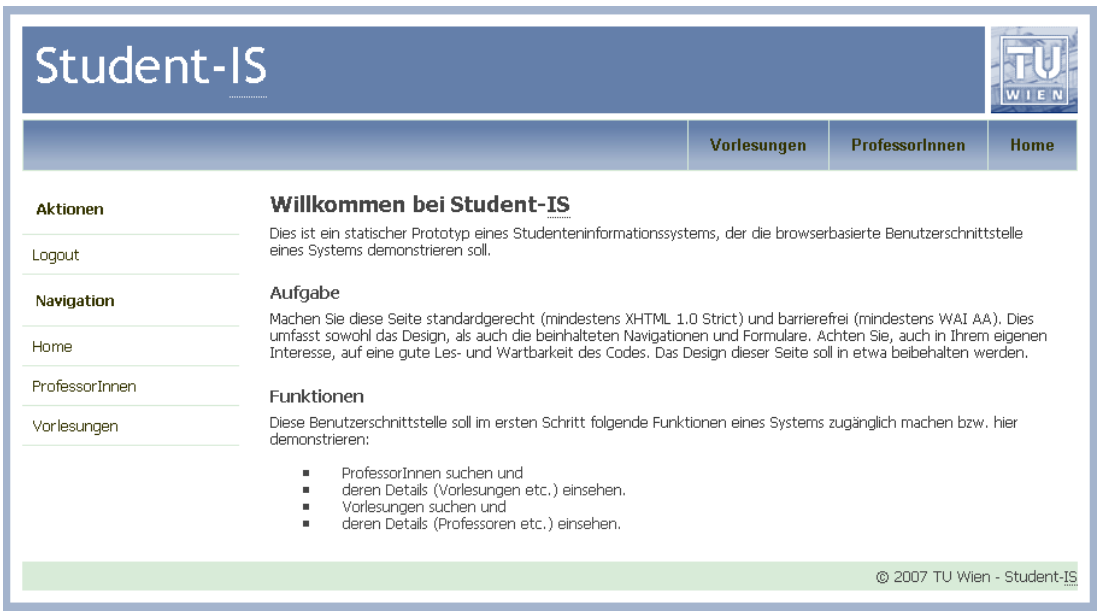

Abbildung 4.1 Home-Seite des Informationssystems

Vorlesungen können über den Link „Vorlesungen“ ausgewählt werden (siehe Abbildung 4.2). Sie hat im Anhang A den Dateinamen: Lectures.html. Diese Seite enthält ein Suchformular über das Lehrveranstaltungen gesucht werden können. Die Ergebnisse der Suche werden in Form einer Tabelle, auf der gleichen Seite angezeigt.

Seite 2: Vorlesungen

Abbildung 4.2 Suchen von Vorlesungen im Informationssystem

VorlesungenDetails können über den Link in der Tabelle von Abbildung 4.2 ausgewählt werden (siehe Abbildung 4.3). Sie hat im Anhang A den Dateinamen: Lecture-details.html. Sie enthält Details über die Vorlesung und den Lehrveranstaltungsleiter.

Seite 3: VorlesungenDetails

Abbildung 4.3 Detaillierte Auskunft über eine Vorlesung

ProfessorInnen können über den Link „ProfessorInnen“ ausgewählt werden (siehe Abbildung 4.4). Sie hat im Anhang A den Dateinamen: Professors.html. Sie ist analog zur Vorlesungssuche aufgebaut.

Seite 4: ProfessorInnen

Abbildung 4.4 Suche nach ProfessorInnen im Informationssystem

ProfessorInnenDetails können über einen Link in der Tabelle von Abbildung 4.4 ausgewählt werden (siehe Abbildung 4.5). Sie hat im Anhang A den Dateinamen: Professordetails.html. Diese Seite enthält Details über den Lehrveranstaltungsleiter und seine Lehrveranstaltungen.

Seite 5: ProfessorInnenDetails

Abbildung 4.5 Detaillierte Auskunft über ProfessorInnen und Vorlesungen

4.4 Implementierung der Referenzanwendung

Diese statischen HTML-Seiten werden nun mit Hilfe der folgenden Komponenten-Frameworks und Code-Generatoren, jeweils als dynamische Web-Anwendung implementiert. Welche Plattformen und Technologien im Detail angewendet werden, wird nun in den folgenden Abschnitten beschrieben.

4.4.1 Datenmodell und Datenbank

Für die Implementierung ist es zuerst notwendig eine kleine Datenbank zu erstellen, welche die Daten des Informationssystems enthält. Für die Datenbank wird MySQL 5.0 verwendet. Die Webserver für die JSF- und Webratio-Anwendungen, greifen über einen JDBC-Treiber auf die Datenbank zu. Für ASP.Net und Arcstyler kommen jedoch andere Datenquellen zum Einsatz. Die Datenbank besteht aus einer Tabelle für die Lehrveranstaltungen und einer Tabelle für die Lehrveranstaltungsleiter. Das Datenbankschema (Studentis) ist in Abbildung 4.6 ersichtlich:

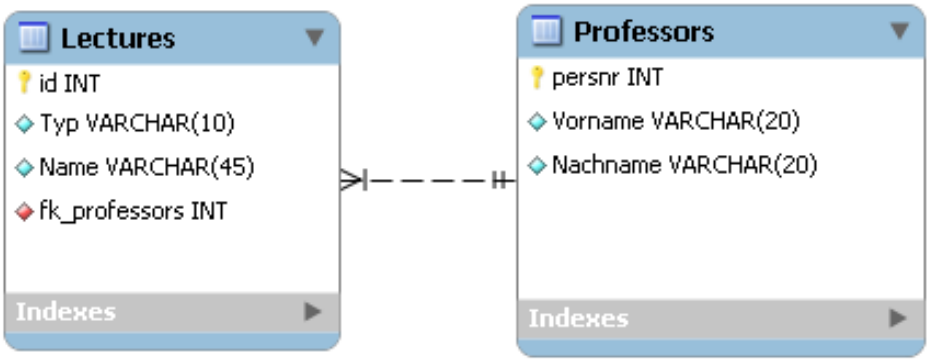

Abbildung 4.6 Datenbankschema für das Studenteninformationssystem

Das SQL-Skript für die Erstellung der Datenbank ist im Anhang A referenziert.

4.4.2 ASP.Net

ASP.Net[1] ist eine Technologie der Firma Microsoft, für die komponentenbasierte Erstellung von Web-Anwendungen. Als Entwicklungsumgebung dient der „Microsoft Visual Web Developer 2008 Express Edition“, mit dem .Net-Framework der Version 3.5. ASP.Net erlaubt die Programmierung der Programmlogik mit C# oder Visual Basic. Hier wird C# gewählt. Als Datenbank wird eine Access-Datei verwendet. Die generierte Web-Anwendung ist lauffähig unter dem „Microsoft-IIS“ bzw. dem integrierten Webserver der Entwicklungsumgebung. Die generierten HTML-Seiten haben die Endung: *aspx*. Die generierten C#-Dateien haben die Endung: *aspx.cs*.

4.4.3 Java Server Faces

Java Server Faces[2] (JSF) ist eine Technologie von Sun und ist auch ein serverseitiger, komponentenbasierter Ansatz zur Entwicklung von Web-Anwendungen. JSF basiert auf Servlets und JSP. Von JSF gibt es mehrere Implementierungen, da Sun mit dieser *Technologie-Spezifikation* einen Standard schaffen möchte. Beispiele für Implementierungen sind: MyFaces (Apache), ICEFaces (Ice soft). RichFaces (RedHat). Sun hat selbst eine Referenzimplementierung von JSF erstellt und diese unter dem „Mojarra-Project“ veröffentlicht. In dieser Arbeit wird die Entwicklungsumgebung Netbeans 6.1 von Sun verwendet. Sie ermöglicht die Erstellung von JSF 1.2 Web-Anwendungen mit der „Woodstock-Componentlibrary“. Mit Netbeans ist es möglich ein WAR-Archiv der Web-Anwendung zu erstellen, dieses ist unter den Webservern Tomcat 6.0 und Glassfish 2.0 lauffähig. Die Anwendung wurde auf Tomcat 6.0 getes-

[1] http://www.asp.net/, (Stand: 6.5.2009)

[2] http://java.sun.com/javaee/javaserverfaces/, (Stand: 6.5.2005)

tet. Die Daten werden von Tomcat über JDBC von der MySQL-Datenbank abgefragt. JSF generiert jsp- und java-Dateien.

4.4.4 Webratio

Webratio[1] von der Firma *Web Models*, ist ein auf MDA basierender Code-Generator für Web-Anwendungen und verwendet die Modellierungssprache *Webml*. Es wird die Webratio Version 5.0 verwendet. Diese Version verwendet Eclipse als Entwicklungsumgebung. Webratio enthält den integrierten Tomcat 5.5 Webserver. Webratio kann über JDBC auf die MySQL-Datenbank zugreifen und das Datenmodell intern replizieren.

4.4.5 Arcstyler

Arcstyler[2] von der Firma *Interactive Objects*, ist ein auf MDA basierender Code-Generator für die verschiedensten Zielplattformen. Arcstyler verwendet UML zur Modellierung und ermöglicht die Transformation dieser Modelle auf die jeweilige Zielplattform. Arcstyler stellt dafür so genannte *Cartridges* zur Verfügung z.B. Jboss, Weblogic, .Net, C#, Java2, WebAccessor. In dieser Arbeit wird die Referenzanwendung mit Hilfe der Cartridges: *WebAccessor* (Web-Anwendungsentwicklung) und *Jboss* (Anwendungs-Server) erstellt. Als Datenquelle kommen *Enterprise Java Beans* (EJB's) zum Einsatz, deren Modellierung von Arcstyler mit UML unterstützt wird und deren Persistenz der Anwendungs-Server ermöglicht. Es wird die Version Arcstyler 5.5 verwendet. Diese Version enthält den integrierten Webserver Tomcat 4.0. Arcstyler generiert aus UML Java-Dateien und Jsp-Dateien. Die Programmlogik muss in den generierten Java-Dateien zum Teil manuell angepasst werden.

Diese hier aufgeführten Entwicklungstechniken haben nicht viel gemeinsam. ASP.Net ist eine Microsoft-Technologie und die anderen drei unterstützen zumindest einen *Servlet-Container* (z.B. Tomcat) als Zielplattform. Bei der Art und Weise wie die Web-Anwendungen modelliert und implementiert werden, unterscheiden sich die Technologien beträchtlich. Bei Webratio kann diese Anwendung ohne manuelle Erstellung von Code generiert werden. Bei Arcstyler sind umfangreiche Anpassungen des Quellcodes nötig. ASP.Net und JSF stellen einen grafischen Designer zur Verfügung, mit dessen Hilfe die Komponenten auf der Web-Seite angeordnet werden können. Alle vier Entwicklungsprojekte sind in Anhang A referenziert.

---

[1] http://www.webratio.com/, (Stand: 6.5.2009)

[2] http://www.arcstyler.com/, (Stand: 6.5.2009)

5 Evaluierung der generierten Web-Seiten

Die im vorigen Kapitel beschriebene Referenzanwendung, wurde mit den vier beschriebenen Entwicklungstechniken implementiert. Die Web-Anwendungen wurden jeweils im Browser getestet und die so entstandenen HTML-Seiten gespeichert. Sie bilden die Basis für die Evaluierung der Barrierefreiheit. In diesem Kapitel werden die Rahmenbedingungen und der Ablauf der Evaluierung beschrieben.

5.1 Hintergrund und Ziele der Evaluierung

Das Ziel der Evaluierung ist, festzustellen wie gut die verwendeten Entwicklungstechniken, für die Erstellung von barrierefreien Webanwendungen geeignet sind. Leider kann die Evaluierung keine erschöpfende Aussage darüber liefern, da nicht alle erdenklichen Kriterien der Barrierefreiheit getestet werden können. So ist die Anzahl der Element-Typen in der Referenzanwendung auf grundlegende Elemente beschränkt. Die Kriterien betrachten hauptsächlich die Prüfpunkte der WCAG1.0, von denen in diesem Fall gar nicht alle anwendbar sind. Außerdem gäbe es noch viele andere Kriterien, die für die Prüfung der Web-Anwendungen herangezogen werden könnten. Wichtig ist jedoch der Vergleich zwischen den verwendeten Entwicklungstechniken, bzw. der Vergleich zwischen der Gruppe der komponentenbasierten Frameworks und der Code-Generatoren. Am Ende der Evaluierung sollte klar sein, welche Stärken und Schwächen bei den verwendeten Softwareentwicklungstechniken, hinsichtlich der Barrierefreiheit des generierten Codes existieren.

5.2 Umfang der Evaluierung und Evaluierungsobjekte

Zu Beginn der Evaluierung, müssen die Evaluierungsobjekte definiert werden. Bei dieser Evaluierung werden pro Web-Anwendung, je zwei der generierten HTML-Seiten als Evaluierungsobjekte herangezogen. Bei jeder der implementierten Web-Anwendungen, werden jene zwei HTML-Seiten ausgewählt, die am meisten dynamisch generierte Elemente enthalten. Dies sind die Seiten: Vorlesungen.html und VorlesungenDetail.html. Die Seiten ProfessorInnen.html und ProfessorInnenDetail.html sind analog zu den genannten aufgebaut und müssen daher nicht extra evaluiert werden. Der Quellcode der zu evaluierenden Web-Seiten ist in Anhang A referenziert.

