

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN  
Department "Institut für Informatik"  
Lehr- und Forschungseinheit Medieninformatik  
Prof. Dr. Heinrich Hußmann

## **Projektarbeit**

# Semi-automatische Usability-Analyse des Layouts von Webseiten

Ronald Ecker  
ecker@informatik.uni-muenchen.de

Bearbeitungszeitraum: 15.04.2006 bis 15.11.2006  
Betreuer: Dipl.-Inf. Richard Atterer  
Verantw. Hochschullehrer: Dr. Albrecht Schmidt







## **Kurzzusammenfassung:**

Diese Arbeit gibt einen Überblick über die Möglichkeiten und Grenzen der automatischen Usability-Analyse von Webseiten. Meist dienen Tools zur Usability-Evaluierung der Unterstützung von Experten und decken nicht den gesamten Evaluierungsprozess ab. WUSAB zeigt, dass anhand zusätzlicher Informationen genauere Aussagen über die Usability einer Webseite getroffen werden können. Dabei beschränkt es sich auf die Analyse des Layouts. Es beschreibt eine Methode die sowohl zur Verwendung mit Web-Engineering-Modellen als auch zur Überprüfung bestehender Webseiten durch den Entwickler geeignet ist und deckt zugleich den gesamten Prozess der Usability-Evaluierung, abgesehen von Sammeln der Daten, ab.

## **Abstract:**

This work gives an overview of automated usability evaluation techniques for websites and demonstrates there lacks and advantages. Most of the existing tools are supporting usability experts and rarely cover all steps in the process of evaluation. WUSAB shows that, in compination with additional content a website can be analysed more exactly. Its limited to the analysis of the layout of websites. WUSAB describes a method which is capable to receive this additional Information from a modell or existing website in addition with annotation from a user. It covers all aspect of evaluation except the data capturing.



## **Aufgabenstellung:**

*Thema:* **Tool zur semi-automatischen Usability-Analyse bestehender Webseiten.**

Ziel dieser Arbeit ist es, einen bereits bestehenden Prototypen einer Software zur semi-automatischen Usability-Analyse zu verbessern und erweitern. Dabei werden folgende Teilaufgaben im Fokus dieser Arbeit stehen:

- Einarbeitung in den Quelltext der bestehenden Software.
- Erhebung und Analyse vorhandener Werkzeuge zur Usability Evaluierung bestehender Websites.
- Entwicklung eines geeigneten Algorithmus zur Ermittlung der Anordnung von Seitenelementen wie Navigation, Inhalt, Werbung, Logo, etc. bei TABLE- und CSS-basierten Layouts.
- Algorithmen zur Evaluierung der Benutzbarkeit der Seite anhand der de facto Usability Standards:
  - Ermittlung und Überprüfung der Flächenverteilung der Seitenelemente.
  - Überprüfung der korrekten Anordnung von Seitenelementen.
  - Überprüfung der Skalierbarkeit von Seitenelementen und Ermittlung der Darstellungsflexibilität bei unterschiedlichen Bildschirmgrößen und -auflösungen.





„Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.“

München, den 22. Nov. 2006

---

Ronald Ecker



## Inhaltsverzeichnis

1	Einleitung .....	1
2	Hintergrund .....	3
2.1	Richtlinien (Guidelines).....	3
2.2	Automatische Evaluierung.....	3
2.2.1	Methoden der Evaluierung.....	4
2.2.2	Grad der Automatisierung.....	4
2.2.3	Methoden und Werkzeuge der automatischen Evaluierung von Webseiten .....	5
3	Usability von Layout und Design.....	9
3.1	Allgemeines .....	9
3.2	Anordnung der Seitenbereiche.....	10
3.2.1	Logo .....	10
3.2.2	Navigation.....	10
3.2.3	Inhalt .....	11
3.2.4	Werbung.....	11
3.2.5	Suche.....	11
3.2.6	Zusätzlicher Inhalt / Informationen.....	12
3.3	Größe der Seitenbereiche.....	13
3.3.1	Inhalt .....	13
3.3.2	Navigation.....	13
3.3.3	Werbung.....	14
3.4	Anpassungsfähigkeit des Seitenlayouts .....	14
4	Umsetzung.....	15
4.1	Allgemeines .....	15
4.2	Arbeitsweise des Prototypen.....	15
4.2.1	Clientseitig .....	15
4.2.2	Serverseitig.....	17
4.3	Parsen des Dokumentes .....	17
4.4	Erhebung der Positionsdaten.....	18
4.5	Konfiguration des Prototypen.....	19
4.6	Auswertung der Daten .....	20
4.6.1	Testen der Anordnung der Seitenelemente .....	20
4.6.2	Testen der Größe der Seitenelemente.....	21
4.6.3	Testen der Anpassungsfähigkeit der Webseite.....	22

4.7	Architektur und Klassen .....	22
5	Testen des Prototypen .....	27
6	Resümee .....	31
	Quellen: .....	33
	Online-Quellen: .....	35





# 1 Einleitung

Ein wesentlicher Bestandteil des Erfolges eines Webauftritts ist seine Benutzerfreundlichkeit. Unter Benutzerfreundlichkeit, im Folgenden auch Usability genannt, versteht die International Standards Organization (ISO):

*„Usability ist the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.“ [1]*

Findet sich der Benutzer auf einer Webseite nicht zurecht oder kann er das erhoffte Ziel mit seinem Besuch auf der Seite nicht mit der gewünschten Effizienz erreichen, wendet er sich meist von der Seite ab [2]. Das Ziel eines kommerziellen, informativen oder privaten Webangebotes ist es, den Benutzer auf der Seite zu halten und ihn zum gewünschten Ziel zu führen, denn nur ein zufriedener Nutzer wird die Seite wiederholt besuchen. Zugleich soll eine möglichst breite Masse an Benutzern angesprochen werden. Das heißt, jedem sollte die Seite dargestellt werden können, unabhängig von technischen Gegebenheiten wie Browserversion oder Betriebssystem. Auch Menschen mit eingeschränkten physischen Fähigkeiten muss die Chance geboten werden das Webangebot zu nutzen. Zu diesem Zweck wurden verschiedene Accessibility & Usability Richtlinien entworfen. Für das World Wide Web Consortium (W3C) [1] bedeutet Web Accessibility:

*“... that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web.”*

Der Grad an Usability und Accessibility stellt also, aus Sicht des Users, ein wesentliches Qualitätsmerkmal der Benutzeroberfläche und Struktur von Webseiten dar. Aus diesem Grund sollte ihnen eine zentrale Rolle in der Entwicklungsphase zukommen. Sie sind jedoch nicht untrennbar miteinander verbunden. Seiten die für einen Benutzer gut bedienbar und verständlich sind können fehlende Accessibility aufweisen. Umgekehrt können Webauftritte mit hervorragender Accessibility Mängel in der Bedienerfreundlichkeit aufweisen [5]. Ein wichtiger Punkt bei der Entwicklung ist die Evaluierung. Nielsen [4] beschreibt Usability-Evaluierung als:

*“... consists of methodologies for measuring the usability aspects of a system’s user interface and identifying specific problems.”*

Sie dient zur Feststellung von Usability und Accessibility-Problemen von Benutzeroberflächen. Vielen Designern fehlt jedoch das theoretische Wissen um die Evaluierung selbst durchzuführen. Oft kann aus Kostengründen nicht für jede Webseite ein Team von Usability-Experten konsultiert werden. Eine Alternative zur klassischen Usability-Evaluierung ist der Einsatz automatischer Werkzeuge [3]. Die automatische Usability- und Accessibility-Evaluierung spart neben Kosten noch Zeit und kann im Entwicklungsprozess einer Seite wiederholt eingesetzt werden, was vor allem bei ständig wachsenden oder sich verändernden Webauftritten von Vorteil ist. Ivory und Hearst [3] konstatieren jedoch, dass die Verwendung von automatischen Evaluierungswerkzeugen zwar eine Ergänzung aber kein Ersatz für die herkömmliche Evaluierung, wie heuristische Untersuchungen, ist. Sie kann beispielsweise die Zufriedenheit eines Benutzers nicht feststellen.

Ziel dieser Arbeit ist die Entwicklung eines Tools zur semiautomatischen Layout-Analyse bestehender Webseiten. Semiautomatisch deshalb, da man zusätzlich zu den aus dem Quellcode einer Seite gewonnen, weitere Informationen benötigt.