Diese Web-Seiten wurden mit dem Browser Firefox 2.0 lokal gespeichert. Zum Beispiel sieht die Seite Vorlesungen.html, optisch jeweils so aus:

ASP.Net

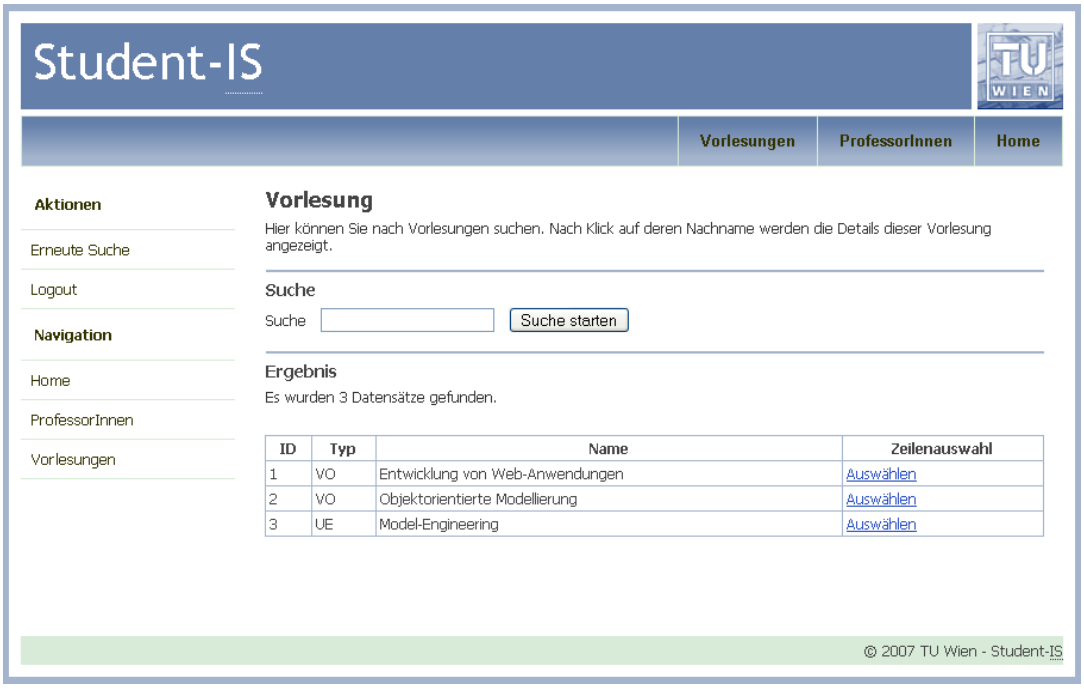

Abbildung 5.1 ASP.Net, Evaluierungsobjekt Vorlesungen.html

JSF

Abbildung 5.2 JSF, Evaluierungsobjekt Vorlesungen.html

Webratio

Abbildung 5.3 Webratio, Evaluierungsobjekt Vorlesungen.html

Arcstyler

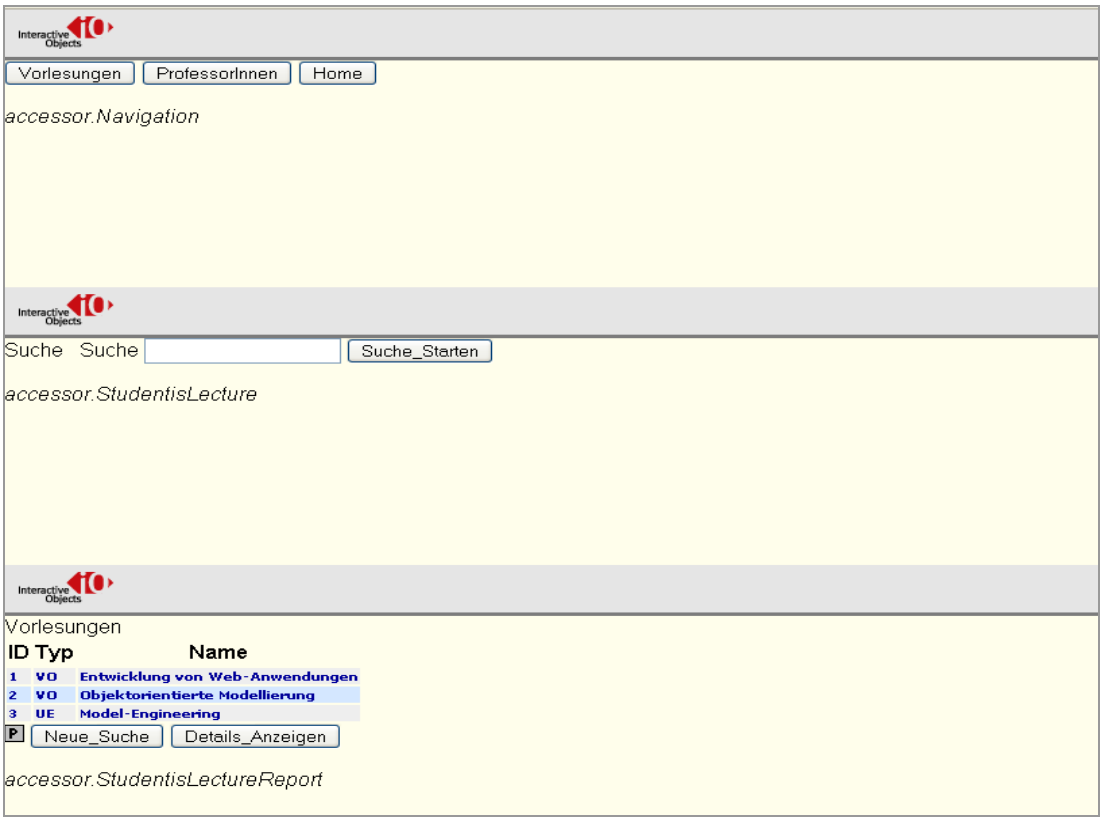

Abbildung 5.4 Arcstyler, Evaluierungsobjekt Vorlesungen.html

Mit Arcstyler war es nicht möglich, dass vorgegebene CSS-Layout zu verwenden.

5.3 Kriterien für die Evaluierung

Die Kriterien für die Evaluierung entsprechen den Prüfpunkten der WCAG1.0. Es werden die Prüfpunkte der Priorität 1 (A) und der Priorität 2 (AA) als Bewertungsgrundlage herangezogen. Prüfpunkte der Priorität 3 (AAA) werden nicht berücksichtigt, da sie in gesetzlichen Bestimmungen noch nicht üblich sind. Da für die Referenzanwendung nicht alle Prüfpunkte relevant sind, werden nur jene Punkte verwendet, zu denen es in der Referenzanwendung auch inhaltliche Elemente gibt. Zum Beispiel gibt es in der Referenzanwendung keine Multimedia-Komponenten. Im folgenden werden alle WCAG1.0-Prüfpunkte der Stufe A und AA aufgezählt und es werden all jene Prüfpunkte gekennzeichnet, die für die Evaluierung relevant sind. Die einzelnen Prüfpunkte sind nummeriert. Dabei kennzeichnet die Ziffer vor dem Punkt die jeweilige Richtlinie aus den WCAG1.0 und die Ziffer nach dem Punkt kennzeichnet den jeweiligen Prüfpunkt zur Richtlinie. Die Nummerierung ist somit kompatibel mit den WCAG1.0.

5.3.1 Kriterien aus den WCAG1.0 der Priorität 1

Kriterium	Prüfpunkte der WCAG 1.0
	Allgemein (Priorität 1)
Ja	1.1 Stellen Sie ein Text-Äquivalent für jedes Nicht-Text-Element bereit (z.B. über „alt“, „longdesc“ oder im Inhalt des Elements). Dies umfasst: Bilder, grafisch dargestellten Text (einschließlich Symbole), Regionen von Imagemaps, Animationen (z.B. animierte GIFs), Applets und programmierte Objekte, ASCII-Zeichnungen, Frames, Scripts, Bilder, die als Punkte in Listen verwendet werden, Platzhalter-Grafiken, grafische Buttons, Töne (abgespielt mit oder ohne Einwirkung des Benutzers), Audio-Dateien, die für sich allein stehen, Tonspuren von Videos und Videos.
Ja	2.1 Sorgen Sie dafür, dass die gesamte mit Farbe dargestellte Information auch ohne Farbe verfügbar ist, z.B. im Kontext oder im Markup.
Ja	4.1 Machen Sie in klarer Weise Änderungen der natürlichen Sprache des Dokumententexts und sämtlicher Text-Äquivalente kenntlich.
Ja	6.1 Bauen Sie Dokumente so auf, dass sie ohne Stylesheets gelesen werden können. Z.B. wenn ein HTML-Dokument ohne ihm zugeordnete Stylesheets dargestellt wird, muss es immer noch möglich sein, das Dokument zu lesen.
Ja	6.2 Sorgen Sie dafür, dass Äquivalente für dynamischen Inhalt aktualisiert werden, wenn sich der dynamische Inhalt ändert.
Ja	7.1 Vermeiden Sie Bildschirmflackern, bis Benutzeragenten dem Benutzer eine Kontrolle über das Flackern ermöglichen.
Nein	14.1 Verwenden Sie für den Inhalt einer Site die klarste und einfachste Sprache, die angemessen ist.
	Und wenn Sie Bilder und Imagemaps verwenden (Priorität 1)
Nein	1.2 Stellen Sie redundante Textlinks für jede aktive Region einer Server-seitigen Imagemap bereit.
Nein	9.1 Stellen Sie Client-seitige anstelle von Server-seitigen Imagemaps bereit, außer wenn die Regionen mit den verfügbaren geometrischen Formen nicht definiert werden können.

Kriterium	Prüfpunkte der WCAG 1.0
	Und wenn Sie Tabellen verwenden (Priorität 1)
Ja	5.1 Kennzeichnen Sie bei Datentabellen Zeilen- und Spaltenüberschriften.
Nein	5.2 Wenn Datentabellen zwei oder mehr logische Ebenen von Zeilen- oder Spaltenüberschriften haben, verwenden Sie Markup, um Datenzellen und Überschriftenzellen einander zuzuordnen.
	Und wenn Sie Frames verwenden (Priorität 1)
Ja	12.1 Betiteln Sie jeden Frame, um Navigation und Identifikation zu erleichtern.
	Und wenn Sie Applets und Scripts verwenden (Priorität 1)
Ja	6.3 Sorgen Sie dafür, dass Seiten verwendbar sind, wenn Scripts, Applets oder andere programmierte Objekte abgeschaltet sind oder nicht unterstützt werden. Ist dies nicht möglich, stellen Sie äquivalente Information auf einer alternativen zugänglichen Seite bereit.
	Und wenn Sie Multimedia verwenden (Priorität 1)
Nein	1.3 Stellen Sie eine Audio-Beschreibung der wichtigen Information der Videospur einer Multimedia-Präsentation bereit, bis Benutzeragenten das Text-Äquivalent einer Videospur vorlesen können.
Nein	1.4 Synchronisieren Sie für jede zeitgesteuerte Multimedia-Präsentation (z.B. Film oder Animation) äquivalente Alternativen (z.B. Untertitel oder Audio-Beschreibungen der Videospur) mit der Präsentation.
	Und wenn alles andere fehlschlägt (Priorität 1)
Nein	11.4 Wenn Sie auch nach besten Bemühungen keine zugängliche Seite erstellen können, stellen Sie einen Link auf eine alternative Seite bereit, die W3C-Technologien verwendet, zugänglich ist, äquivalente Information (oder Funktionalität) enthält und ebenso oft aktualisiert wird wie die nicht zugängliche (originale) Seite.

Tabelle 5.1 Kriterien für die Evaluierung nach WCAG1.0, Priorität 1 [W3cc00]

5.3.2 Kriterien aus den WCAG1.0 der Priorität 2

Kriterium	Prüfpunkte der WCAG 1.0
	Allgemein (Priorität 2)
Ja	2.2 Sorgen Sie dafür, dass die Kombinationen aus Vordergrund- und Hintergrundfarbe ausreichend kontrastieren, wenn sie von jemandem betrachtet werden, dessen Farbensehen beeinträchtigt ist, oder wenn sie mit einem Schwarzweißbildschirm betrachtet werden.
Ja	3.1 Wenn eine angemessene Markup-Sprache existiert, verwenden Sie Markup anstelle von Bildern, um Information darzustellen.
Ja	3.2 Erstellen Sie Dokumente, die gegen veröffentlichte formale Grammatiken validieren.
Ja	3.3 Verwenden Sie Stylesheets, um Layout und Präsentation zu beeinflussen.
Ja	3.4 Verwenden Sie relative anstelle von absoluten Einheiten in den Attributwerten der Markup-Sprache und Stylesheet-Property-Werten.
Ja	3.5 Verwenden Sie Überschriften-Elemente, um die Struktur eines Dokuments darzustellen und verwenden Sie sie gemäß der Spezifikation.
Ja	3.6 Verwenden Sie korrekten Markup für Listen und Listenelemente.
Ja	3.7 Verwenden Sie Markup für Zitate. Verwenden Sie keinen Markup, der für Zitate gedacht ist, um visuelle Effekte wie Einrückung zu erzielen.
Ja	6.5 Sorgen Sie dafür, dass dynamischer Inhalt zugänglich ist oder stellen Sie eine alternative Präsentation oder Seite bereit.
Ja	7.2 Bis Benutzeragenten eine Kontrolle über das Blinken ermöglichen, vermeiden Sie es, Inhalt blinken zu lassen (d.h. die Präsentation regelmäßig zu ändern, z.B. ab- und einzuschalten).

Kriterium	Prüfpunkte der WCAG 1.0
	Allgemein (Priorität 2)
Ja	7.4 Bis Benutzeragenten es zulassen, den Refresh zu stoppen, erstellen Sie keine Seiten mit automatischer periodischer Aktualisierung.
Nein	7.5 Bis Benutzeragenten es zulassen, die automatische Weiterleitung (Redirect) zu stoppen, verwenden Sie keinen Markup, um eine Weiterleitung zu erzielen. Konfigurieren Sie stattdessen den Server so, dass er eine Weiterleitung ausführt.
Nein	10.1 Lassen Sie keine Pop-Ups oder andere Fenster erscheinen und wechseln Sie das aktuelle Fenster nicht, ohne den Benutzer zu informieren, bis Benutzeragenten es gestatten, die Erzeugung neuer Fenster zu unterbinden.
Ja	11.1 Verwenden Sie W3C-Technologien, wenn sie verfügbar und der Aufgabe angemessen sind und benutzen Sie die neueste Version, wenn sie unterstützt wird.
Ja	11.2 Vermeiden Sie überholte Features von W3C-Technologien.
Ja	12.3 Unterteilen Sie große Informationsblöcke in leichter zu handhabende Gruppen, wo angebracht.
Ja	13.1 Identifizieren Sie das Ziel jedes Links auf klare Weise.
Ja	13.2 Stellen Sie Metadaten bereit, um semantische Information zu Seiten und Sites hinzuzufügen.
Ja	13.3 Stellen Sie Informationen zum allgemeinen Layout einer Site bereit (z.B. über eine Sitemap oder ein Inhaltsverzeichnis).
Ja	13.4 Verwenden Sie Navigationsmechanismen in konsistenter Weise.
	Und wenn Sie Tabellen verwenden (Priorität 2)
Ja	5.3 Verwenden Sie keine Tabellen für Layout, wenn diese in linearisierter Form keinen Sinn ergeben. Ansonsten, wenn die Tabelle keinen Sinn ergibt, stellen Sie ein alternatives Äquivalent bereit (das eine linearisierte Version sein kann).
Nein	5.4 Wenn eine Tabelle für Layout verwendet wurde, verwenden Sie keinen Struktur-Markup zum Zweck der visuellen Formatierung.
	Und wenn Sie Frames verwenden (Priorität 2)
Ja	12.2 Beschreiben Sie den Zweck von Frames und ihre Beziehung untereinander, wenn dies aus den Titeln allein nicht ersichtlich wird.
	Und wenn Sie Formulare verwenden (Priorität 2)
Ja	10.2 Sorgen Sie bei allen Formular-Kontrollelementen mit implizit zugeordneten Beschriftungen dafür, dass die Beschriftung korrekt positioniert ist, bis Benutzeragenten eine explizite Zuordung von Beschriftung und Formular-Kontrollelement ermöglichen.
Ja	12.4 Ordnen Sie Beschriftungen explizit ihren Kontrollelementen zu.
	Und wenn Sie Applets und Scripts verwenden (Priorität 2)
Ja	6.4 Sorgen Sie dafür, dass die Eingabebehandlung von Scripts und Applets vom Eingabegerät unabhängig ist.
Nein	7.3 Vermeiden Sie Bewegung in Seiten, bis Benutzeragenten das Einfrieren von Bewegung ermöglichen.
Ja	8.1 Machen Sie programmierte Elemente wie Scripts und Applets direkt zugänglich oder kompatibel mit assistiven Technologien
Nein	9.2 Sorgen Sie dafür, dass jedes Element, das über seine eigene Schnittstelle verfügt, in geräteunabhängiger Weise bedient werden kann.
Nein	9.3 Spezifizieren Sie in Scripts logische Event-Handler anstelle von geräteabhängigen Event-Handlern.