In Kapitel 2 werden verschiedene Methoden der automatischen Usability-Evaluierung und einige vorhandene Werkzeuge beschrieben und die Problemstellung der Arbeit näher erläutert.

Kapitel 3 befasst sich mit der Usability von Layout und Design.

In Kapitel 4 wird der praktische Teil der Arbeit beschrieben. Abschließend werden Ergebnisse von Tests an bestehenden Webseiten vorgestellt.



## 2 Hintergrund

Zu Beginn dieses Abschnitts werden vorhandene Accessibility und Usability Richtlinien vorgestellt. Des Weiteren werden verschiedene Methoden sowie die mögliche Automatisierung notwendiger Arbeitsschritte der Evaluierung skizziert. Abschließend gilt es, die Methoden der Evaluierung, welche automatisch auf Webseiten angewandt werden können, zu präsentieren und die Problemstellung dieser Arbeit anhand bestehender automatischer Evaluierungswerkzeuge zu erläutern.

### 2.1 Richtlinien (Guidelines)

Zur Sicherstellung von Accessibility, *capable of being reached*, und Usability, *capable of being used*, wurden eine Reihe von Guidelines, für Webseitendesigner und andere die in den Entwicklungsprozess einer Webseite involviert sind, entworfen. Usability-Guidelines bestehen meist aus abstrakten Richtlinien die eher der Natur von Empfehlungen gleichen. Accessibility-Richtlinien hingegen haben die Form von spezifischen Vorschriften oder Regeln, die genaue Angaben über HTML-Elemente machen, wie das ALT-Attribut bei IMG-Tags, die der Quellcode einer Webseite enthalten muss.

Die wichtigsten Usability-Guidelines sind der Web Style Guide von Lynch und Horton [2] und die Research-Based Web Design- & Usability-Guidelines vom U.S Department of Health and Human Services (HHS) [3]. Letzteres stellt eine Zusammenfassung aller vorhandenen Usability Web Design Guidelines und Style Guides sowie Studien des HHS dar. Im Fokus dieser Richtlinien stehen informationsbasierte Webseiten. Ähnlich wie der Web Style Guide decken diese eine große Anzahl an für Usability relevanten Design Aspekten wie Navigationsstruktur, Gestaltung des Inhaltes und Design des Layouts von Webseiten ab. Nielsen aktualisiert außerdem regelmäßig seine Alertbox [4], in der er Ratschläge und neue Erkenntnisse zum Thema Usability im Web veröffentlicht.

Accessibility-Guidelines wie die im Rahmen der Web Accessibility Initiative (WAI) des W3C entworfenen Web Content Accessibility Guidelines (WCAG) [1] und Section 508 [5] der amerikanischen Regierung, dienen zur barrierefreien Umsetzung von Webseiten. Ziel dieser Richtlinien ist es behinderten Menschen zu ermöglichen Webangebote zu nutzen, unabhängig von der verwendeten Software oder dem benutzten Endgerät (z.B. desktop Browser, voice Browser, Mobiltelefon, etc.). Dabei sollten Webdesigner nicht in der Verwendung von Designelementen wie Video, Audio oder Bildern eingeschränkt werden. Jedoch sollten Webseiten auch ohne diese gestalterischen Elemente benutzbar sein. Davon profitieren nicht nur Menschen mit physischer Beeinträchtigung. Sie garantieren zusätzlich eine gewisse Browserunabhängigkeit.

### 2.2 Automatische Evaluierung

Wie schon in der Einleitung erwähnt ist die Evaluierung ein wichtiger Bestandteil der Entwicklung von benutzerfreundlichen Schnittstellen. Grundsätzlich sind drei Schritte für die Evaluierung nötig:

- Sammeln von Usability Daten: Fehler bei der Bedienung, Verletzung verschiedener

Richtlinien, subjektive Wertung der Testperson.

- Analysieren der Usability Daten: Interpretation der Daten, Identifizierung von Usability Problemen.
- Kritik: Vorschläge zur Verbesserung, Beheben von Fehlern wenn möglich.

Es existiert eine große Anzahl an Techniken zur Evaluierung der Usability, die oft grundlegend verschiedene Ergebnisse liefern, da sie meist nur eine ausgewählte Teilmenge an möglichen Nutzeraktion oder Usability Aspekten berücksichtigen. Deshalb empfiehlt Nielsen die Verwendung verschiedener Evaluationstechniken [4]. Der Einsatz von Usability-Experten zur Evaluierung, welche das nötige Wissen und die nötigen Techniken beherrschen, kostet Zeit und Geld. Automatische Evaluierungswerkzeuge können dem Entwickler oder Usability-Experten die Arbeit wesentlich erleichtern. Im Zentrum der vorliegenden Arbeit stehen genau diese automatischen Techniken der Usability-Evaluierung.

Um eine Unterscheidung verschiedener Usability- und Accessibility-Evaluationstechniken zu ermöglichen, betrachtet Ivory [3] diese aus vier Perspektiven:

- Methodenklassen: Erlauben eine abstrakte Unterscheidung der Methoden.
- Methodentypen: Beschreiben die Methoden genauer.
- Der Grad der Automatisierung: Lässt eine Aussage über die automatisierbaren Schritte der Evaluierungsmethode zu.
- Aufwand: Beschreibt den Aufwand der vonseiten einer Person erbracht werden muss um eine Methode anzuwenden.

Im Nachstehenden wird Ivorys Arbeit genauer vorgestellt um einen Überblick der existierenden Methoden zu erhalten.

### **2.2.1 Methoden der Evaluierung**

Eine Übersicht aller Methodenklassen und der dazugehörigen Methodentypen zur Evaluierung von Webseiten wird in Tabelle 1 gezeigt. Ivory unterscheidet folgende Methodenklassen:

- Testen: Betrachtung der Interaktionen von Testpersonen mit der Benutzeroberfläche und Feststellen von Mängeln der Usability.
- Inspektion: Eine Menge von Kriterien oder Richtlinien werden verwendet um die Usability zu bestimmen.
- Befragung: Testperson äußert Meinung zur Benutzeroberfläche.
- Analytisches Modellieren: Anhand von Modellen der Benutzerschnittstelle oder Benutzeraktion Vorhersagen über die Usability denkbar.
- Simulation: Anhand von Modellen der Benutzerschnittstelle und Benutzeraktion können Interaktionen simuliert werden.

Mit Hilfe der ersten drei Methodenklassen können konkrete sowie generelle Aussagen über die Usability einer bestehenden Webseite getroffen werden. Analytisches Modellieren und die Simulation können, schon während des Entwicklungsprozesses, unter Verwendung verschiedener Modelle Vorhersagen über die spätere Benutzerfreundlichkeit treffen.

### **2.2.2 Grad der Automatisierung**

Der Grad der Automatisierung zeigt welche Schritte einer Evaluierungsmethode automatisch umsetzbar sind. Von Balbo [6] wurden vier Kategorien der Automatisierung definiert:

- keine Automatisierung
- automatisches Sammeln von Daten
- automatische Auswertung
- automatisches Generieren von Verbesserungsvorschlägen oder / und Beheben von Fehlern

Da, abhängig von der verwendeten Methode, nicht alle Aspekte der Evaluierung automatisch durchführbar sind, werden oft zusätzliche Informationen von Modellen oder einem Benutzer benötigt. Balbo [6] bezeichnet dies als *Effort Level* und unterscheidet zwischen:

- Minimaler Aufwand: es werden keine zusätzlichen Informationen benötigt.
- Entwicklung eines Modells (M): um die Methode auszuführen wird ein Modell der Benutzerschnittstelle oder des Benutzers benötigt.
- Informelle Benutzung (I): die Testperson kann eine frei wählbare Aufgabe auswählen und durchführen.
- Formelle Benutzung (F): die Testperson muss sich an vorgegebene Aufgaben halten und diese erfüllen.

### 2.2.3 Methoden und Werkzeuge der automatischen Evaluierung von Webseiten

Tabelle 1 zeigt, dass nicht alle Evaluierungsmethoden für Webseiten automatisiert werden können. Tabelle 2 listet alle möglichen Methoden zur Evaluierung von Benutzeroberflächen im Allgemeinen auf.