Tabelle 5.2 Kriterien für die Evaluierung nach WCAG1.0, Priorität 2 [W3cc00]

Insgesamt wurden 9 Prüfpunkte der Priorität 1 und 24 Prüfpunkte der Priorität 2 ausgewählt.

5.4 Evaluierungshilfsmittel

Für die Evaluierung der Barrierefreiheit, kommen eine Reihe von Prüfprogrammen zur Anwendung. Über die grundsätzlichen Möglichkeiten der Prüfprogrammen, wurde in Kapitel 3 berichtet. In folgender Tabelle werden all jene Prüfprogramme aufgelistet, die für die automatische Evaluierung verwendet werden.

Programm	**Ver.**	**Beschreibung**
ACCVerify[1]	10	Demo-Version eines professionellen Prüfprogrammes für komplette Webseiten. Unterstützt unterschiedliche Richtliniensätze und erstellt HTML-Reports.
TAW[2]	3.08	Freies WCAG1.0-Prüfprogramm für komplette Webseiten, welches auch HTML-Reports erstellt.

Tabelle 5.3 Prüfprogramme für die automatische Evaluierung

Die folgende Tabelle enthält jene Prüfprogramme die für die teilautomatische und manuelle Evaluierung verwendet werden. Bei diesen Prüfprogrammen handelt es sich meist um „Browser-Plugins“. Diese haben den Vorteil, dass dynamische generierte Inhalte (z.B. das Ergebnis einer Suche), damit erst erfassbar werden.

Programm	**Ver.**	**Beschreibung**
Fangs[3]	1.0.4	Screenreader Emulator als Firefox Add On, liefert die Ausgabe eines Screenreaders in Textform
Firefox Accessibility Extension[4]	1.4.5	Firefox Add On. Prüfprogramm in Form einer Werkzeugleiste im Browser, erlaubt verschiedenste Tests.

[1] http://www.hisoftware.com/products/accverify.html (Stand: 2.3.2009)

[2] http://www.tawdis.net/taw3/cms/en (Stand: 2.3.2009)

[3] http://www.standards-schmandards.com/projects/fangs/ (Stand: 2.3.2009)

[4] http://firefox.cita.uiuc.edu/ (Stand: 2.3.2009)

Fortsetzung Prüfprogramme für die Evaluierung:

Programm	**Ver.**	**Beschreibung**
Firefox Web-browser[1]	2.0	Moderner grafischer Web-Browser
W3C Markup Validation Servi-ce[2]	0.8.4	Dieses Webtool prüft das Dokumenten-Markup gegen veröffentlichte Dokumenttypen wie z.B.: HTML, XHTML, SMIL,…
W3C CSS Valida-tion Service[3]	KA	Validierungsservice für CSS (Cascading Stylesheets)
WAT Toolbar[4]	2.0	Internet Explorer Plugin. Prüfprogramm in Form einer Werkzeugleiste im Browser, erlaubt verschiedenste Tests.
WAVE Toolbar[5]	0.9.4	Freies Accessibility-Evaluierungs-Werkzeug in Form einer Werkzeugleiste in Firefox.

Tabelle 5.4 Prüfprogramme für teilautomatische und manuelle Evaluierung

5.5 Evaluierungsmethode

Die Evaluierung selbst ist ein Prozess in drei Stufen, angelehnt an den in Abschnitt 3.2 vorgeschlagenen Evaluierungsprozess der W3C. Zuerst erfolgt mit Hilfe von zwei automatischen Prüfprogrammen eine automatische Evaluierung gegen die WCAG1.0. Da nicht alle Kriterien automatisch bewertet werden können erfolgt im zweiten Schritt eine teilautomatische Evaluierung mit kriterienspezifischen Prüfprogrammen. Im dritten und letzten Schritt, erfolgt eine manuelle bzw. visuelle Evaluierung im Browser bzw. mit speziellen Browsern. Dabei werden jene Kriterien überprüft die bei der automatischen bzw. teilautomatischen Evaluierung noch nicht bewertet werden konnten.

---

[1] http://www.mozilla.com/en-US/ (Stand: 2.3.2009)

[2] http://validator.w3.org/ (Stand 2.3.2009)

[3] http://jigsaw.w3.org/css-validator/ (Stand 2.3.2009)

[4] http://www.wat-c.org/tools/ (Stand 2.3.2009)

[5] http://wave.webaim.org/ (Stand 2.3.2009)

5.5.1 Automatische Evaluierung

Für die automatische Evaluierung stehen zwei Prüfprogramme zur Verfügung. Zuerst werden die zu evaluierenden HTML-Seiten mit dem professionellen WCAG1.0-Prüfprogramm *ACCVerify* evaluiert. Dieses Programm generiert detaillierte HTML-Berichte über die Konformität der getesteten Web-Seiten mit den WCAG1.0. Im zweiten Schritt der automatischen Evaluierung kommt das freie WCAG1.0-Prüfprogramm *TAW* zum Einsatz. Dieses Programm ist auch in der Lage HTML-Berichte zu generieren. Allerdings bietet dieses Programm weniger Funktionalität als ACCVerify und die Fähigkeiten zur automatischen Erkennung von WCAG-Verletzungen sind nicht so gut ausgeprägt wie bei ACCVerify. Eine manuelle Kontrolle der Evaluierungsergebnisse ist bei TAW daher öfter nötig als bei ACCVerify. Grundsätzlich liefern die automatischen WCAG-Prüfprogramme brauchbare Ergebnisse, jedoch liefern sie nicht immer eine korrekte und vollständige Bewertung. Eine manuelle Überprüfung der Ergebnisse ist daher oft nötig. Ein weiterer Punkt den es bei der automatischen Prüfung zu beachten gilt, ist die Verschiedenheit der Prüfalgorithmen. Die beiden Programme liefern zum Teil unterschiedliche Ergebnisse. Das ist der Grund warum zwei Prüfprogramme verwendet werden. Missinterpretationen der einzelnen Programme können so leichter erkannt werden. Der Vorteil der automatischen Evaluierung liegt in der Zeiteffizienz der Evaluierung, da alle WCAG-Prüfpunkte von einem Programm geprüft werden können. Der Nachteil der automatischen Prüfung liegt in der Ungenauigkeit und der fehlenden Effektivität der Prüfalgorithmen. Es kann nicht garantiert werden, dass die geprüften Web-Seiten nach Korrektur der gefundenen Barrieren, wirklich für alle Benutzer Barrierefrei sind.

5.5.2 Teilautomatische Evaluierung

Der zweite Schritt im Evaluierungsprozess ist die teilautomatische Evaluierung. In diesem Schritt werden all jene Kriterien geprüft, die bei der automatischen Evaluierung, von den WCAG-Prüfprogrammen nicht bewertet werden konnten. Als Hilfsmittel für die teilautomatische Evaluierung kommen spezielle Prüfprogramme zum Einsatz, die jeweils nur ein bestimmtes Kriterium überprüfen können. Eine vollständige Auflistung dieser Prüfprogramme ist in Abschnitt 5.4 ersichtlich. Dieser Evaluierungsschritt ist der zeitaufwändigste. Da jedes noch nicht bewertete Kriterium, bei jeder Web-Seite, mit dem jeweiligen Prüfprogramm getestet werden muss. Es kann aber auch vorkommen, dass die jeweiligen Kriterien bereits visuell im Browser evaluiert werden können und es gar nicht nötig ist ein spezielles Prüfprogramm zu verwenden. Besonders hilfreich für diesen Schritt sind Prüfprogramme, die als Browser-Plugin in Form von Werkzeugleisten zur Verfügung stehen. Damit lassen sich dann alle ge-

wünschten Veränderungen, wie z.B. abschalten von Scripts, deaktivieren von Stylesheets, hervorheben von Überschriften oder Strukturelementen, etc. bequem im Browser durchführen. Meistens liefern die automatischen Prüfprogramme auch noch Hinweise in Richtung potenzieller Verletzungen, auch wenn das entsprechende Kriterium nicht automatisch bewertet werden konnte. Diese Hinweise müssen dann visuell oder mit Hilfe der speziellen Prüfprogramme verifiziert werden. Als Leitlinie für die teilautomatische Evaluierung wird das Dokument *Techniques for Web Content Accessibility Guidelines 1.0* [W3ct00] verwendet, da hier die Bedingungen der Erfüllung bzw. Nicht-Erfüllung von Prüfpunkten genauer spezifiziert sind.

5.5.3 Manuelle Evaluierung

Als manuelle Evaluierung werden all jene Prüfschritte bezeichnet, die ohne spezielle Softwareunterstützung auskommen und bei denen die Kriterien mit konventionellen Mitteln (z.B. Browser, Tastatur und Maus), manuell bzw. visuell beurteilt werden können.

5.5.4 Bewertungsschema

Im Gegensatz zur Evaluierung von ganzen Web-Auftritten mit mehreren Seiten, geht es bei dieser Evaluierung um den Vergleich der Barrierefreiheit des generierten Codes, von den zuvor gewählten Komponenten-Frameworks und Code-Generatoren. Das Ziel bei der Bewertung ist es, festzustellen welche Verletzungen der definierten Kriterien aufgetreten sind. Wie oft eine Verletzung des gleichen Kriteriums vorkommt, spielt für diese Evaluierung eine untergeordnete Rolle. Da bereits eine einmalige Verletzung eines Kriteriums eine Aussage über die Qualität des generierten Codes Aufschluss geben kann. Interessant ist hingegen wie viele Kriterien von einer bestimmten Web-Anwendung verletzt worden sind. Für die Bewertung der Kriterien wird folgendes Schema verwendet:

Beschreibung	**Bewertung**
Das Kriterium wurde erfüllt	Ja
Das Kriterium wurde nicht erfüllt	Nein
Für das entsprechende Kriterium wurden keine relevanten Elemente gefunden.	KA

Tabelle 5.5 Bewertungsschema für die Evaluierung

Aufgrund der unterschiedlichen Evaluierungs-Methoden wird der Bewertung noch eine Information über die Art der Evaluierung des entsprechenden Kriteriums hinzugefügt. Diese Information wird hier als Bewertungsmodus bezeichnet. Es gibt dabei folgende Modi:

Beschreibung	**Bewertungsmodus**
Die Bewertung erfolgte automatisch mit dem WCAG-Prüfprogramm ACCVerify	A1
Die Bewertung erfolgte automatisch mit dem WCAG-Prüfprogramm TAW	A2
Die Bewertung erfolgte manuell/visuell im Browser oder auf teilautomatische Weise, mit Hilfe von kriterienspezifischen Werkzeugen	M

Tabelle 5.6 Bewertungsmodus für die Evaluierung

5.5.5 Berichtformat und Aufbereitung der Ergebnisse

Die Ergebnisse der Evaluierung stammen entweder aus den generierten Berichten der Prüfprogramme oder aus der visuellen Beobachtung, bei der Verwendung von browserbasierten Prüfprogrammen. Diese Ergebnisse müssen in ein einheitliches Format gebracht werden. Dazu wird folgende Excel-Tabelle verwendet. Diese Tabelle wird für jede evaluierte Seite ausgefüllt. Mit dieser Darstellung der Ergebnisse, ist es möglich einen Vergleich zwischen den Entwicklungstechniken herzustellen.

Nr.	**Evaluierungskriterien**	**Pr.**	**Beschreibung**	**Bw.**	**Zeilen-Nr.**	**Mod.**
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1				
2.1	Informationserhalt bei fehlen von Farbe	1				
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1				
6.1	Lesbarkeit ohne Style-sheets	1				
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1				
7.1	Vermeidung von Bildschirmflackern	1				
2.2	Guter Kontrast bei Farben	2				
3.1	Markup statt Bilder	2				
3.2	Einhaltung formaler Grammatiken	2				
3.3	Verwendung von Style-sheets	2				
3.4	Verwendung von relativen Einheiten	2				
3.5	Korrekte Darstellung der Dokumentenstruktur	2				

Fortsetzung Berichtformat:

3.6	Korrektes Markup für Listen	2				
3.7	Korrektes Markup für Zitate	2				
6.5	Zugänglichkeit von dynamischen Inhalten	2				
7.2	Vermeidung von blinkenden Inhalten	2				
7.4	Vermeidung von autom. Refresh der Seite	2				
11.1	Verwendung von W3C-Technologien	2				
11.2	Vermeidung von überholten W3C-Features	2				
12.3	Unterteilung von Informationsblöcken	2				
13.1	Kennzeichnung von Links	2				
13.2	Verwendung von Metadaten	2				
13.3	Bereitstellung von Layout-Informationen	2				
13.4	Konsistente Nutzung von Navigationsmechanismen	2				
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1				
5.3	Tabellen nicht für Layout-Zwecke verwenden	2				
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1				
6.4	Eingabegeräte unabhängige Event-Handler definieren	2				
8.1	Für assistive Technologien zugänglich machen	2				
	Frames					
12.1	Betiteln der Frames	1				
12.2	Genauere Beschreibung der Frames	2				
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2				
12.4	Explizite Zuordnung von Beschriftungen	2				

Tabelle 5.7 Berichtformat für die Evaluierungsergebnisse

Beschreibung der Spalten:

- *Nr.:* Gibt den Prüfpunkt der WCAG1.0 an
- *Evaluierungskriterien:* Kurze Beschreibung des Prüfpunktes
- *Pr.:* Angabe der Priorität des Prüfpunktes; Priorität 1 = A, Priorität 2 = AA
- *Beschreibung:* Fehlerbeschreibung des verletzten Prüfpunktes
- *Bw.*: Bewertung; Ja, Nein, Keine Angabe (KA)
- *Zeilen-Nr:*. betreffende Zeile im Quellcode
- *Mod.:* Bewertungsmodus; automatisch oder manuell

6 Ergebnisse der Evaluierung

In diesem Kapitel werden die Ergebnisse der Evaluierung präsentiert. Neben einer detaillierten Aufstellung der Ergebnisse, welche mit Hilfe der verschiedenen Prüfprogramme ermittelt wurden, findet auch eine Interpretation dieser Ergebnisse statt. Aus diesen interpretierten Ergebnissen wird dann geschlossen, wie gut die jeweiligen Entwicklungstechniken für die Erstellung von barrierefreien Web-Anwendungen geeignet sind.

6.1 Zusammenfassung der Ergebnisse

Die Ergebnisse der Evaluierung werden in den folgenden Tabellen zusammengefasst. Es wird dabei jeweils die Anzahl der verletzten Prüfpunkte pro anwendbare Prüfpunkte angegeben. Diese Wertepaare werden für jede evaluierte HTML-Seite eingetragen. Es wird unterschieden zwischen der Anzahl der Verletzungen der Konformitätsstufe A und AA. Seite1 entspricht der HTML-Seite Vorlesungen.html und Seite2 entspricht der HTML-Seite VorlesungenDetail.html. Nach dem Kriterienkatalog aus Abschnitt 5.3 umfasst die Konformitätsstufe A: 9 Prüfpunkte und die Konformitätsstufe AA: 24 Prüfpunkte. Die Ermittlung dieser Ergebnisse wird in Abschnitt 6.3 erklärt.