Es können drei Methodentypen der Klasse *Testen* auf Webseiten angewandt werden. Das *Messen der Leistungsfähigkeit* (Performance Measurement) einer Webseite unter der Verwendung von Log-Dateien anhand derer Aktionen des Benutzers rekonstruiert werden können. WebVIP [6], ein Evaluierungstool entwickelt von NIST, fügt der zu testenden Webseite einen Code hinzu, welcher jede Aktion, wie Selektieren eines Links oder betätigen des „Zurück“ Buttons, des Benutzers in einer Log-Datei, zusammen mit einem Zeitstempel, ablegt. Dazu muss eine lokale Kopie des Quellcodes der Seite vorhanden sein. Daten wie z.B. die Bewältigungsdauer einer Aufgabe oder das Klickverhalten eines Benutzers auf der Seite werden ermittelt und gespeichert. Der Grad der Automatisierung beschränkt sich in diesem Fall auf das Sammeln von Daten. Ein wesentlicher Nachteil dieses Tools besteht darin, dass aufgrund der Kopie der Seite nicht die Möglichkeit besteht etwaige externe Links zu berücksichtigen.

VISVIP [13] visualisiert den Navigationspfad des Benutzers durch eine Webseite mit Hilfe der von WebVIP erstellten *Log-Dateien* (Log File Analysis). Die auswertende Person hat die Möglichkeit, verschiedene Ansichten des Navigationspfades zu wählen und kann somit Aussagen über die Eignung der Navigationsstruktur treffen.

WebCAT [14], ein ebenso von NIST entwickeltes Tool zum *Entfernten Testen* (Remote Testing), fordert Testpersonen auf verschiedene Themen in Kategorien einzuordnen. Die Testperson sollte keinesfalls die vom Entwickler geplante Einteilung der Themen in Kategorien kennen. Je nach Übereinstimmung der Zuordnungen von Testpersonen und Webseitendesigner, kann eine Bewertung der Organisation von Informationen auf der Seite stattfinden.

Method Class Method Type	Automation Type			
	None	Capture	Analysis	Critique
<b>Testing</b>				
Thinking-Aloud Protocol	F (1)			
Question-Asking Protocol	F (1)			
Shadowing Method	F (1)			
Coaching Method	F (1)			
Teaching Method	F (1)			
Codiscovery Learning	F (1)			
Performance Measurement	F (1)	F (3)	IFM (9)	
Log File Analysis				
Retrospective Testing	F (1)			
Remote Testing		IF (3)		
<b>Inspection</b>				
Guideline Review	IF (4)		(5)	(6)
Cognitive Walkthrough	IF (2)			
Pluralistic Walkthrough	IF (1)			
Heuristic Evaluation	IF (1)			
Perspective-Based Inspection	IF (1)			
Feature Inspection	IF (1)			
Formal Usability Inspection	F (1)			
Consistency Inspection	IF (1)			
Standards Inspection	IF (1)			
<b>Inquiry</b>				
Contextual Inquiry	IF (1)			
Field Observation	IF (1)			
Focus Groups	IF (1)			
Interviews	IF (1)			
Surveys	IF (1)			
Questionnaires	IF (1)	IF (1)		
Self-Reporting Logs	IF (1)			
Screen Snapshots	IF (1)			
User Feedback	IF (1)			
<b>Analytical Modeling</b>				
No Methods Surveyed				
<b>Simulation</b>				
Information Proc. Modeling			M (1)	
Information Scent Modeling		M (1)		

Tabelle 1 [3]: Methoden der automatischen Evaluierung von Webseiten.

Eine weitere Methode zur Evaluierung der Usability ist die *Befragung*, welche, ähnlich wie das Testen, von Benutzeraktionen abhängig ist. Im Gegensatz zum Testen dient die Befragung jedoch nicht der Betrachtung der Leistungsfähigkeit einer Webseite. Ihr Ziel ist die subjektive Meinung des Benutzers zur Benutzeroberfläche. NetRaker ist ein auf HTML-Formularen und Popups basierendes Tool, das der Testperson während der Benutzung einer Webseite Frage zur Benutzeroberfläche stellt. Der Vorteil dieses Tools ist, dass es eine große Anzahl an Benutzern erreichen kann. Es gestaltet sich allerdings als schwierig, aus den unspezifischen Antworten der Testpersonen konkrete Aussagen über Usabilityprobleme der Seite zu treffen, da keine automatische Analyse oder Auswertung stattfindet.

Chi E. beschreibt in [8] eine Methode zur *Simulation* der Interaktion eines Benutzers mit einer Webseite. Er präsentiert ein System, welches das Verhalten eines Users und die Usability einer Webseite analysiert und vorhersagt. Sein Ansatz sammelt und erstellt Navigationspfade für ein Webinterface. Dazu erzeugt er Modelle einer existierenden Seite, welche Informationen über die Ähnlichkeit der Inhalte zwischen den Seiten, Logdaten des Servers und die Linkstruktur enthalten. Die auswertende Person bestimmt Startpunkte in der Seite und Informationen, in der Form spezifischer Seiten, die erreicht werden sollten. Die Simulation erzeugt eine Reihe von Modellen, die potenzielle Nutzer darstellen sollen, welche die Links und Informationen des Seitenmodells durchlaufen. Navigationsentscheidungen werden zufällig getroffen, sodass die verschiedenen Benutzermodelle unterschiedlich lange Pfade der Navigation aufweisen. Wird eine festgelegte Anzahl an Versuchen oder durchlaufenen Seiten erreicht, stoppt die Simulation.

Alle zum Ziel führenden Navigationspfade werden aufgezeichnet. Diese können visualisiert werden, um Aussagen über die Usability treffen zu können. Ivory stellt jedoch in Frage ob diese Methode tatsächlich Nutzeraktionen widerspiegelt [3].

Method Class Method Type	Description
<b>Testing</b>	
Thinking-Aloud Protocol	user talks during test
Question-Asking Protocol	tester asks user questions
Shadowing Method	expert explains user actions to tester
Coaching Method	user can ask an expert questions
Teaching Method	expert user teaches novice user
Codiscovery Learning	two users collaborate
Performance Measurement	tester records usage data during test
Log File Analysis	tester analyzes usage data
Retrospective Testing	tester reviews videotape with user
Remote Testing	tester and user are not collocated during test
<b>Inspection</b>	
Guideline Review	expert checks guideline conformance
Cognitive Walkthrough	expert simulates user's problem solving
Pluralistic Walkthrough	multiple people conduct cognitive walkthrough
Heuristic Evaluation	expert identifies violations of heuristics
Perspective-Based Inspection	expert conducts narrowly focused heuristic evaluation
Feature Inspection	expert evaluates product features
Formal Usability Inspection	expert conducts formal heuristic evaluation
Consistency Inspection	expert checks consistency across products
Standards Inspection	expert checks for standards compliance
<b>Inquiry</b>	
Contextual Inquiry	interviewer questions users in their environment
Field Observation	interviewer observes system use in user's environment
Focus Groups	multiple users participate in a discussion session
Interviews	one user participates in a discussion session
Surveys	interviewer asks user specific questions
Questionnaires	user provides answers to specific questions
Self-Reporting Logs	user records UI operations
Screen Snapshots	user captures UI screens
User Feedback	user submits comments
<b>Analytical Modeling</b>	
GOMS Analysis	predict execution and learning time
UIDE Analysis	conduct GOMS analysis within a UIDE
Cognitive Task Analysis	predict usability problems
Task-Environment Analysis	assess mapping of user's goals into UI tasks
Knowledge Analysis	predict learnability
Design Analysis	assess design complexity
Programmable User Models	write program that acts like a user
<b>Simulation</b>	
Information Proc. Modeling	mimic user interaction
Petri Net Modeling	mimic user interaction from usage data
Genetic Algorithm Modeling	mimic novice user interaction
Information Scent Modeling	mimic Web site navigation

**Tabelle 2 [3]: Beschreibungen aller Evaluierungsmethoden von Benutzeroberflächen.**

Das am weitest verbreitete Einsatzgebiet von automatischen Werkzeugen zur Evaluierung ist das *Überprüfen von Guidelines* (Guideline Review) der Methodenklasse *Inspektion*. Die meisten Tools beschränken sich auf Accessibility-Guidelines. Sie sind aufgrund ihrer spezifischen Definition einfacher zu überprüfen als Usability-Richtlinien. Diese Methode setzen eine Vielzahl von Tools, wie z.B. A-Prompt, Evallris, WebSAT, TAW und EVALACCESS ein und verwenden eine ähnliche Technik. Das bekannteste Tool in diesem Bereich ist Bobby [watchfire]. Es analysiert den HTML-Code einer Webseite und überprüft ob alle für die Accessibility relevanten HTML-Tags vorhanden sind. Es ermöglicht dem Benutzer verschiedene Guidelines auszuwählen. Ferner berechnet es die Ladezeit einer Seite und überprüft die auf der Seite enthaltenen Links auf deren Gültigkeit. Bobby evaluiert also die Accessibility und einige Aspekte der Usability und generiert automatisch