Web-Seite	Priorität	ASP.Net	JSF	Webratio	Arcstyler
Seite1	A	1/7	3/7	2/9	4/9
Seite1	AA	5/19	8/18	7/20	16/20
Seite2	A	2/7	2/6	2/6	4/8
Seite2	AA	5/17	7/17	6/17	15/18

Tabelle 6.1 Anzahl der verletzten Prüfpunkte pro anwendbare Prüfpunkte

Eine andere Darstellung der Ergebnisse ist in Abbildung 6.1 dargestellt. Dabei wird die APPF *Accessibility Programing Progess Formula* aus Abschnitt 3.2 verwendet. Die genaue Berechnung wird in Abschnitt 6.3 erklärt. Der Prozentwert gibt an, wie groß der Anteil der erfüllten Prüfpunkte in Bezug auf die Gesamtzahl aller anwendba-

ren Prüfpunkte ist. Somit kann der Fortschritt der Barrierefreiheit einer evaluierten Web-Seite, relativ zu den anderen evaluierten Web-Seiten bemessen werden.

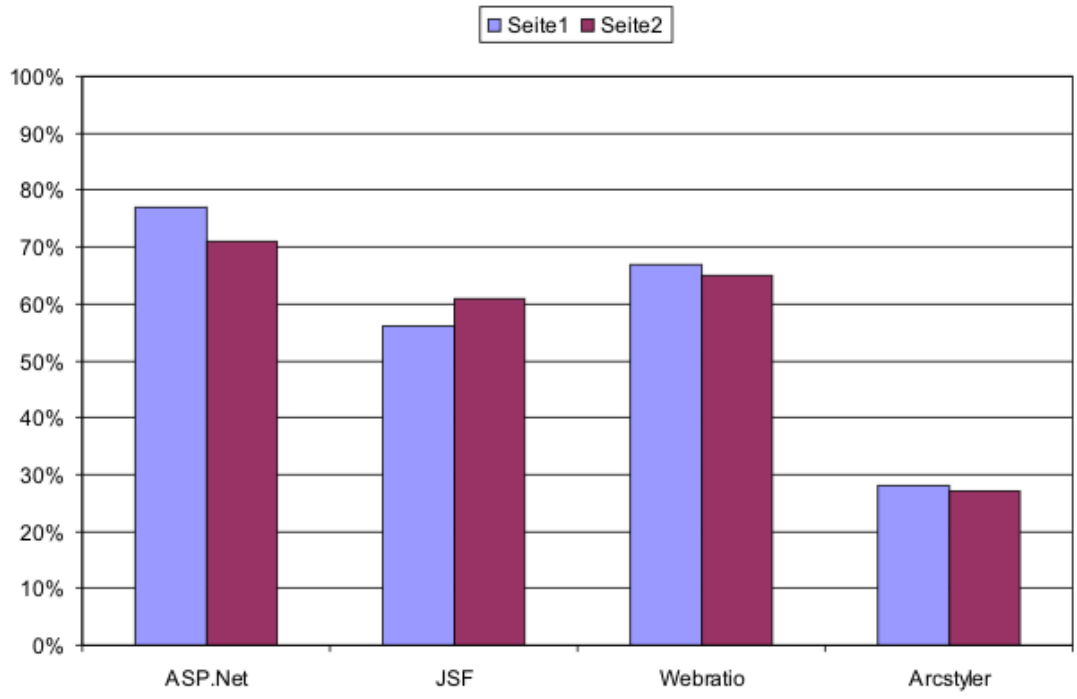

Abbildung 6.1 Fortschritt der Barrierefreiheit, berechnet mit der APPF

Bei dieser Evaluierung hat ASP.Net am besten abgeschnitten, JSF und Webratio erreichen ein fast ähnliches Ergebnis, Arcstyler erreichte den kleinsten Fortschrittswert.

6.2 Detaillierte Auflistung der Ergebnisse

Die Ergebnisse aus den verschiedenen Evaluierungsschritten wurden zusammengefasst und in Excel-Tabellen dokumentiert. Eine Auflistung der aggregierten Evaluierungsergebnisse aller evaluierten HTML-Seiten ist in Anhang C ersichtlich. Pro Web-Anwendung wurden jeweils zwei der generierten HTML-Seiten evaluiert, es sind dies die HTML-Seiten: Vorlesungen.html und VorlesungenDetail.html. Die Tabellen sind nach den verwendeten Entwicklungstechniken: ASP.Net, JSF, Webratio und Arcstyler geordnet. Die originalen Excel-Tabellen sowie die Rohberichte der Prüfprogramme, befinden sich auf der CDROM und sind in Anhang A referenziert.

6.3 Interpretation der Ergebnisse

Die Ergebnisse aus der Evaluierung werden nun dahingehend interpretiert, wie einfach bzw. schwer, die verletzten Prüfpunkte repariert werden können kann. Da es bei den WCAG nur eine Bewertung mit Ja (erfüllt) bzw. Nein (nicht erfüllt) gibt, ist es mit diesen Rohdaten schwierig eine Beurteilung der Entwicklungstechnologien vorzunehmen. Daher werden die Prüfpunkte die mit Nein beurteilt wurden, noch einmal bezüglich des Behebungsaufwandes hinterfragt.

6.3.1 Was kann einfach repariert werden?

Für jeden verletzten Prüfpunkt wurde eruiert, ob zur Behebung des Fehlers nur kleinere Anpassungen notwendig sind oder ob es sich dabei um ein aufwendigeres Problem handelt, dass nur unter größerem Zeitaufwand, bzw. gar nicht gelöst werden kann. Das Ergebnis pro evaluierte Seite und pro Entwicklungstechnologie, ist in den folgenden zwei Tabellen zusammengefasst. Alle Verletzungen der Prüfpunkte die mit (Nein*) bewertet wurden, können durch einfache Anpassungen behoben werden.

Nr.	Evaluierungskriterien	Pr.	ASP.Net	JSF	Webratio	Arcstyler
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Nein*	Nein*	Nein*	Nein
2.1	Informationserhalt bei Fehlen von Farbe	1	Ja	Nein	Ja	Ja
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Nein*	Nein	Nein	Nein
6.1	Lesbarkeit ohne Stylesheets	1	Ja	Nein*	Ja	Ja
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	KA	KA	KA	KA
7.1	Vermeidung von Bildschirmflackern	1	Ja	Ja	Ja	Ja
2.2	Guter Kontrast bei Farben	2	Nein*	Nein*	Nein*	Ja
3.1	Markup statt Bilder	2	Ja	Ja	Ja	Nein
3.2	Einhaltung formaler Grammatiken	2	Nein	Nein	Nein	Nein
3.3	Verwendung von Stylesheets	2	Ja	Nein	Ja	Nein
3.4	Verwendung von relativen Einheiten	2	Nein*	Nein*	Ja	Nein
3.5	Korrekte Darstellung der Dokumentenstruktur	2	Ja	Nein*	Nein*	Nein
3.6	Korrektes Markup für Listen	2	Ja	Nein	Ja	KA

Fortsetzung der Gesamtergebnisse der Datei: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	ASP.Net	JSF	Webratio	Arcstyler
	Allgemein					
3.7	Korrektes Markup für Zitate	2	KA	KA	KA	KA
6.5	Zugänglichkeit von dynamischen Inhalten	2	KA	KA	KA	Nein
7.2	Vermeidung von blinkenden Inhalten	2	Ja	Ja	Ja	Ja
7.4	Vermeidung von autom. Refresh der Seite	2	Ja	Ja	Ja	Ja
11.1	Verwendung von W3C-Technologien	2	Nein	Nein	Nein	Nein
11.2	Vermeidung von überholten W3C-Features	2	Ja	Nein	Nein	Nein
12.3	Unterteilung von Informationsblöcken	2	Ja	Nein	Ja	Nein
13.1	Kennzeichnung von Links	2	Nein	Ja	Nein	Nein
13.2	Verwendung von Metadaten	2	Nein	Nein	Nein	Nein
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Nein	Nein	Nein	Nein
13.4	Konsistente Nutzung von Navigationsmechanismen	2	Ja	Ja	Ja	Nein
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Ja	Ja	Ja	Ja
5.3	Tabellen nicht für Layout-Zwecke verwenden	2	Ja	Ja	Ja	Nein
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Nein	Nein	Nein	Nein
6.4	Eingabegeräte unabhängige Event-Handler definieren	2	KA	KA	Ja	Nein
8.1	Für assistive Technologien zugänglich machen	2	KA	KA	KA	KA
	Frames					
12.1	Betiteln der Frames	1	KA	KA	KA	Nein
12.2	Genauere Beschreibung der Frames	2	KA	KA	KA	Nein
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2	Ja	KA	Ja	KA
12.4	Explizite Zuordnung von Beschriftungen	2	Nein*	Ja	Nein	Ja

Tabelle 6.2 Gesamtergebnisse der Datei: Vorlesungen.html

Gesamtergebnisse der Datei VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	ASP.net	JSF	Webratio	Arcstyler
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Ja	KA	KA	Nein
2.1	Informationserhalt bei fehlen von Farbe	1	Ja	Ja	Ja	Ja
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Nein	Nein	Nein	Nein
6.1	Lesbarkeit ohne Stylesheets	1	Ja	Nein*	Ja	Ja
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	KA	KA	KA	KA
7.1	Vermeidung von Bildschirm-flackern	1	Ja	Ja	Ja	Ja
2.2	Guter Kontrast bei Farben	2	Nein*	Nein*	Nein*	Ja
3.1	Markup statt Bilder	2	Ja	Ja	Ja	Nein
3.2	Einhaltung formaler Gramma-tiken	2	Nein	Nein	Nein	Nein
3.3	Verwendung von Stylesheets	2	Ja	Nein	Ja	Nein
3.4	Verwendung von relativen Einheiten	2	Nein*	Nein*	Ja	Nein
3.5	Korrekte Darstellung der Do-kumentenstruktur	2	Ja	Nein*	Nein*	Nein
3.6	Korrektes Markup für Listen	2	Ja	Nein	Ja	KA
3.7	Korrektes Markup für Zitate	2	KA	KA	KA	KA
6.5	Zugänglichkeit von dynami-schen Inhalten	2	KA	KA	KA	Nein
7.2	Vermeidung von blinkenden Inhalten	2	Ja	Ja	Ja	Ja
7.4	Vermeidung von autom. Refresh der Seite	2	Ja	Ja	Ja	Ja
11.1	Verwendung von W3C-Technologien	2	Nein	Nein	Nein	Nein
11.2	Vermeidung von überholten W3C-Features	2	Ja	Nein	Nein	Nein
12.3	Unterteilung von Informati-onsblöcken	2	Ja	Ja	Ja	Nein
13.1	Kennzeichnung von Links	2	Nein	Ja	Nein	Nein
13.2	Verwendung von Metadaten	2	Nein	Nein	Nein	Nein
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Nein	Nein	Nein	Nein

Fortsetzung der Gesamtergebnisse der Datei: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	ASP.net	JSF	Webratio	Arcstyler
	Allgemein					
13.4	Konsistente Nutzung von Navigationsmechanismen	2	Ja	Ja	Ja	Nein
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Ja	Ja	Ja	Ja
5.3	Tabellen nicht für Layout-Zwecke verwenden	2	Ja	Ja	Ja	Nein
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Nein	Nein	Nein	Nein
6.4	Eingabegeräte unabhängige Event-Handler definieren	2	KA	KA	KA	KA
8.1	Für assistive Technologien zugänglich machen	2	KA	KA	KA	KA
	Frames					
12.1	Betiteln der Frames	1	KA	KA	KA	Nein
12.2	Genauere Beschreibung der Frames	2	KA	KA	KA	Nein
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2	KA	KA	KA	KA
12.4	Explizite Zuordnung von Beschriftungen	2	KA	KA	KA	KA

Tabelle 6.3 Gesamtergebnisse der Datei: VorlesungenDetail.html

6.3.2 Welche Verletzungen wurden automatisch erkannt?

Wenn ein Prüfpunkt automatisch erfasst werden konnte, so wurde dies in den Tabellen mit den Detailergebnissen, Tabelle C.1 bis Tabelle C.8 vermerkt. Wie zu erwarten war, konnten nicht alle Prüfpunkte automatisch erfasst werden. Eine Übersicht über die automatisch erfassten Prüfpunkte der Datei Vorlesungen.html ist in Abbildung 6.2 ersichtlich. Alle anderen Prüfpunkte wurden entweder teilautomatisch bzw. manuell erfasst. Bei der Datei VorlesungenDetail.html ist das Ergebnis ähnlich, da die Inhalte dieser HTML-Seite ein Subset der Seite Vorlesungen.html darstellen. Da nicht alle Prüfpunkte automatisch erfasst werden konnten, sind die rein automatisch erstellten Evaluierungen kein erschöpfender Indikator für die Barrierefreiheit. Es muss daher zusätzlicher Aufwand in manuelle Evaluierungen investiert werden, um ein vollständigeres Ergebnis zu erhalten. Die aufschlussreichsten Ergebnisse, liefern Evaluierungen bei denen auch Menschen mit besonderen Bedürfnissen mitwirken. Diese sind aber

auch sehr aufwendig in der Vorbereitung und zeitintensiv in der Durchführung. Sie erfordern auch Expertenwissen und Erfahrung in diesem Fachbereich.

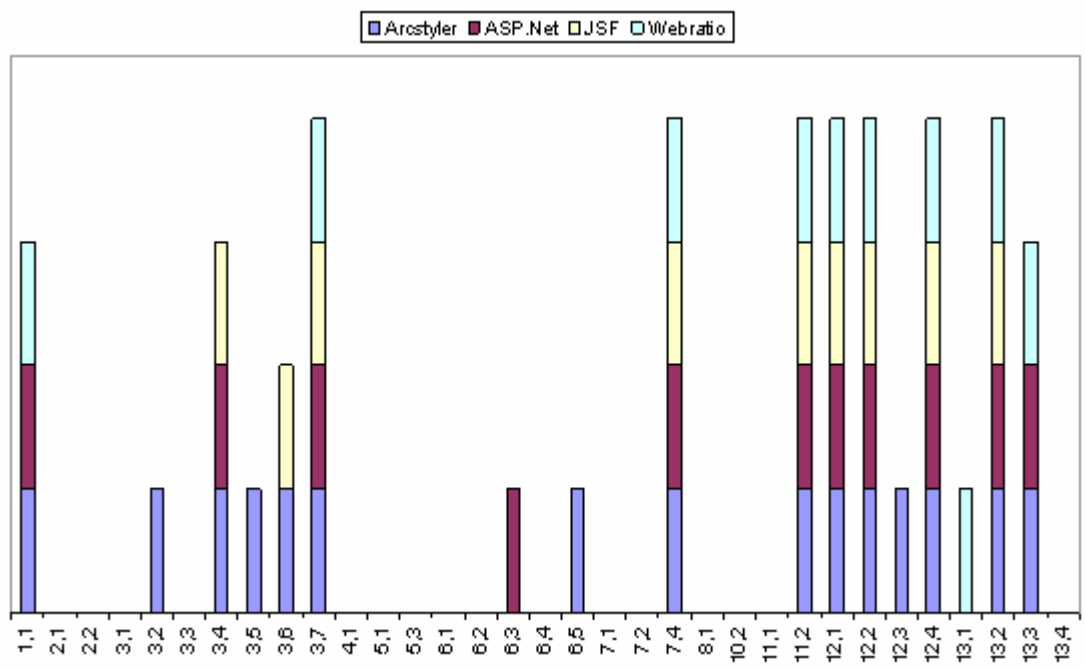

Abbildung 6.2 Datei Vorlesungen.html, automatisch erfasste Prüfpunkte

6.3.3 Auswertung der Evaluierungsergebnisse

Die Auswertung der Rohdaten, ohne Interpretation der verletzten Prüfpunkte, ist in Tabelle 6.4 dargestellt. Dabei steht ihm Zähler die Anzahl der mit Nein und Nein* bewerteten Prüfpunkte. Im Nenner steht die Gesamtanzahl der Prüfpunkte, der jeweiligen Prioritätsstufe, abzüglich der nicht anwendbaren (KA) Prüfpunkte in dieser Stufe. Seite1 ist dabei immer die generierte HTML-Seite: Vorlesungen.html und Seite2 ist die generierte HTML-Seite: VorlesungenDetail.html.