### Verbesserungsvorschläge.

Die hier beschriebenen Werkzeuge dienen großteils der Unterstützung von Usability Experten, da sie nur einige Aufgaben der Evaluierung automatisch übernehmen. Tools wie Bobby hingegen generieren oft eine Vielzahl an allgemeinen Fehlermeldungen [7] oder können nur einige wenige Aspekte der Usability berücksichtigen, da nicht alle nötigen Informationen aus dem Quellcode einer Webseite gewonnen werden können. Atterer [7] schlägt vor, die von Web Engineering Lösungen wie UWE oder OO-H erzeugten Modelle zusätzlich in die Evaluierung einzubeziehen. Dabei handelt es sich um die abstrakte Beschreibung der Benutzeroberfläche oder Navigation einer Webseite. Solche Modelle bieten die Möglichkeit Verwendungszwecke wie Navigation, Werbung oder Inhalt verschiedener Seitenbereiche zu beschreiben. Mit diesen zusätzlichen Informationen könnten Usability-Evaluierungen und Analysen, wie die Überprüfung des Layouts, durchgeführt werden. Atterer stellt in [7] den dieser Arbeit zugrunde liegenden Prototypen WUSAB vor. Er verwendet eine Webinterface zum Annotieren der einzelnen Seitenbereiche, was den Informationen entspricht die ebenso aus Modellen gewonnen werden könnten. Die Zielsetzung dieser Arbeit ist die Umsetzung der Analyse des Layouts. Sie beschränkt sich auf die Überprüfung der korrekten Anordnung verschiedener Seitenbereiche, die Größe der Seitenbereiche und die Anpassung der Seite an verschiedene Bildschirmauflösungen von TABLE- und DIV-basierten Layouts.

Die von WUSAB verwendete Methode wird als *Analyse des Designs* (Design Analysis) bezeichnet und fällt in die Klasse des *analytischen Modellierens*. Ein wesentlicher Vorteil dieser Methode ist, dass man schon in einem frühen Stadium der Entwicklung einer Webseite Probleme der Usability erkennen kann.

## 3 Usability von Layout und Design

Im Folgenden sollen verschiedene für das Design und Layout spezifische Richtlinien aus vorhandenen Usability-Guidelines beschrieben werden. Außerdem findet die Festlegung von für die Implementierung nötigen Grenzwerten statt. Die in diesem Abschnitt festgelegten Regeln decken nicht alle Aspekte der benutzerfreundlichen Gestaltung einer Webseite ab. Es werden jedoch mehrere Designaspekte des Layouts betrachtet die in der Umsetzung keine Beachtung finden.

### 3.1 Allgemeines

Das User-Interface-Design der Benutzeroberfläche ist ausschlaggebend für die Entwicklung von benutzerfreundlichen Webseiten. Im Mittelpunkt des User-Interface-Designs steht die Konzeption und ist nicht mit dem grafischen Design zu verwechseln, dessen Zweck die grafische Umsetzung der Benutzeroberfläche unter Berücksichtigung des User-Interface-Designs, ist. Das Design der Benutzeroberfläche hingegen ist die Aufgabe eines Usability Experten [9]. Er stellt sicher, dass das Zusammenspiel zwischen den Funktionen und der Bedeutung des Inhaltes einer Webseite für den Benutzer verständlich ist [10]. Das beinhaltet, neben der Konzeption von Interaktionsmetaphern und der Organisation des Inhaltes, die visuelle Beschreibung der einzelnen Komponenten einer Webseite. Er legt das Layout einer Seite fest, bestimmt welche Schriftgrößen oder Schriftarten verwendet werden oder an welcher Stelle die Navigation oder andere Seitenbereiche platziert werden müssen. Dabei kann es, abhängig vom Verwendungszweck der Seite, zu Unterschieden kommen. In dieser Arbeit gilt es, einige designspezifische Aspekte der Usability von informationsbasierten Webseiten automatisch zu überprüfen. Zu diesem Zweck werden folgende Seitenbereiche unterschieden:

- Inhalt: Beinhaltet die Informationen der Seite.
- Logo: Logo der Institution oder Firma.
- Navigation: Hauptmenü sowie etwaige Untermenüs, falls notwendig.
- Werbung: Webbanner oder Anzeigen.
- Zusätzliche Information / Inhalt: Weiterführende Informationen, verwandte Themen oder Impressum.
- Suche: Suchfunktion der Seite.

Bei Navigation und Inhalt handelt es sich um für die Webseite obligatorische Seitenelemente.

Wie schon in Kapitel 2 erwähnt, bestehen die Usability-Guidelines, die dem Entwickler einer Webseite als Richtlinien dienen, aus allgemeinen Beschreibungen des Designs der Benutzeroberfläche. Sie verzichten größtenteils auf konkrete Angaben, wie die Position oder maximale Größe verschiedener Seitenelemente. Dementsprechend schwierig stellte sich die Festlegung konkreter Regeln für die spätere Implementierung heraus. Zudem galt es Grenzwerte wie die Mindestbreite des Inhaltes oder der Navigation zu ermitteln. Deren Bestimmung erfolgte anhand der Betrachtung verschiedener Webseiten. Da es sich um Grenzwerte handelt und die meisten populären Webseiten ein sehr ausgeklügeltes Design haben, wurden Webseiten

die „gerade noch so gehen“ gesucht und vermessen. Im Folgenden werden Regeln für die praktische Umsetzung aus den bestehenden Style-Guides sowie anhand verschiedener de-facto-Standards festgelegt.

Als de-facto-Standard versteht Nielsen [4] ein Design das von 80% der populärsten Webseiten umgesetzt wird, außer es existiert eine um 100% benutzerfreundlichere Lösung. Als Konvention bezeichnet er eine von 50-79% der populärsten Webseiten eingesetzte Lösung, die dem Benutzer häufig somit vertraut erscheint und dadurch intuitiv bedienbar ist. Liegt das verwendete Design unter 50% erwartet der Benutzer keine bestimmte Realisierung. Er begründet die Existenzberechtigung dieser Regelung mit Jakob`s Law: Der Benutzer verbringt die meiste Zeit auf anderen Webseiten.

## 3.2 Anordnung der Seitenbereiche

Einer der wichtigsten Aspekte bei der Anordnung von Seitenelementen ist die Konsistenz. Wurde einmal ein Entwurf für den Aufbau einer Seite erstellt, sollte dieser einheitlich für alle Seiten des Webauftrittes beibehalten werden. Im Rahmen dieser Arbeit wird die Konsistenz jedoch nicht berücksichtigt. Ferner hänge die „beste“ Anordnung der Seitenelemente vom Verwendungszweck des Webauftrittes ab. Die sinnvolle Platzierung für Trainings-, Schulungs-, News-, E-Commerce- und Informationsseiten kann stark variieren. Der folgende Abschnitt versucht die optimale Anordnung von Informationsseiten darzustellen.

### 3.2.1 Logo

Das Logo ist das Erkennungszeichen des Betreibers und der Webseite. Es wird in ca. 80% der Fälle an der oberen linken Ecke der Webseite platziert. Diese Position ist also der de-facto-Standard [4].

**Regel:** Die Unterkante des Logos muss über der Oberkante des Inhaltes und der vertikalen Navigation liegen. Wird eine horizontale Navigation umgesetzt so kann das Logo auch rechts der Navigation, vor allem aber über allen anderen Seitenelementen positioniert werden.

### 3.2.2 Navigation

Grundsätzlich können zwei Arten der Navigation anhand ihrer Anordnung unterschieden werden. Die horizontale oder Breitennavigation, bei der die einzelnen Menüpunkte nebeneinander angeordnet sind und die vertikale oder Tiefennavigation, Menüpunkte werden untereinander platziert. Weiters kann man zwischen Hauptmenü und Untermenü unterscheiden. Spool [10] stellt in seinen Studien fest, dass für die Anordnung des Hauptmenüs der obere horizontale Bereich einer Webseite zweckmäßig ist. Auch Adkins [11] kommt zu dem Schluss, dass der obere Rand einer Webseite am häufigsten für die Navigation verwendet wird. HHS [3] hingegen bezeichnet den linken Rand einer Webseite als geeigneten Bereich für die Umsetzung des Hauptmenüs sowie des Untermenüs. Nielsen [12] wiederum bezeichnet den rechten Seitenrand als besten Ort für die Navigation, da er sich nahe der Scrollbar befindet. Auf den meisten Webseiten lassen sich Haupt- sowie Submenü am oberen sowie linken Rand ausmachen.

**Regel:** Für die vertikale Navigation gilt, dass die rechte / linke Kante des Seitenbereiches links / rechts der linken / rechten Kante des Inhaltes und allen anderen Seitenbereichen liegen muss. Bei einer Überlappung der Seitenelemente darf die Breite der Navigation 100 Pixel nicht unterschreiten, da sonst nicht ausreichend Platz für die Navigation zur Verfügung steht. Es ist kein weiteres Seitenelement links der Navigation zulässig.

**Regel:** Für die horizontale Navigation gilt, dass die Unterkante über der Oberkante des Inhaltes liegen muss. Der Abstand der rechten Kante darf nicht weiter als 50 Pixel rechts der linken Kante des Inhaltes liegen. Die Mindesthöhe beträgt 20 Pixel.

Ferner dient die Navigation der Orientierung im Inhalt und als Überblick über die auf der Webseite vorhandenen Informationen. Sie sollte dem Benutzer zeigen wo er sich gerade



befindet. Der Link zur Homepage eines Webauftrittes sollte zusätzlich auf jeder der Unterseiten enthalten sein. Zur guten Orientierung auf einer Webseite müssen die Namen der Menüpunkte dem Benutzer Aufschluss über den zu erwartenden Inhalt geben.

### 3.2.3 Inhalt

Der Inhalt stellt das wichtigste Element einer Seite dar. Er sollte auf jeder Webseite dominieren, denn er stellt den Grund des Besuches dar. Fast jeder Benutzer scannt den Inhalt bevor er sich mit der Navigation auseinandersetzt. Nielsen [4] beschreibt Scannen als Überfliegen des Inhaltes. Dabei werden nur einige wenige Sätze und Wörter gelesen. Der Entwickler einer Webseite kann diesen Prozess beeinflussen und mit einfachen Designelementen wie z.B. Listen, bedeutungsvollen Überschriften und Hervorheben einzelner Schlüsselworte die Aufmerksamkeit gezielt lenken. Erscheint dem Benutzer der Inhalt als relevant, beschäftigt er sich mit der Navigation. Aus diesem Grund sollte der Inhalt so gestaltet sein, dass der Betrachter gleich erkennt worum es geht [4]. Kann der Benutzer keinen Inhalt ausmachen, besteht Gefahr, dass er die Seite wieder verlässt. Lynch und Horton [2] stellen fest, dass der Benutzer dem Inhalt „above the fold“ mehr Aufmerksamkeit schenkt. Gelingt es dem Entwickler, in diesem Bereich den Benutzer zu überzeugen, so ist die Wahrscheinlichkeit groß, dass er sich näher mit dem Webauftritt beschäftigt.

**Regel:** Die Breite des Inhaltes muss mindestens 250 Pixel betragen und muss in der ersten Hälfte der Webseite beginnen.

**Regel:** Die Oberkante des Inhaltes darf nicht weiter als 300 Pixel unter der Oberkante des Browserfensters beginnen.

### 3.2.4 Werbung

Die Finanzierung vieler Webseiten erfolgt durch Werbung. Auch wenn sie von den meisten Benutzern als „lästig“ empfunden wird, ist sie für einige Webauftritte überlebenswichtig. Die falsche Gestaltung von Werbeflächen kann sich jedoch negativ auf das Erscheinungsbild der Webseite auswirken. Werden verschieden große Werbebanner beim erneuten Laden einer Seite eingesetzt, verschiebt sich der Inhalt (GMX). Die drei negativsten Effekte der Werbung für den Benutzer sind Popups, die im Vordergrund erscheinen und den eigentlichen Inhalt überdecken, das langsame Laden der Seite aufgrund aufwendiger Werbebanner und der Versuch von versteckten Werbungen einen Klick des Nutzers zu ergattern [4]. Es besteht jedoch die Möglichkeit Werbung einzusetzen ohne den Benutzer abzuschrecken oder ihn zu „belästigen“. Der Einsatz von Textlinkwerbung ist sehr zurückhaltend und unterstützt den Benutzer eher bei der Suche nach Kaufoptionen. Nielsen [4] konstatiert, dass der Nutzer sogar danach sucht sofern Kaufabsicht besteht. Demnach zählt also der Inhalt der Werbung, weniger das Bemerkte [7]. Wichtig ist, dass die Werbung getrennt vom Inhalt und allen Seitenbereichen platziert ist. Nach Bernard [8] erwartet der Benutzer die Werbung am oberen Seitenrand. Eine ebenso weit verbreitete Position der Werbung ist der linke Seitenbereich.

**Regel:** Die Unterkante der Werbung muss über den Oberkanten, bzw. die linke Kante rechts der rechten Kanten aller weiteren Seitenbereiche liegen.

### 3.2.5 Suche

Neben der klassischen Navigation ist die Suchfunktion einer Seite die zweite Möglichkeit Informationen zu finden. Rund ein Drittel aller Benutzer versuchen über die Suchfunktion die gewünschten Informationen auf der Website zu finden [10]. Als de-facto-Standard für die Positionierung der Suchfunktion hat sich der rechte obere Seitenbereich, also der oberhalb des Inhaltes etabliert [9]. Bernard [8] bezeichnet den oberen mittleren Bereich als geeignete Position für das Suchfeld. Adkins [11] hingegen findet heraus, dass die häufigste Platzierung der Suchfunktion am rechten oberen Seitenbereich stattfindet.

**Regel:** Alle vier Kanten des Suchfeldes müssen innerhalb der horizontalen, vertikalen

linksbündigen oder rechtsbündigen Navigation liegen. Befindet sich das Suchfeld in einer der vertikalen Menüs, muss es im oberen Bereich liegen.

**Regel:** Liegt das Suchfeld nicht in den Navigationsbereichen, muss es rechts, links oder oberhalb des Inhaltes positioniert sein.

### 3.2.6 Zusätzlicher Inhalt / Informationen

In relevanten Studien findet dieser Seitenbereich jedoch kaum Erwähnung. Bei der Betrachtung einiger Websites fällt jedoch auf, dass zusätzlicher Inhalt für die Platzierung von Links zu näheren Informationen, News oder weiterführenden Themen verwendet wird. Ein weiterer möglicher Verwendungszweck ist das Angeben von Kontaktdaten oder Impressum in diesem Seitenelement, welches sich oft innerhalb des Inhaltes befindet.

**Regel:** Zusätzliche Informationen können innerhalb des Inhaltes rechtsbündig, wenn dort keine Navigation vorhanden, oder linksbündig, wenn keine Navigation auf der linken Seite, platziert werden. Ist dies der Fall, darf er eine Breite von 320 Pixel nicht überschreiten. Die verbleibende Breite des Inhaltes muss mindestens 250 Pixel betragen.

Für die Anordnung der Seitenelemente ergeben sich dadurch folgende Möglichkeiten.

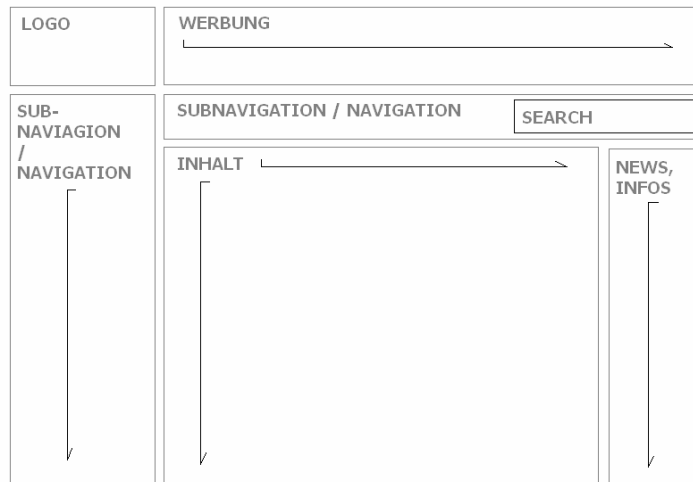


Abbildung 1: Beispiellayout 1 für die korrekte Anordnung der Seitenelemente.