Web-Seite	Priorität	ASP.Net	JSF	Webratio	Arcstyler
Seite1	A	3/7	5/7	3/9	4/9
Seite1	AA	8/19	11/18	10/20	16/20
Seite2	A	2/7	2/6	2/6	4/8
Seite2	AA	7/17	10/17	8/17	15/18

Tabelle 6.4 Anzahl der WCAG-Verletzungen (ohne Interpretation)

Werden nun die mit Nein* gekennzeichneten Prüfpunkte nicht mehr als Verletzung gezählt, da sie ja einfach repariert werden können, ergibt sich folgendes Endergebnis in Tabelle 6.5:

Web-Seite	Priorität	ASP.Net	JSF	Webratio	Arcstyler
Seite1	A	1/7	3/7	2/9	4/9
Seite1	AA	5/19	8/18	7/20	16/20
Seite2	A	2/7	2/6	2/6	4/8
Seite2	AA	5/17	7/17	6/17	15/18

Tabelle 6.5 Anzahl der WCAG-Verletzungen (mit Interpretation)

Eine Darstellung der Endergebnisse für Seite1, ist in Form eines Balkendiagramms, in Abbildung 6.3 ersichtlich. Dabei wird nur die Prioritätsstufe A dargestellt.

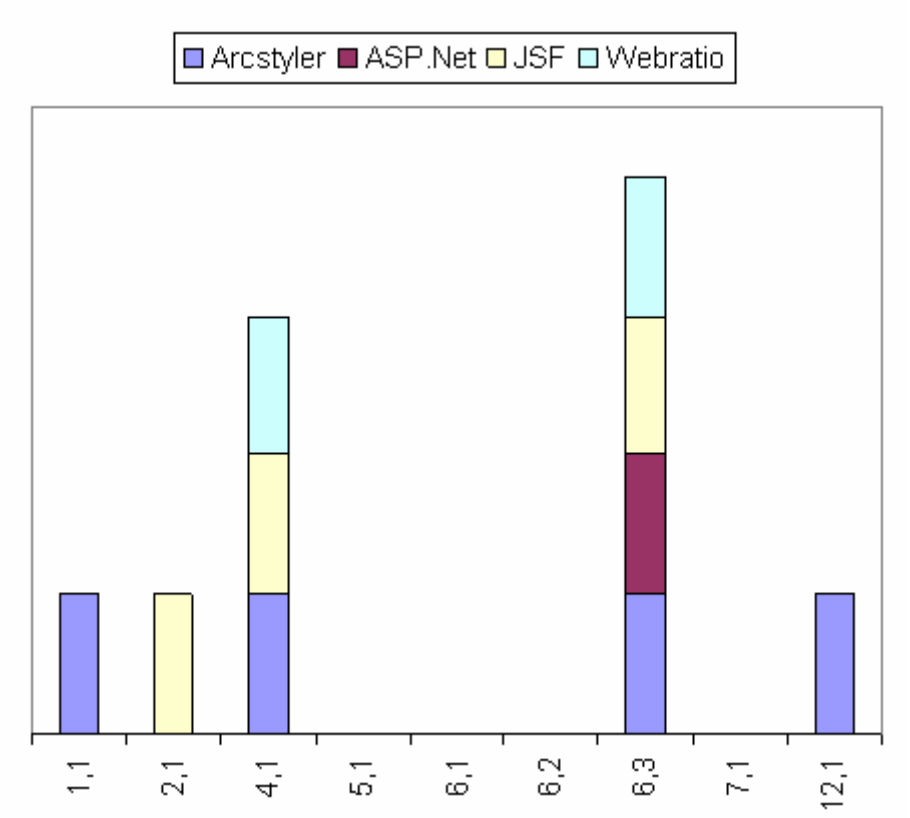

Abbildung 6.3 Priorität A, Verletzungen der Prüfpunkte bei Seite1

Die Verletzungen der Prüfpunkte von Prioritätsstufe AA werden für Seite1 in Abbildung 6.4 dargestellt.

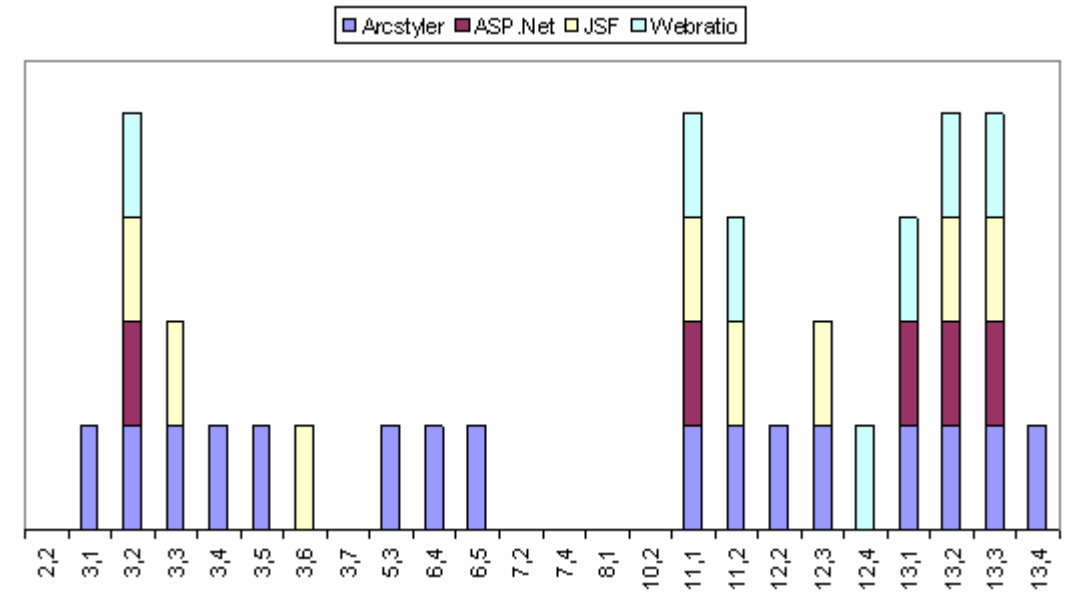

Abbildung 6.4 Priorität AA, Verletzungen der Prüfpunkte bei Seite1

Die Ergebnisse der Seite2 werden nicht grafisch dargestellt, da sie der Seite1 sehr ähnlich sind.

Eine andere und kompaktere Darstellung der Ergebnisse, ergibt sich unter der Verwendung der *Accessibility Programming Progress Formula (*APPF), aus Abschnitt 3.2 Diese Formel berechnet den Fortschritt der Barrierefreiheit einer Web-Seite als Prozentwert:

$P = [(a + b) / (T – c)] *100\%$

P… Fortschritt der Barrierefreiheit in %,

T… Anzahl aller Prüfpunkte in der Checkliste = 33; 9 für A, 24 für AA

a…Anzahl der eingehaltenen Prüfpunkte (Bewertung: Ja)

b…Anzahl der leicht zu reparierenden Prüfpunkte (Bewertung: Nein*)

c…Anzahl der nicht anwendbaren Prüfpunkte (Bewertung: KA)

Diese Formel wird nun auf die Ergebnisse aus Tabelle 6.2 und Tabelle 6.3 angewendet, dabei werden aber die Prioritätsstufen nicht berücksichtigt, sondern jede Verletzung wird als gleichwertig erachtet. Es ergibt sich dann folgendes Endergebnis, welches in Tabelle 6.6 dargestellt ist. Wenn mehrere Seiten verwendet werden, kann auch der Durchschnitt bzw. Median der Prozentwerte gebildet werden.

Web-Seite	ASP.Net	JSF	Webratio	Arcstyler
Seite1	77%	56%	67%	28%
Seite2	71%	61%	65%	27%
Durchschnitt	74%	59%	66%	28%

Tabelle 6.6 Endergebnis, Fortschritt der Barrierefreiheit in Prozent

Bei dieser Evaluierung hat ASP.Net am besten abgeschnitten. JSF und Webratio haben etwa gleich abgeschnitten. Arcstyler hat am meisten verletzte Prüfpunkte.

6.4 Stärken und Schwächen der Entwicklungstechniken

Abschließend werden die verwendeten Softwareentwicklungstechniken gegenübergestellt und ihre Stärken und Schwächen, in Bezug auf die Barrierefreiheit des generierten Codes beleuchtet. Es gibt Prüfpunkte, die bei allen vier Web-Anwendungen auf fast jeder evaluierten Seite verletzt wurden. Dies sind die Prüfpunkte 3.2, 4.1, 6.3, 11.1, 13.2 und 13.3. Wobei 13.2 (Verwendung von Metadaten) und 13.3 (Site-Map) schon in der Referenzanwendung fehlen und somit gar nicht abgebildet wurden. Bei 3.2 (Einhaltung formaler Grammatiken) haben alle Entwicklungstechniken Schwächen, da sie noch kaum moderne Dokumenttypen (z.B. XHTML 1.1) generieren. Oft war es nicht feststellbar, welcher Dokumenttyp wirklich generiert worden ist. Bei Prüfpunkt 4.1 (Kennzeichnung der Sprache bei Änderung), wird die Kennzeichnung von Änderungen in der Sprache, bei der Codegenerierung nicht berücksichtigt. Der Prüfpunkt 6.3 (Informationserhalt bei Deaktivierung von Scripts), wird auch nie eingehalten. Es werden auch keine „Noscript-Elemente“ generiert. Es ist zur Entwicklungszeit auch nicht nachvollziehbar, an welcher Stelle der Einsatz von JavaScript zu erwarten ist, bzw. auch nicht beeinflussbar. Prüfpunkt 11.1 (Verwendung von W3C-Technologien) wird deshalb nicht eingehalten, weil oft JavaScript zur Anwendung kommt und dies keine W3C-Technologie ist.

6.4.1 ASP.Net

ASP.Net hat in der Evaluierung am besten abgeschnitten. Dies ist darauf zurückzuführen, dass Microsoft bei der Entwicklung dieser Technologie die Barrierefreiheit des generierten HTML-Codes als einen wichtigen Aspekt berücksichtigt hat. Bei ASP.Net kann der HTML-Code zur Entwicklungszeit editiert werden. HTML-Code und Komponenten-Designer synchronisieren sich ständig. Die Komponentenbiblio-

thek erlaubt die Anpassung vieler Eigenschaften, welche für die Barrierefreiheit relevant sind. Wenn etwas über die Komponentenbibliothek nicht einstellbar ist, so können zur Entwicklungszeit noch zusätzliche Elemente im Quellcode eingebaut werden. Der generierte Code sieht fast gleich aus wie der editierte Code zur Entwicklungszeit. Nur die Komponenten sind nun als HTML-Code (mit oder ohne JavaScript) dargestellt. Negativ aufgefallen ist die Verwendung von Javascript für die Links in der Datentabelle. Es war auch nicht möglich einen Link-Text generieren zu lassen, der das Ziel explizit ausweist. Somit ist der Link-Text für die Auswahl der Vorlesungen immer der gleiche. Abschließend kann gesagt werden, dass mit ASP.Net für die Microsoft-Plattform eine Entwicklungstechnologie zur Verfügung steht, mit der die Entwicklung von barrierefreien Web-Anwendungen gut möglich ist.

6.4.2 JSF

JSF hat in dieser Evaluierung mittelmäßig abgeschnitten. Die Entwicklung der JSF-Web-Anwendung gestaltete sich nicht so einfach wie bei ASP.Net. Der Einfluss auf den generierten Code zur Entwicklungszeit ist praktisch nicht gegeben. Die vorhandenen Komponenten erlaubten auch nicht so viele Anpassungsmöglichkeiten wie bei ASP.Net. Auch sieht der generierte Code etwas merkwürdig aus, da fast alle Komponenten mit Hilfe von JavaScript abgebildet werden. Somit scheitert die Betrachtung in einem Textbrowser ohne JavaScript. JSF war die einzige Technologie, welche eine sprechende Link-Bezeichnung, für die Links in der Datentabelle ermöglichte. Die Integration von vordefinierten CSS-Stylesheets gestaltete sich schwierig. Da für jede Komponente eine bestimmte „Style-Class“ explizit zugeordnet werden musste. Außerdem gibt es noch ein Haupt-Stylesheet der Komponentenbibliothek, welches höhere Priorität hat als das angegebene Benutzer-Stylesheet. Die Einhaltung eines vordefinierten Layout’s wird dadurch erschwert. Oft wird man gezwungen „Tag-Styles“ (Definition der Stile im HTML-Code) zu verwenden. Obwohl alle Stile im Stylesheet definiert werden sollten. Ein Problem stellt auch die Strukturierung der Inhalte dar. Es war nicht möglich die Listen- und Absatz-Elemente in gleicher Art und Weise zu integrieren, wie dies bei der Referenzanwendung der Fall ist. Alle genannten Punkte gelten natürlich nur für die verwendete Referenz-Implementierung von SUN und die Woodstock-Komponentenbibliothek. Da es von JSF mehrere Implementierungen, bzw. Komponentenbibliotheken gibt, wäre es interessant zu wissen wie sich diese verhalten. In der neuen Version von Netbeans (Version 6.5), wird die Woodstock-Bibliothek durch die Komponentenbibliothek des derzeitigen JSF-Marktführers ICEFaces ersetzt.