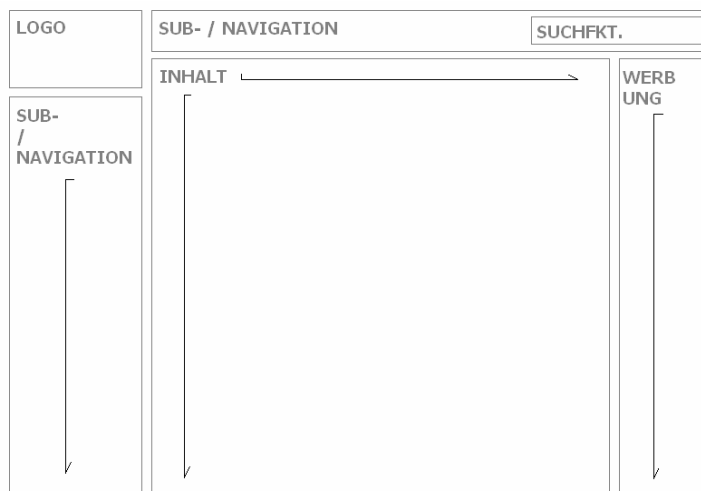


Abbildung 2: Beispiellayout 2 zur korrekten Anordnung der Seitenelemente.



Abbildung 3: Beispiellayout 2 zur korrekten Anordnung der Seitenelemente.

### 3.3 Größe der Seitenbereiche

Der Platz auf dem Bildschirm zur Darstellung einer Webseite ist begrenzt. Es werden Rund 19% der Gesamten Bildschirmfläche vom Betriebssystem (Taskleiste) oder dem Browser (Symbolleiste, Tabs) verwendet. Weitere 20% bleiben meist ungenutzt [4]. Es ist jedoch wichtig zu erwähnen, dass ungenutzte Bildschirmflächen nicht immer verschwendete Fläche sind. Ein gewisser Anteil an *white space* lockert das Design einer Seite auf und kann, richtig eingesetzt, die Lesbarkeit fördern [2]. Die folgenden prozentualen Angaben gelten für untergeordnete Informationsseiten. Bei Start oder Navigationsseiten ist es sinnvoll, den Inhalt weniger groß zu gestalten und mehr Platz für die Navigation zu verwenden.

#### 3.3.1 Inhalt

Das oberste Ziel einer Webseite ist es, dem Benutzer Informationen zur Verfügung zu stellen. Aus diesem Grund sollte der Inhalt den größten Seitenbereich einnehmen. Nach Nielsen sollte der Inhalt mindestens 50% des gesamten Seitendesigns ausmachen, besser jedoch 80% [4]. In den wenigsten Fällen kann er vollständig auf einer Browserseite angezeigt werden, häufiger als bei anderen Seitenelementen ist deshalb ein Scrollbar notwendig. Dabei muss darauf geachtet werden, dass dieser auch als solcher erkennbar sind. Der Scrollbar dient zusätzliche zur Navigation auch zur Orientierung auf langen Seiten. Ist der Inhalt jedoch zu lang findet sich der Nutzer nicht mehr zurecht und eine Aufteilung des Inhaltes auf mehrere Unterseiten, in Verbindung mit entsprechenden Navigationserweiterungen, ist notwendig.

**Regel:** Der Inhalt sollte mindestens 80% des Seitendesigns ausmachen und muss mehr als 50% betragen.

#### 3.3.2 Navigation

Als zweitwichtigstes Element nennt Nielsen die Navigation. Zwar bezeichnet er sie als notwendiges Übel, unterstreicht aber ihre Funktion als Hilfe zur Orientierung auf einer Webseite. Sie sollte jedoch bei untergeordneten Informationsseiten 20% der Bildschirmfläche nicht überschreiten [4].

**Regel:** Die Fläche der Navigation darf nicht mehr als 20% des gesamten Seitendesigns betragen.

### 3.3.3 Werbung

Fast alle Betrachter einer Webseite ignorieren Werbeflächen und Banner. Aus diesem Grund sollte sie nicht unnötig wertvolle Bildschirmfläche verschwenden. Häufig werden animierte Werbebanner eingesetzt, welche die Aufmerksamkeit des Benutzers vom Inhalt ablenken. Dies stellt einen Nachteil für Benutzer und Betreiber einer Webseite als dar. Für den Benutzer erscheint die Seite wenig seriös, wenn das als erstes bemerkte Seitenelement die Werbung ist. Die Gefahr, dass der Benutzer die Seite wieder verlässt, steigt.

**Regel:** Die Werbung darf nicht mehr als 20% der Fläche einer Webseite einnehmen.

### 3.4 Anpassungsfähigkeit des Seitenlayouts

Unter Anpassungsfähigkeit des Seitenlayouts versteht man die Fähigkeit einer Webseite, sich verschiedenen Bildschirmauflösungen so anzupassen, dass sie trotz möglicher Einschränkungen noch für den Nutzer verwendbar ist. Die Usability sollte über verschiedene Auflösungen hinweg erhalten bleiben. Nielsen [4] schlägt eine Optimierung der Seite auf eine Auflösung von 1024 x 768 Pixel vor, da rund 60 % aller Internetuser einen entsprechenden Bildschirm verwenden. 17 % der Nutzer jedoch verwenden immer noch einen Bildschirm der Auflösung 800 x 600 Pixel. Außerdem ist zu berücksichtigen, dass einige Benutzer etwaige Sidebars für Downloads oder Favoriten, verwenden, welche den Platz für die Webseite zusätzlich einschränken.

Des Weiteren könnte die Anpassungsfähigkeit von Webseiten auch für andere Endgeräte wie Handhelds oder Mobiltelefone überprüft werden.

Grundsätzlich kann eine Webseite auf zwei Arten den Anforderungen der Anpassungsfähigkeit gerecht werden. Sie kann flexibel umgesetzt werden, so dass den verwendeten Seitenelementen keine feste Breite zugewiesen wird, sondern sich die Webseite automatisch der verfügbaren Bildschirmfläche anpasst. Wird sie mit festen Breitenangaben realisiert, muss sie dem kleinsten Bildschirm, auf der die Seite angezeigt werden soll, entsprechend angepasst werden. Lynch und Horton legt die *graphic safe area* für Bildschirme mit einer Auflösung von 800 x 600 Pixel auf 760 x 410 Pixel fest.

**Regel:** Die Webseite muss in einem Browser unter der Auflösung 1024 x 768 Pixel, ohne horizontalen Scrollbar angezeigt werden können.

**Regel:** Die Webseite sollte in einem Browser unter der Auflösung 800 x 600 Pixel, ohne horizontalen Scrollbar angezeigt werden können.

## 4 Umsetzung

### 4.1 Allgemeines

Grundlage dieser Arbeit waren Richard Atterers [7] Untersuchungen und der von ihm entworfene Prototyp. Atterer implementierte die erste Version von WUSAB mit Java 1.4. Da zu Beginn der Weiterentwicklung eine neue Java-Version zur Verfügung stand, wurde auf Java 1.5 gewechselt. Als Container der Web-Applikation setzte er Tomcat 4.1 ein. Das Webinterface wurde mittels Java-Server-Pages (JSP) umgesetzt. Aufgrund der Tatsache, dass es sich hierbei um eine Webapplikation handelt und ein für die Entwicklung geeigneter Browser nötig war, entschied ich mich für den Mozilla Firefox 1.5. Als Entwicklungsumgebung verwendete ich NetBeans 4.1 mit Java 1.5.