6.4.3 Arcstyler

Arcstyler fällt etwas aus der Reihe, der generierte Code dieses modellbasierten Code-Generators hat bei der Evaluierung am schlechtesten abgeschnitten. Arcstyler ist aber auch nicht auf die Entwicklung von barrierefreien Web-Anwendungen spezialisiert. Das „WebAccessor-Cartridge“, welches die Entwicklung von Web-Anwendungen ermöglichen soll, stellt nur wenige Komponenten zur Verfügung. Zum Beispiel gibt es keine Links, sondern nur Buttons. Die Abbildung der Referenzanwendung war nur in einem sehr beschränkten Umfang möglich. Die Kernfunktionalität konnte zwar abgebildet werden, aber das vordefinierte Layout nicht. Die Integration von vordefinierten Stylesheets wird nicht unterstützt. Es war auch nicht möglich die Struktur der Anwendung abzubilden, da für die Strukturierung der Seite noch Frames verwendet werden müssen. Dies bringt so manche Folgeprobleme mit sich. Der generierte Code enthält wenig HTML-Code. Es wird eine spezielle API: „DynAPI“ für JavaScript verwendet. Somit funktioniert die Anwendung bei abgeschaltetem JavaScript nicht. Bei der Evaluierung sind zahlreiche Verletzungen der Kriterien aufgetreten. Sie sind im Detail in Tabelle C.7 ersichtlich.

6.4.4 Webratio

Webratio hat bei der Evaluierung mittelmäßig bis gut abgeschnitten. Die verwendete Version von Webratio (5.0) hat einige Vorteile gegenüber den alten Versionen. Diese Version von Webratio läuft in einer „Eclipse-Umgebung“ und unterstützt vordefinierte Layout’s. Dies wird mit speziellen Layoutprojekten erreicht, die parallel zum Modellierungsprojekt gepflegt werden können. Die Referenzanwendung konnte ohne große Probleme abgebildet werden. Allerdings stellt sich die Frage, ob vielleicht bei komplexeren Anwendungen, Probleme bei der Zuordnung von Modell- zu Layout-Elementen auftreten. Das vordefinierte Stylesheet konnte ohne Probleme integriert werden. Der Vorteil der Layoutprojekte ist, dass während der Entwicklungszeit auch HTML-Code editiert und mit dem Modell verknüpft werden kann. Diese Möglichkeit fehlt bei Arcstyler. Allerdings ist die Einflussnahme auf den Code längst nicht so komfortabel wie bei ASP.Net. Der Vorgang entspricht eher dem „Try- and Error-Prinzip“, da im Vorhinein nicht klar ersichtlich ist, was der Generator aus der parametrisierten Layoutvorlage einer Komponente erzeugen wird. Somit ist die Entwicklung von barrierefreien Web-Anwendungen mit Webratio, noch eine zeit- und lern-intensive Angelegenheit.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Heutige Ansätze zur Entwicklung von Web-Anwendungen, versprechen ähnliche Möglichkeiten, wie sie bei der Entwicklung von Desktop-Anwendungen bereits üblich sind. Es kommen komponentenbasierte Frameworks und auf MDA basierende Code-Generatoren zum Einsatz. Die Qualität des generierten Codes genügt aber nicht immer dem Qualitätsaspekt der Barrierefreiheit. Menschen mit besonderen Bedürfnissen, sind bei der Nutzung von Web-Inhalten oft auf assistierende Technologien angewiesen. Generierter Code kann mit unbeabsichtigten Barrieren behaftet sein, die eine Nutzung auf diese Art erschweren. Gesetze und Richtlinien zur Barrierefreiheit im Web, sollen Rahmenbedingungen schaffen um die Qualität von Web-Inhalten bzw. Web-Anwendungen zu verbessern. Die Gesetze nehmen jene in die Pflicht, die Web-Anwendungen erstellen, bzw. Web-Auftritte betreiben. Die Entscheidungsfindung für die richtige Entwicklungstechnologie, wird dadurch aufwändiger. Da auf die Angaben der Hersteller von Entwicklungstechnologien, nicht immer vertraut werden kann, muss vor der Entscheidungsfindung eine Evaluierung der Technologien erfolgen. Das Ziel dabei ist, den generierten Code hinsichtlich Barrierefreiheit zu überprüfen.

In dieser Diplomarbeit wurde eine derartige Evaluierung durchgeführt. Die Evaluierung der komponentenbasierten Frameworks: *ASP.Net* und *JSF*, sowie der modellbasierten Code-Generatoren: *Arcstyler* und *Webratio* hat ergeben, dass die Generierung von barrierefreiem HTML-Code noch nicht selbstverständlich ist. Die Evaluierungskriterien orientierten sich an den Prüfpunkten der WCAG1.0, Stufe A und AA. Mit den vier genannten Entwicklungstechniken wurde eine barrierefreie Referenzanwendung (statischer Prototyp), jeweils als dynamische Web-Anwendung abgebildet. Eine Auswahl von daraus generierten Web-Seiten, wurde mit Hilfe von verschiedenen Prüfwerkzeugen evaluiert. Aus den Ergebnissen, wurde der Fortschritt der Barrierefreiheit mit Hilfe der *Accessibility Programming Progess Formula* (APPF) berechnet. Dabei erreichten die evaluierten HTML-Seiten, der implementierten Web-Anwendungen, folgende Bewertungen: ASP.Net 74%, Webratio 66%, JSF 59% und Arcstyler 28%. Die Ergebnisse der Evaluierung, stimmen mit den Erfahrungen, die während der Implementierung gemacht wurden, durchaus überein. Dies ist hauptsäch-

lich darauf zurückzuführen, dass ASP.Net und Webratio es erlauben, den endgültigen HTML-Code zur Entwicklungszeit zu beeinflussen. Diese Möglichkeiten sind aber bei ASP.Net viel stärker ausgeprägt als bei Webratio. Generell hat sich aber gezeigt, dass es mit den komponentenbasierten Frameworks einfacher sein wird, komplexe Web-Anwendungen barrierefrei zu erstellen. Da es prinzipiell möglich ist, selbst Komponenten zu entwickeln bzw. zu verändern. Die Code-Generatoren hingegen, arbeiten auf einer höheren Abstraktionsebene. Dadurch ist es viel schwieriger eine Architektur zur Verfügung zu stellen, die auch eine komfortable Einflussmöglichkeit auf die Komponenten bzw. den endgültigen Code bietet. Dies hat sich im Besonderen bei Arcstyler gezeigt, da hier die Modellierung mit UML erfolgte und es keine Möglichkeit gab, den endgültigen HTML-Code zur Entwicklungszeit zu beeinflussen. Die Barrierefreiheit im Web, wird aber auf der Code-Ebene definiert. Daraus leiten sich für die Hersteller von Entwicklungstechnologien für Web-Anwendungen zwei Schlussfolgerungen ab. Entweder wird die Architektur so gewählt, dass es zur Entwicklungszeit möglich ist, den endgültigen HTML-Code weitestgehend zu beeinflussen, z.B. mit geschützten Code-Bereichen für die Komponenten. Oder es werden Architekturen gewählt, die nur einen geringen Funktionsumfang zulassen und nur Komponenten enthalten, die speziell hinsichtlich Barrierefreiheit entwickelt und getestet worden sind. Da sich aber gerade der Bereich der Web-Technologien besonders schnell verändert, ist die Wahl des ersten Architekturtyps, über längere Zeit gesehen, wahrscheinlich der flexiblere Ansatz.

7.2 Ausblick

Die derzeitige Situation der Barrierefreiheit von Web-Auftritten im öffentlichen- und im privaten Sektor ist noch alles andere als zufriedenstellend. Eine Studie in Auftrag der EU-Kommission ergab in 2008 ein ernüchterndes Ergebnis. Laut der Studie "Measuring progress of eAccessibility in Europe" (MeAC), erreichten 60% der behördlichen Web-Auftritte und 95% der Web-Auftritte des privaten Sektors, nicht einmal den Level A der WCAG1.0. Im März 2009 hat der Europäische Rat in den „Council Conclusions on accessible information society“ die Ablöse der WCAG1.0 durch die WCAG2.0 vorgeschlagen. Es ist aber unwahrscheinlich, dass sich die reale Situation ändert, solange keine verbindliche Gesetzgebung für den privaten Sektor etabliert wird. Manche Vertreter des privaten Sektors, sehen den Aspekt der Barrierefreiheit als zusätzlichen Kostenfaktor. Zu dem besteht die Gefahr, dass sich die WAI mit der Entwicklung der WCAG2.0 zu weit in eine theoretische Idealvorstellung vorgewagt hat, die aber im Moment nur mit sehr hohem Aufwand zu erfüllen ist. Die WCAG2.0 sind technologieneutral und die Interpretation der Richtlinien lässt mehr

Spielräume zu als bei WCAG1.0. Die effiziente Erstellung von barrierefreien Web-Auftritten, ist abhängig von der Schulung des Entwicklungspersonals und von den zur Verfügung stehenden Entwicklungstechnologien. Die Herausforderung, die auf Gesetzgeber und Betreiber von Web-Auftritten gleichermaßen zu kommt, ist die Bildung eines verbindlichen Konsenses über Mindeststandards. Neben den gesetzlichen Maßnahmen müssten auch Zertifizierungen vorgeschrieben werden. Hersteller von Komponenten-Frameworks und Code-Generatoren müssen einen Anreiz haben, ihre Architekturen so auszulegen, dass die Generierung von barrierefreiem Code leichter möglich ist. Dies kann aber nur der Markt erzwingen. Bei einer zu starken Regulierung durch Gesetze besteht die Gefahr, dass sich die Barrierefreiheit nur noch auf alternative Web-Inhalte beschränkt, und somit eine Zweiklassengesellschaft im Web entsteht. Die Möglichkeiten die durch Technologien wie z.B. AJAX geschaffen werden, bringen zusätzliche Barrieren mit sich. Die WAI hat darauf mit dem Entwurf der WAI-ARIA reagiert. Die Frage ist jedoch, in wie fern Herstellerfirmen von Entwicklungstechnologien bereit sind, derartige Hilfsmittel in ihre Werkzeuge zu integrieren. Die Entwicklung von barrierefreien Web-Anwendungen könnte sich viel einfacher gestalten, wenn dies von den Entwicklungstechniken aktiv unterstützt würde.

“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”

-- Tim Berners-Lee, W3C Direktor und Erfinder des World Wide Web --

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Barr06] Barrierefrei informieren und kommunizieren, *BITV*, http://www.bik-online.info/info/gesetze/bitv/index.php, 2006. Zugriff am: 12.10.2008.

[Bund02] Bundesministerium der Justiz, *Gesetz zur Gleichstellung Behinderter Menschen*, http://bundesrecht.juris.de/bgg/index.html, 2002. Zugriff am: 12.10.2008.

[Bund08] Bundeskanzleramt RIS, *E-Government Gesetz*, http://www.ris2.bka.gv.at/GeltendeFassung.wxe?QueryID=Bundesnormen&Gesetzesnummer=20003230, 2008. Zugriff am: 09.10.2008.

[Chis05] Chisholm, W., A., Henry, S., L. Interdependent components of web accessibility. In *Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, ACM, 2005.

[Coop07] Cooper, M. Accessibility of emerging rich web technologies: web 2.0 and the semantic web. In *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility, SESSION: Understanding accessibility*, ACM, 2007.

[Digi08] Digitales Österreich, *Web Accessibility*, http://www.bka.gv.at/site/5566/default.aspx, Zugriff am: 09.10.2008.

[Euro09] Europa-Information Society, *Assessment of the Status of eAccessibility in Europe*, http://ec.europa.eu/information_society/activities/einclusion/library/studies/meac_study/index_en.htm, Zugriff am: 10.05.2009.

[Gund06] Gunderson, J., Rangin, H., B., AND Hoyt, N. Functional web accessibility techniques and tools from the university of Illinois. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, ACM, 2006.

[Hell06] Hellbusch, J., E., Mayer, T. *Barrierefreies Webdesign*, KnowWare Aps, 2006.

[Henr05] Henry, S., L. *Essential Components of Web Accessibility*, http://www.w3.org/WAI/intro/components.php, 2005. Zugriff am: 10.4.2009.

[Holm07] Holman, J., Lazar, J., Feng, J., H., AND Darcy, J. Developing usable CAPTCHAs for blind users. In *Proceedings of the Ninth International ACM SIGACCESS Conference on Computers and Accessibility*, ACM, 2007.

[Kann06] Kannengiesser, I., Prickartz, B. *Web-Ergonomie & Barrierefreiheit im Internet*, E. Ferger Verlag, 2006.

[Kirc02] Kirchner, M. Evaluation, repair, and transformation of Web pages for Web content accessibility. Review of some available tools. In *Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02)*, IEEE, 2002.

[Komm05] Kommission der Europäischen Gemeinschaft, *Mitteilung der Kommission an den Rat, das Europäische Parlament […]: eAccessibility*, http://www.supportam.org/waec/docs/mod01/COM(2005)_425_eAccessibility_de.pdf, 2005. Zugriff am: 06.10.2008.

[Lawc05] Law, C., Jacko, J., AND Edwards, P. Programmer-focused website accessibility evaluations. In *Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility*, ACM, 2005.

[Mies02] Miesenberger, K., Klaus, J., AND Zagler, W. Computers Helping People with Special Needs. In *Proceedings of the 8th International Conference, ICCHP 2002*, Springer Verlag, 2002.

[Radt06] Radtke, A., Charlier, M. *Barrierefreies Webdesign, Attraktive Websites zugänglich gestalten*, Addison Wesley, 2006.

[Sect08] Section 508, *Section 508 Standards*, http://www.section508.gov/index.cfm?FuseAction=Content&ID=12, Zugriff am: 20.10.2008.

[Stey08] Steyer, R. *Ajax Frameworks, RIAs mit Dojo & Co*, Addison-Wesley, 2008.

[Thee05] The European Commission, *Before i2010: eEurope initiative*, http://ec.europa.eu/information_society/eeurope/2002/index_en.htm, 2005. Zugriff am: 06.10.2008.

[Thee08] The European Commission, *eAccessibility*, http://ec.europa.eu/information_society/activities/einclusion/policy/accessibility/index_en.htm, Zugriff: am 09.10.2008.

[Thie07] Thiessen, P., Chen, C. Ajax live regions: chat as a case example. In *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A)*, ACM, 2007.

[W3ca08] W3C, *Accessible Rich Internet Applications (WAI-ARIA) Version 1.0*, http://www.w3.org/TR/2008/WD-wai-aria-20080204/, 2008. Zugriff am: 3.4.2009.

[W3cc00] W3C, *Checkliste der Checkpunkte zu den Zugänglichkeitsrichtlinien für Web-Inhalte 1.0*, http://www.w3c.de/Trans/WAI/checkliste.html, 2000. Zugriff am: 10.10.2008.

[W3cc05] W3C, *Conformance Evaluation of Web Sites for Accessibility*, http://www.w3.org/WAI/eval/conformance.html, 2005. Zugriff am: 4.5.20009.

[W3ct00] W3C, *Techniques for Web Content Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-WEBCONTENT-TECHS, 2000. Zugriff am: 17.10.2008.

[W3ct08] W3C, *The WCAG 2.0 Documents*, http://www.w3.org/WAI/intro/wcag20, 2008. Zugriff am: 20.05.2009.

[W3cw99] W3C, *Web Content Accessibility Guidelines 1.0*, http://www.w3.org/TR/WCAG10/, 1999. Zugriff am: 09.10.2008.

[W3cw06] W3C, *Web Accessibility Evaluation Tools: Overview*, http://www.w3.org/WAI/ER/tools/Overview.html, 2006. Zugriff am: 24.4.2009.

[W3cw08a] W3C, *Web Content Accessibility Guidelines 2.0*, http://www.w3.org/TR/WCAG20/, 2008. Zugriff am: 17.10.2008.