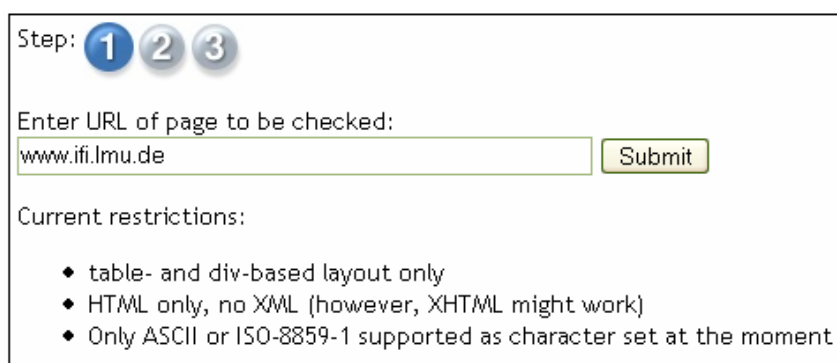
### 4.2 Arbeitsweise des Prototypen

Die URL der zu überprüfenden Webseite wird über ein Webinterface eingegeben. Der Benutzer muss die verschiedenen Seitenelemente annotieren. Der Server überprüft die eingegebene Seite auf die Usability des Layouts und gibt Erfolgs- und Fehlermeldungen aus. Die Eingabe der URL sowie das Parsen des Dokumentes waren Bestandteil des Prototyps von Richard Atterer. Ferner implementierte er den Mechanismus zum Annotieren von TABLE-basierten Layouts.

#### 4.2.1 Clientseitig

Der Benutzer muss bei zwei von drei Schritten Eingaben tätigen.

Der erste Schritt besteht aus der Eingabe der URL die überprüft werden soll:



The screenshot shows a web interface for Step 1 of the WUSAB process. At the top, there are three numbered steps: 1 (highlighted in blue), 2, and 3. Below this, the text 'Enter URL of page to be checked:' is followed by a text input field containing 'www.ifi.lmu.de' and a 'Submit' button. Underneath, the text 'Current restrictions:' is followed by a bulleted list of three items: 'table- and div-based layout only', 'HTML only, no XML (however, XHTML might work)', and 'Only ASCII or ISO-8859-1 supported as character set at the moment'.

Abbildung 4: Screenshot von WUSAB Schritt 1 - Eingabe der URL.

Im zweiten Schritt (Abbildung 5) wird der Benutzer aufgefordert, den jeweiligen Seitenbereichen einen entsprechenden Typen zuzuweisen:

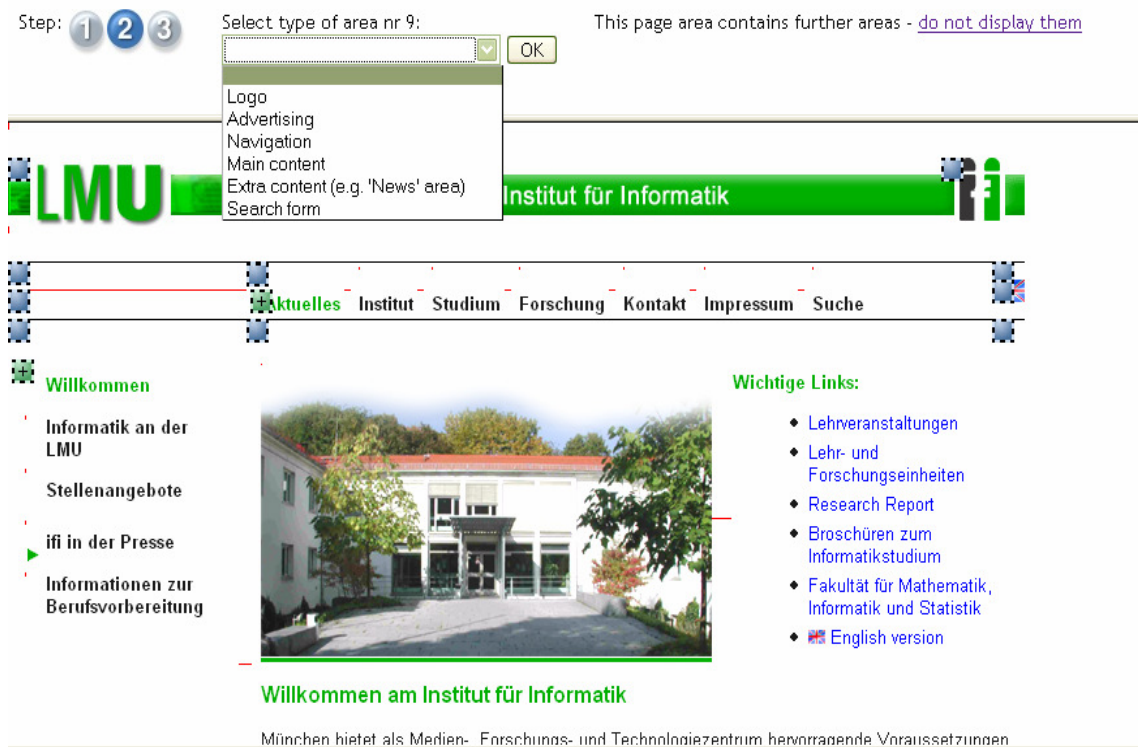


Abbildung 5: Screenshot von WUSAB Schritt 2 - Annotieren der Seitenbereiche.

Per Klick auf das „+“ oder das blaue Viereck wird der dazugehörige Seitenbereiche markiert. Zu Beginn sind alle oberen Elternelemente sichtbar. Will der User ein Kindelement annotieren, so muss er das Elternelement mit einem Klick auf „+“ expandieren. Im Drop-down-Menü mit dem Namen „Select type of area nr x“ kann er den entsprechenden Typen des markierten Seitenelements auswählen. Wurde dieser Seitenbereich bereits annotiert erscheint der ausgewählte Typ.

Hat der User alle Seitenbereiche markiert, kann er die Seite analysieren und auswerten lassen. Dazu muss er den Button „done print result“ klicken, der anstatt des OK Buttons erscheint. Die Fehler- und Erfolgsmeldungen werden ihm in drei Tabellen ausgegeben. Abbildung 6 zeigt die Ausgabe der Ergebnisse, die Ergebnisse der Größenanalyse:

Results of area size analysis	
RESULT	DESCRIPTION
WARNING:	Navigation areas are less than 30 % and greater than 20 %. Try to keep navigation areas as less as possible.
OK:	Content area is greater than 50 %.
OK:	Advertising areas are less than 20 %.

Abbildung 6: Screenshot von WUSAB Schritt 3: Ausgabe der Testergebnisse.

## 4.2.2 Serverseitig

Im ersten Schritt, nachdem der Benutzer die URL eingegeben und abgeschickt hat, speichert der Server die Seite als HTML-Dokument in der Session.

Im nächsten Schritt wird das Dokument für zwei verschiedene Zwecke geparkt (siehe 4.3). Für das Ermitteln der Positionsdaten werden dem ursprünglichen Quellcode des Dokuments IDs und ein Skript hinzugefügt (siehe 4.4). Für das Annotieren wird die Seite um einen Auswahlmechanismus erweitert, der dem Benutzer ermöglicht, den Seitenbereichen zu markieren.

Im letzten Schritt werden die ermittelten Positionsdaten und die Eingaben des Benutzers analysiert und überprüft. Fehler- und Erfolgsmeldungen werden generiert und ausgegeben. Das Sequenzdiagramm in Abbildung 7 zeigt den logischen und zeitlichen Ablauf des Tools.