[W3cw08b] W3C, *WAI-ARIA Overview*, http://www.w3.org/WAI/intro/aria.php, 2008. Zugriff am: 3.4.2009.

[W3cw08c] W3C, *WAI-ARIA Primer*, http://www.w3.org/TR/2008/WD-wai-aria-primer-20080204/, 2008. Zugriff am: 3.4.2009.

[W3cw08d] W3C, *WAI-ARIA Best Practices*, http://www.w3.org/TR/2008/WD-wai-aria-practices-20080204/, 2008. Zugriff am: 3.4.2009.

[Wabc07] WAB Cluster, *Unified Web Evaluation Methodology version 1.2*, http://www.wabcluster.org/uwem1_2/, 2007. Zugriff am: 10.4.2009.

[Weis04] Weist, D. *Accessibility- Barrierefreies Internet, Hintergründe - Technik - Lösungen für Menschen mit Behinderungen*, VDM Verlag Dr. Müller, 2004.

[Wenz08] Wenz, C. *JavaScript und Ajax, Das umfassende Handbuch*, 8. ed. Galileo Press, 2008.

[Zeld06] Zeldman, J. *Webdesign mit Webstandards, Grenzenlos kompatibel*, 2. ed. Addison-Wesley, 2006.

Anhang A Quellcode und Ergebnisse

1. **Quellcode der Referenzanwendung**

   CDROM\Referenzanwendung

   \Index.html

   \Lectures.html

   \Lecturedetails.html

   \Professors.html

   \Professordetails.html

2. **Entwicklungsprojekte der Web-Anwendungen**

   CDROM\WebAnwendungen

   \ASP.Net

   \Arcstyler

   \JSF

   \Webratio

   \Datenbank

3. **Installationsanleitung**

   CDROM\WebAnwendungen\Installationsanleitung.pdf

4. **Generierte Web-Seiten**

   CDROM\Evaluierung\generierte_HTML_Seiten

5. **Berichte und Ergebnisse der Evaluierung**

   CDROM\Evaluierung

Anhang B W3C® DOCUMENT LICENSE

http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

Public documents on the W3C site are provided by the copyright holders under the following license. By using and/or copying this document, or the W3C document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to copy, and distribute the contents of this document, or the W3C document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the document, or portions thereof, that you use:

1. A link or URL to the original W3C document.
2. The pre-existing copyright notice of the original author, or if it doesn't exist, a notice (hypertext is preferred, but a textual representation is permitted) of the form: "Copyright © [$date-of-document] World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231"
3. If it exists, the STATUS of the W3C document.

When space permits, inclusion of the full text of this NOTICE should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, if additional requirements (documented in the Copyright FAQ) are satisfied, the right to create modifications or derivatives is sometimes granted by the W3C to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

This formulation of W3C's notice and license became active on December 31 2002. This version removes the copyright ownership notice such that this license can be used with materials other than those owned by the W3C, moves information on style sheets, DTDs, and schemas to the Copyright FAQ, reflects that ERCIM is now a host of the W3C, includes references to this specific dated version of the license, and removes the ambiguous grant of "use". See the older formulation for the policy prior to this date. Please see our Copyright FAQ for common questions about using materials from our site, such as the translating or annotating specifications. Other questions about this notice can be directed to site-policy@w3.org.

Joseph Reagle <site-policy@w3.org>

Last revised $Id: copyright-documents-20021231.html,v 1.6 2004/07/06 16:02:49 slesch Exp $

Anhang C Evaluierungsergebnisse

ASP.Net

Evaluierte HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Textfeld ohne ALT-Attribut	Nein	74	A1
2.1	Informationserhalt bei Fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Änderung der Sprache bei Tabellendaten nicht gekennzeichnet; Keine Kennzeichnung der vorherrschenden Sprache	Nein	92	M
6.1	Lesbarkeit ohne Stylesheets	1	Gitter der Datentabelle verschwindet	Ja		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine Nicht-Text Inhalte, die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1		Ja		M
2.2	Guter Kontrast bei Farben	2	Farbkontrast-Verhältnis getestet mit: Juicy Studio Toolbar, 3 Fehler	Nein		M
3.1	Markup statt Bilder	2		Ja		M
3.2	Einhaltung formaler Grammatiken	2	Html-Output entspricht Doctype HTML 4.1 Transitional, 8 Fehler mit W3C-HTML-Validator; CSS Validierung ist OK	Nein		M
3.3	Verwendung von Stylesheets	2		Ja		M
3.4	Verwendung von relativen Einheiten	2	Elemente INPUT und TABLE verwenden absolute Einheiten	Nein	76,84	A2
3.5	Korrekte Darstellung der Dokumentenstruktur	2		Ja		M
3.6	Korrektes Markup für Listen	2		Ja		M
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	siehe 6.3	KA		M
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2	Element http-equiv mit Attribut refresh wird nicht benützt	Ja		A1

Fortsetzung der evaluierten HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
11.1	Verwendung von W3C-Technologien	2	Es wird auch Javascript verwendet	Nein	14,88	M
11.2	Vermeidung von überholten W3C-Features	2		Ja		A1
12.3	Unterteilung von Informationsblöcken	2		Ja		M
13.1	Kennzeichnung von Links	2	Gleicher Link-Text für unterschiedliche Ressourcen	Nein	88,90, 92	M
13.2	Verwendung von Metadaten	2	Meta-Element ist unvollständig, Attribute "description, keywords, language" fehlen	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Site-Map vorhanden	Nein		A1
13.4	Konsistente Nutzung von Navigationsmechanismen	2		Ja		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltentitel sind vorhanden, Tabellentitel fehlt, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2		Ja		M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Links funktionieren nicht ohne javascript, Element NOSCRIPT fehlt	Nein	88,90, 92	A2
6.4	Eingabegeräte unabhängige Event-Handler definieren	2		KA		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1		KA		A1
12.2	Genauere Beschreibung der Frames	2		KA		A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2		Ja		M
12.4	Explizite Zuordnung von Beschriftungen	2	Das Label "Suche" ist dem Textfeld "TextBox1"nicht explizit zugeordnet	Nein	72	A1,A2

Tabelle C.1 ASP.Net, Detailergebnisse der Datei: Vorlesungen.html

Evaluierte HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1		Ja		A1
2.1	Informationserhalt bei Fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Span-Element zeichnet Label als US-en aus obwohl es in deutsch ist	Nein	72	M
6.1	Lesbarkeit ohne Stylesheets	1	Gitter der Datentabelle verschwindet.	Ja		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1		KA		M
7.1	Vermeidung von Bildschirmflackern	1		JA		M
2.2	Guter Kontrast bei Farben	2	Farbkontrast-Verhältnis getestet mit: Juicy Studio Toolbar, 3 Fehler	Nein		M
3.1	Markup statt Bilder	2		Ja		M
3.2	Einhaltung formaler Grammatiken	2	Html-Output entspricht Doctype HTML 4.1 Transitional, 9 Fehler mit W3C-HTML-Validator; CSS Validierung ist OK	Nein		M
3.3	Verwendung von Stylesheets	2		Ja		M
3.4	Verwendung von relativen Einheiten	2	Absolute Einheiten werden verwendet "height: 41px"	Nein	80	A2
3.5	Korrekte Darstellung der Dokumentenstruktur	2		Ja		M
3.6	Korrektes Markup für Listen	2		Ja		M
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	siehe 6.3	KA		M
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2	Element http-equiv mit Attribut refresh wird nicht benützt	Ja		A1

Fortsetzung der evaluierten HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
11.1	Verwendung von W3C-Technologien	2	Es wird auch Javascript verwendet	Nein	17, 100	M
11.2	Vermeidung von überholten W3C-Features	2		Ja		A1
12.3	Unterteilung von Informationsblöcken	2		Ja		M
13.1	Kennzeichnung von Links	2	Link-Text gibt keine Info über das Linkziel	Nein	100	A1
13.2	Verwendung von Metadaten	2	Meta-Element ist unvollständig Attribute "description, keywords, language" fehlen	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Site-Map vorhanden	Nein		A1
13.4	Konsistente Nutzung von Navigationsmechanismen	2		Ja		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltentitel sind vorhanden, Tabellentitel fehlt, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2		Ja		M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Link funktioniert nicht ohne aktiviertem Javascript	Nein	100	A2
6.4	Eingabegeräte unabhängige Event-Handler definieren	2		KA		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1		KA		A1
12.2	Genauere Beschreibung der Frames	2		KA		A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2		KA		A1
12.4	Explizite Zuordnung von Beschriftungen	2		KA		A1

Tabelle C.2 ASP.Net, Detailergebnisse der Datei: VorlesungenDetail.html

Java Server Faces

Evaluierte HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Textäquivalent für Textfeld vorhanden aber mit falschem Text	Nein		M
2.1	Informationserhalt bei Fehlen von Farbe	1	Textfeldrahmen verschwindet bei Graustufenansicht	Nein		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Die Seite enthält keine Lang-Attribute	Nein		M
6.1	Lesbarkeit ohne Stylesheets	1	Überschrift h1 ist nicht richtig dargestellt, Navigations-Links sind unübersichtlich angeordnet, Tabellengitter fehlt	Nein		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine Nicht-Text Inhalte, die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1		Ja		M
2.2	Guter Kontrast bei Farben	2	Farbkontrast-Verhältnis getestet mit: Juicy Studio Farbkontraste, 5 Fehler	Nein		M
3.1	Markup statt Bilder	2		Ja		M
3.2	Einhaltung formaler Grammatiken	2	Html-Output entspricht nicht dem Doctype XHTML 1.0 Transitional, 88 Fehler mit W3C-HTML-Validator, HTML 3.2 vermutet (3 Fehler); CSS Validierung ist bei master-all.css fehlgeschlagen 22 Fehler. Benutzer-Stylesheet ist OK	Nein		M
3.3	Verwendung von Stylesheets	2	Neben dem Benutzer Stylesheet: stylesheet.css verwendet die JSF-Woodstock-Bibliothek noch ein Master-Stylesheet: master-all.css, zwei Stylesheets sind verwirrend, Tag-styles werden im Tabellenkopf verwendet	Nein		M
3.4	Verwendung von relativen Einheiten	2	Das Table-Element benutzt absolute Einheiten statt relativen	Nein	40-50	A2
3.5	Korrekte Darstellung der Dokumentenstruktur	2	h1 fehlt bei Text: "Student-IS", h2 fehlt bei Text: "Navigation, Aktionen"	Nein		M

Fortsetzung der evaluierten HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
3.6	Korrektes Markup für Listen	2	Kein Listen-Markup vorhanden, obwohl Listen in der Referenzanwendung vorhanden sind	Nein		A1
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	siehe 6.3	KA		M
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2	Element http-equiv mit Attribut refresh wird nicht benützt	Ja		A1
11.1	Verwendung von W3C-Technologien	2	Java-Script wird für fast jede Komponente verwendet	Nein		M
11.2	Vermeidung von überholten W3C-Features	2	Element IMG mit überholtem Attribut "border"	Nein	22	A1,A2
12.3	Unterteilung von Informationsblöcken	2	Das Paragraph-Tag wird beim Suche-Teil und beim Ergebnis-Teil nicht verwendet	Nein		M
13.1	Kennzeichnung von Links	2	Linktext gibt Hinweis auf das Linkziel	Ja		M
13.2	Verwendung von Metadaten	2	Meta-Element ist unvollständig Attribute "description, keywords, language" fehlen	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Site-Map vorhanden	Nein		M
13.4	Konsistente Nutzung von Navigationsmechanismen	2	Seitenlayout und Navigationsmechanismen sind konsistent mit Nachbarseiten	Ja		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltentitel sind vorhanden, Tabellentitel fehlt, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2		Ja		M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Seite funktioniert nicht bei deaktivieren von Java-Sript, kein NOSCRIPT-Element	Nein		M
6.4	Eingabegeräte unabhängige Event-Handler definieren	2		KA		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M

Fortsetzung der evaluierten HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Frames					
12.1	Betiteln der Frames	1		KA		A1
12.2	Genauere Beschreibung der Frames	2		KA		A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2	siehe 12.4	KA		M
12.4	Explizite Zuordnung von Beschriftungen	2	Label ist dem Textfeld explizit zugeordnet.	Ja		A1

Tabelle C.3 JSF, Detailergebnisse der Datei: Vorlesungen.html

Evaluierte HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1		KA		M
2.1	Informationserhalt bei Fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Die Seite enthält keine Lang-Attribute	Nein		M
6.1	Lesbarkeit ohne Stylesheets	1	Überschrift h1 ist nicht richtig dargestellt, Navigations-Links sind unübersichtlich angeordnet, Tabellengitter fehlt	Nein		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine Nicht-Text Inhalte, die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1		Ja		M
2.2	Guter Kontrast bei Farben	2	Farbkontrast-Verhältnis getestet mit: Juicy Studio Farbkontraste, 5 Fehler	Nein		M

Fortsetzung der evaluierten HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
3.1	Markup statt Bilder	2		Ja		M
3.2	Einhaltung formaler Grammatiken	2	Html-Output entspricht nicht dem Doctype XHTML 1.0 Transitional, 65 Fehler mit W3C-HTML-Validator, HTML 3.2 vermutet (4 Fehler); CSS Validierung ist bei master-all.css fehlgeschlagen 18 Fehler. Benutzer-Stylesheet ist OK	Nein		M
3.3	Verwendung von Stylesheets	2	Neben dem Benutzer Stylesheet: stylesheet.css verwendet die JSF-Woodstock-Bibliothek noch ein Master-Stylesheet: master-all.css, zwei Stylesheets sind verwirrend, Tag-styles werden im Tabellenkopf verwendet	Nein		M
3.4	Verwendung von relativen Einheiten	2	Element TD benutzt absolute Einheiten bei "width"	Nein	28ff, 45ff	A2
3.5	Korrekte Darstellung der Dokumentenstruktur	2	h1 fehlt bei Text: "Student-IS", h2 fehlt bei Text: "Navigation, Aktionen" sowie bei der Bezeichnung der Lehrveranstaltung	Nein		M
3.6	Korrektes Markup für Listen	2	Kein Listen-Markup vorhanden, obwohl Listen in der Referenzanwendung vorhanden sind	Nein		M
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	siehe 6.3	KA		M
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2	Element http-equiv mit Attribut refresh wird nicht benützt	Ja		A1
11.1	Verwendung von W3C-Technologien	2	Java-Script wird für fast alle Komponenten verwendet	Nein		M
11.2	Vermeidung von überholten W3C-Features	2	Element TD mit überholtem Attribut "width" vorhanden	Nein	28ff, 45ff	A1,A2
12.3	Unterteilung von Informationsblöcken	2		Ja		M
13.1	Kennzeichnung von Links	2		Ja		A1
13.2	Verwendung von Metadaten	2	Meta-Element ist unvollständig Attribute "description, keywords, language" fehlen	Nein		A1

Fortsetzung der evaluierten HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Site-Map vorhanden	Nein		M
13.4	Konsistente Nutzung von Navigationsmechanismen	2		Ja		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltentitel sind vorhanden, Tabellentitel fehlt, summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2		Ja		M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Seite funktioniert nicht bei deaktivieren von Java-Script, kein NOSCRIPT-Element vorhanden	Nein		M
6.4	Eingabegeräte unabhängige Event-Handler definieren	2		KA		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1		KA		A1
12.2	Genauere Beschreibung der Frames	2		KA		A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2		KA		M
12.4	Explizite Zuordnung von Beschriftungen	2		KA		A1

Tabelle C.4 JSF, Detailergebnisse der Datei: VorlesungenDetail.html

Webratio

Evaluierte HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Textfeld hat kein Alt-Attribut bzw. Label	Nein	100	A1
2.1	Informationserhalt bei Fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Das Lang-Attribut fehlt fast überall	Nein		M
6.1	Lesbarkeit ohne Style-sheets	1	Die Links in der Navigations-leiste haben keinen Abstand zueinander, sonst ok	Ja		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine Nicht-Text-Elemente die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1		Ja		M
2.2	Guter Kontrast bei Farben	2	Farbkontrast-Verhältnis getestet mit: Juicy Studio Farbkontraste, 3 Fehler	Nein		M
3.1	Markup statt Bilder	2		Ja		M
3.2	Einhaltung formaler Grammatiken	2	HTML-Output entspricht nicht Doctype HTML 1.0 Strict (40 Fehler) oder Transitional (35 Fehler), getestet mit W3C-HTML-Validator; CSS Validierung ist OK	Nein		M
3.3	Verwendung von Stylesheets	2	Tag-Styles werden beim Textfeld und bei der Tabelle verwendet	Ja	100,109	M
3.4	Verwendung von relativen Einheiten	2		Ja		M
3.5	Korrekte Darstellung der Dokumentenstruktur	2	Title-Element der Seite stimmt nicht mit dem h1-Text überein	Nein		M
3.6	Korrektes Markup für Listen	2		Ja		M
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	siehe 6.3	KA		M
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2		Ja		A1

Fortsetzung der evaluierten HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
11.1	Verwendung von W3C-Technologien	2	Javascript wird verwendet	Nein		M
11.2	Vermeidung von überholten W3C-Features	2	Element TH mit veraltetem Attribut "noWrap" gefunden	Nein	A2: 111, 112, 113	A1,A2
12.3	Unterteilung von Informationsblöcken	2		Ja		M
13.1	Kennzeichnung von Links	2	Gleicher Link-Text für unterschiedliche Ressourcen	Nein	78,83	A1
13.2	Verwendung von Metadaten	2	Metadaten im Header unvollständig	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Seite mit Site-Map. vorhanden	Nein		A1
13.4	Konsistente Nutzung von Navigationsmechanismen	2		Ja		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltentitel sind vorhanden, Tabellentitel fehlt, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2		Ja		M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Links und Button funktionieren nicht ohne javascript, Element NOSCRIPT fehlt	Nein		M
6.4	Eingabegeräte unabhängige Event-Handler definieren	2	Event-Handler "onclick" und "onkeypress" im Formularteil vorhanden	Ja	100,101	M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1		KA		A1
12.2	Genauere Beschreibung der Frames	2		KA		A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2	Das Label ist korrekt positioniert	Ja		M
12.4	Explizite Zuordnung von Beschriftungen	2	Label ist nicht explizit dem Textfeld zugeordnet	Nein	A1:54, A2:100	A1,A2

Tabelle C.5 Webratio, Detailergebnisse der Datei: Vorlesungen.html

Evaluierte HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Keine Nicht-Text-Elemente vorhanden	KA		A1
2.1	Informationserhalt bei fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Das Lang-Attribut fehlt fast überall	Nein		M
6.1	Lesbarkeit ohne Stylesheets	1	Die Links in der Navigationsleiste haben keinen Abstand zueinander, sonst ok	Ja		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine Nicht-Text-Elemente die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1		Ja		M
2.2	Guter Kontrast bei Farben	2	Farbkontrast-Verhältnis getestet mit: Juicy Studio Farbkontraste, 3 Fehler	Nein		M
3.1	Markup statt Bilder	2		Ja		M
3.2	Einhaltung formaler Grammatiken	2	HTML-Output entspricht nicht Doctype HTML 1.0 Strict (17 Fehler) oder Transitional (15 Fehler), getestet mit W3C-HTML-Validator; CSS Validierung ist ok	Nein		M
3.3	Verwendung von Stylesheets	2	Tag-Styles werden bei den Tabellen verwendet sonst ist alles im Stylesheet definiert	Ja	87,12	M
3.4	Verwendung von relativen Einheiten	2		Ja		M
3.5	Korrekte Darstellung der Dokumentenstruktur	2	Überschriften werden korrekt abgebildet aber Title-Element der Seite stimmt nicht mit h1-Text überein	Nein		M
3.6	Korrektes Markup für Listen	2		Ja		M
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	siehe 6.3	KA		M
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2		Ja		A1
11.1	Verwendung von W3C-Technologien	2	Javascript wird verwendet	Nein		M

Fortsetzung der evaluierten HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
11.2	Vermeidung von überholten W3C-Features	2	Element TH mit veraltetem Attribut "noWrap" gefunden	Nein	A2:89, 122	A1,A2
12.3	Unterteilung von Informationsblöcken	2		Ja		M
13.1	Kennzeichnung von Links	2	Der Link-Text in der Datentabelle kennzeichnet nicht das Ziel	Nein		A1
13.2	Verwendung von Metadaten	2	Metadaten im Header unvollständig	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Seite mit Site-Map. vorhanden	Nein		A1
13.4	Konsistente Nutzung von Navigationsmechanismen	2		Ja		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltentitel sind vorhanden; Tabellentitel, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2		Ja		M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Links funktionieren nicht ohne javascript, Element NOSCRIPT fehlt	Nein		M
6.4	Eingabegeräte unabhängige Event-Handler definieren	2		KA		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1		KA		A1
12.2	Genauere Beschreibung der Frames	2		KA		A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2		KA		A1
12.4	Explizite Zuordnung von Beschriftungen	2		KA		A1

Tabelle C.6 Webratio, Detailergebnisse der Datei VorlesungenDetail.html

Arcstyler

Evaluierte HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Kein NOFRAMES Element vorhanden; IMG-Element ohne Alt-Text vorhanden; INPUT-Element vom Typ Image ohne ALT-Text vorhanden	Nein	95; 102; 132	A1
2.1	Informationserhalt bei fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Die Sprache und auch Änderungen der Sprache sind nicht berücksichtigt, das Lang-Attribut kommt nie vor	Nein		M
6.1	Lesbarkeit ohne Stylesheets	1		Ja		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine nicht-Text Inhalte, die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1	Die Seite enthält ein GIF-Bild, welches aber statisch ist	Ja		M
2.2	Guter Kontrast bei Farben	2		Ja		M
3.1	Markup statt Bilder	2	Jeder Frame enthält ein Firmen-Logo (gif-Datei) Der Firmenname ist Teil der Gif-Datei und nicht als Markup gestaltet	Nein	102	M
3.2	Einhaltung formaler Grammatiken	2	DOCTYPE-Tag nicht definiert; Test mit W3C-Markup-Validation ergibt Doctype HTML 4.01, 22 Fehler; CSS-Test des default-Stylesheet ergibt 17 Fehler	Nein		A1,A2
3.3	Verwendung von Stylesheets	2	Das Stylesheet der Referenzanwendung konnte nicht eingebaut werden; Tag-Styles werden oft verwendet	Nein		M
3.4	Verwendung von relativen Einheiten	2	Absolute Einheiten beim Frameset	Nein	95	A2
3.5	Korrekte Darstellung der Dokumentenstruktur	2	Es werden keine Header-Elemente verwendet; Seitentitel ist Falsch, es gibt keine h1 Überschrift	Nein		A1,A2
3.6	Korrektes Markup für Listen	2	Die Listen aus der Referenzanwendung konnten nicht abgebildet werden	KA		A1

Fortsetzung der evaluierten HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	Im Frameset wird kein NOFRAMES-Element definiert	Nein	95	A2
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2		Ja		A1
11.1	Verwendung von W3C-Technologien	2	Es wird auch Javascript verwendet	Nein		M
11.2	Vermeidung von überholten W3C-Features	2	Überholte Attribute bei Element-script: "language"; Überholtes kursiv-Tag "i"	Nein	44-51;150	A2
12.3	Unterteilung von Informationsblöcken	2	Keine Header- oder Paragraph-Elemente vorhanden	Nein		A2
13.1	Kennzeichnung von Links	2	Links können nur als Buttons dargestellt werden, Für die Datentabellen konnten keine Links für die Zeilenauswahl erstellt werden	Nein		M
13.2	Verwendung von Metadaten	2	Metadaten im Header unvollständig	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Site-Map vorhanden	Nein		A1
13.4	Konsistente Nutzung von Navigationsmechanismen	2	Die Navigation konnte nicht so wie in der Referenzanwendung abgebildet werden, alternativ kommen Buttons als Link-Ersatz zur Anwendung	Nein		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltenüberschriften sind vorhanden, Titel-Element fehlt, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2	In jedem Sub-Frame ist ein Layout-Table vorhanden	Nein	103	M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Event-Handler der Buttons und die Auswahl der Tabellenzeile funktionieren bei deaktiviertem Javascript nicht mehr, es gibt kein NOSCRIPT-Element	Nein		M

Fortsetzung der evaluierten HTML-Seite: Vorlesungen.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Applets und Scripts					
6.4	Eingabegeräte unabhängige Event-Handler definieren	2	Event-Handler "onclick" wird für Buttons verwendet. Bei abgeschalteten Mouse-Events funktioniert auch das Keyboard nicht mehr; Die Zeilenauswahl in der Tabelle ist nur per Mauszeiger möglich	Nein		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1	Frame Elemente enthalten kein Title-Attribut	Nein	99ff	A1,A2
12.2	Genauere Beschreibung der Frames	2	Das "longdesc" Attribut wird nicht verwendet	Nein	99ff	A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2		KA		M
12.4	Explizite Zuordnung von Beschriftungen	2		Ja		A1

Tabelle C.7 Arcstyler, Detailergebnisse der Datei: Vorlesungen.html

Evaluierte HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
1.1	Text-Äquivalente für Nicht-Text-Elemente	1	Kein NOFRAMES Element vorhanden; IMG-Element ohne Alt-Text vorhanden; INPUT-Element vom Typ Image ohne ALT-Text vorhanden	Nein	95; 102; 132	A1
2.1	Informationserhalt bei fehlen von Farbe	1		Ja		M
4.1	Kennzeichnung der Sprache bei Änderung der Sprache	1	Die Sprache und auch Änderungen der Sprache sind nicht berücksichtigt, das Lang-Attribut kommt nie vor	Nein		M
6.1	Lesbarkeit ohne Stylesheets	1		Ja		M
6.2	Änderung der Äquivalente bei geänderten dyn. Inhalten	1	Es gibt keine Nicht-Text Inhalte, die sich dyn. Ändern	KA		M
7.1	Vermeidung von Bildschirmflackern	1		Ja		M

Fortsetzung der evaluierten HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
2.2	Guter Kontrast bei Farben	2		Ja		M
3.1	Markup statt Bilder	2	Jeder Frame enthält ein Firmen-Logo (Gif-Datei) Der Firmenname ist Teil der Gif-Datei und nicht als Markup gestaltet	Nein		M
3.2	Einhaltung formaler Grammatiken	2	DOCTYPE-Tag nicht definiert; Test mit W3C-Markup-Validation ergibt Doctype HTML 4.01, 22 Fehler; CSS-Test des default-Stylesheet ergibt 17 Fehler	Nein		A1,A2
3.3	Verwendung von Stylesheets	2	Das Stylesheet der Referenzanwendung konnte nicht eingebaut werden; Tag-Styles werden oft verwendet	Nein		M
3.4	Verwendung von relativen Einheiten	2	Absolute Einheiten beim Frameset	Nein	95	A2
3.5	Korrekte Darstellung der Dokumentenstruktur	2	Es werden keine Header-Elemente verwendet; Seitentitel ist Falsch, es gibt keine h1 Überschrift	Nein		A1,A2
3.6	Korrektes Markup für Listen	2	Die Listen aus der Referenzanwendung konnten nicht abgebildet werden	KA		A1
3.7	Korrektes Markup für Zitate	2		KA		A1
6.5	Zugänglichkeit von dynamischen Inhalten	2	Im Frameset wird kein NOFRAMES-Element definiert	Nein	95	A2
7.2	Vermeidung von blinkenden Inhalten	2		Ja		M
7.4	Vermeidung von autom. Refresh der Seite	2		Ja		A1
11.1	Verwendung von W3C-Technologien	2	Es wird auch Javascript verwendet	Nein		M
11.2	Vermeidung von überholten W3C-Features	2	Element-TR mit Attribut "bgcolor"; Überholte Attribute bei Element-script: "language"; Überholtes kursiv-Tag "i"	Nein	44-51;150	A1,A2
12.3	Unterteilung von Informationsblöcken	2	Es werden keine Header- oder Paragraph-Elemente verwendet	Nein		A2

Fortsetzung der evaluierten HTML-Seite: VorlesungenDetail.html

Nr.	Evaluierungskriterien	Pr.	Beschreibung	Bw.	Zeilen-Nr.	Mod.
	Allgemein					
13.1	Kennzeichnung von Links	2	Links können nur als Buttons dargestellt werden, Für die Datentabellen konnten keine Links für die Zeilenauswahl erstellt werden	Nein		M
13.2	Verwendung von Metadaten	2	Metadaten im Header unvollständig	Nein		A1
13.3	Bereitstellung von Layout-Informationen (Site-Map)	2	Keine Site-Map vorhanden	Nein		A1
13.4	Konsistente Nutzung von Navigationsmechanismen	2	Die Navigation konnte nicht so wie in der Referenzanwendung abgebildet werden, alternativ kommen Buttons als Link-Ersatz zur Anwendung	Nein		M
	Tabellen					
5.1	Kennzeichnung von Tabelleninhalten	1	Spaltenüberschriften sind vorhanden, Titel-Element fehlt, Summary fehlt	Ja		M
5.3	Tabellen nicht für Layout-Zwecke verwenden	2	In jedem Sub-Frame ist ein Layouttable vorhanden	Nein	103	M
	Applets und Scripts					
6.3	Informationserhalt bei Deaktivierung von Scripts	1	Event-Handler der Buttons und die Auswahl der Tabellenzeile funktionieren bei deaktiviertem Javascript nicht mehr, es gibt kein NOSCRIPT-Element	Nein		M
6.4	Eingabegeräte unabhängige Event-Handler definieren	2		KA		M
8.1	Für assistierende Technologien zugänglich machen	2		KA		M
	Frames					
12.1	Betiteln der Frames	1	Frame Elemente enthalten kein Title-Attribut	Nein	99ff	A1,A2
12.2	Genauere Beschreibung der Frames	2	Das "longdesc" Attribut wird nicht verwendet	Nein	99ff	A1
	Formulare					
10.2	Korrekte Position bei impliziter Zuordnung von Beschriftungen	2		KA		M
12.4	Explizite Zuordnung von Beschriftungen	2		KA		A1

Tabelle C.8 Arcstyler, Detailergebnisse der Datei: VorlesungenDetail.html