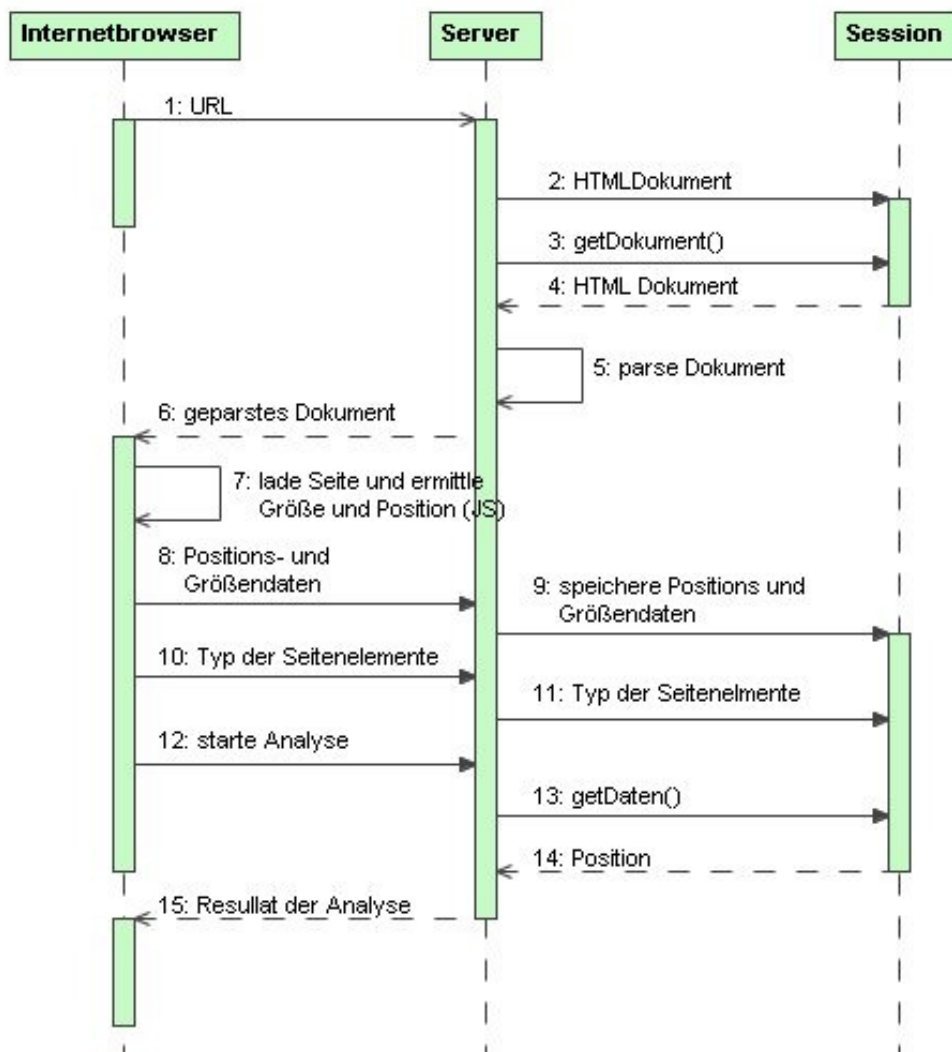


Abbildung 7: Sequenzdiagramm.

## 4.3 Parsen des Dokumentes

Im Allgemeinen ist ein Parser ein „Zerteiler“ der im vorliegenden Fall HTML-Dokumente

analysiert und sie, mithilfe einer Grammatik, in eine Datenstruktur (einen Parsebaum) überführt.

JavaCC (Java Compiler Compiler) ist der am häufigsten eingesetzte Generator von Parsern für die Verwendung in Java Applikationen. Ein Parsergenerator ist ein Tool, das eine spezifizierte Grammatik einliest und sie in ein Javaprogramm, den Parser, umwandelt. Das so erzeugte Programm erkennt Übereinstimmungen mit der Grammatik. JavaCC setzt JJTree ein. JJTree erlaubt das Erstellen von Datenstrukturen wie Parsbäume für den JavaCC [9].

Quietix, der für diese Arbeit verwendete, kostenlose Parser, ist ein auf JavaCC basierender HTML-Parser der mit Visitor-Pattern arbeitet. Visitor-Pattern dienen der Kapselung von Operationen die auf Elemente einer Datenstruktur, die im Vergleich zu Objekten „dumm“ sind, angewandt werden können. Mit dieser Technik wird ermöglicht, Operationen für eine Datenstruktur zu ändern, ohne die Klassen der Elemente zu modifizieren. Die Verwendung von Visitor-Pattern erlaubt die Trennung von Klassen für Elemente der Datenstruktur und dem Algorithmus über selbige. Des Weiteren können verschiedene Visitors auf eine Datenstruktur angewandt werden, wie es für diese Arbeit nötig war. Die Datenstruktur, welche der Visitor durchlaufen soll, muss ihn dazu akzeptieren (*accept(Visitor)*). Jedes Element muss wiederum den Visitor akzeptieren um sicherzustellen, dass auch alle Kindelemente durchlaufen werden. Im Visitor selbst sind Operationen für die Knoten oder Elemente der Datenstruktur definiert, die er mit der Methode *visit(element)* ausführt.

Das vom Quietix HTML-Parser erzeugte HTML-Dokument, gespeichert in der Session, wird von zwei Visitors durchlaufen.

Zum einen wird es vom *HtmlVisitor PositionSetter.java* (siehe 4.7) zur Ermittlung der Positionsdaten um IDs und ein Skript erweitert. Dazu werden HTML-Tags um den Klassennamen *wusab\_id\_{ID}* ergänzt und ermöglichen somit die spätere Identifizierung durch das Skript. Falls der HTML-Tag schon ein oder mehrere Klassenattribute besitzt, wird der zu erweiternde Klassenname angehängt. Damit ist gewährleistet, dass alle Klassenattribute berücksichtigt werden und keine Änderung des Layouts (CSS-Klassen) stattfindet. Da in dieser Arbeit nur DIV- und TABLE-Layouts zu analysieren sind, beschränkt sich dieser Vorgang auf `<div>`-, `<table>`- und `<td>`-Tags. Des Weiteren werden die relevanten HTML-Tags als Objekte in einer Session gespeichert. Außerdem wird das Dokument um einen `<Skript>`-Tag zur Ermittlung der Positionsdaten erweitert.

Zum anderen wird, in Form von Icons, der Auswahlmechanismus zum Annotieren der Seitenbereiche durch den Benutzer am Anfang eines jeden relevanten Tags platziert. Dies geschieht durch den *HtmlVisitor RegionSelDumper.java* (siehe 4.7). Durchläuft er das Dokument zum ersten Mal, werden nur die obersten Elternelemente, wieder beschränkt auf `<div>`-, `<table>`- und `<td>`-Tags, berücksichtigt. Expandiert der Benutzer einen Seitenbereich, durchläuft er das Dokument von neuem und erweitert wieder nur die obersten Kindknoten mit den für die Selektion von Seitenelementen vorgesehenen Icons. Wird ein Element wieder reduziert, wird auch der Auswahlmechanismus aller Kindelemente ausgeblendet. Dadurch kann der Benutzer jeden relevanten Seitenbereich markieren und annotieren. Da zu diesem Zeitpunkt die Größen der Seitenbereiche bereits bekannt sind, werden Elemente mit einer Breite oder Höhe von kleiner als 10 Pixel nicht berücksichtigt. Es wird davon ausgegangen, dass sie aufgrund ihrer Größe keine wichtige Rolle im Layout einnehmen und somit nicht annotiert werden müssen.

#### **4.4 Erhebung der Positionsdaten**

Die Positions- und Größendaten der Seitenbereiche spielen, neben dem Verwendungszweck der Seitenelemente, eine zentrale Rolle für die Analyse der Usability des Layouts. Positions- und Größenangaben können, im Gegensatz zum Typen der Seitenbereiche, automatisch ermittelt werden.

Zu Beginn dieser Projektarbeit wurde versucht, anhand der HTML-Struktur der zu untersuchenden Webseite auf die Größe und Position der einzelnen Seitenbereiche zu schließen.



Es stellte sich jedoch heraus, dass dies kaum möglich ist. Man würde einen browserähnlichen Interpreter des Quellcodes benötigen, welcher die genauen Positionen und Größen der Elemente ermitteln kann. Die Umsetzung hätte jedoch den Rahmen dieser Projektarbeit gesprengt.

Alternativ dazu kann die Webseite um ein Skript erweitert werden. Javascript eignet sich für die Arbeit mit HTML-Dokumenten am besten, da es auf die DOM Struktur und Style-Attribute der einzelnen Elemente zugreifen kann. Dazu muss die Seite mit dem angehängten JavaScript (siehe 4.3) in einem Browser geladen und dargestellt werden. Unter Berücksichtigung der Tatsache, dass dieses Werkzeug auch mit Modellen externer Tools funktionieren soll, musste dies auf zwei Arten umgesetzt werden.

Zum einen kann das Tool serverseitig einen Browser öffnen, indem der modifizierte Quellcode eines Modells angezeigt und geladen wird. Dazu muss der Quellcode in HTML-Format mit den vorgesehenen Größen der Seitenbereiche übergeben werden. Ferner muss das Modell Rückschlüsse auf den Verwendungszweck der Seitenbereiche zulassen. Dafür kann es beispielsweise dem Server bekannte Klassennamen oder IDs verwenden.

Wird WUSAB über das Webinterface bedient, wird die zur Ermittlung der Positionsdaten geparste und modifizierte Seite im Browser des Benutzers geladen. Dies geschieht in einem, für den Benutzer unsichtbaren Frame. Die Daten werden dann mittels *XMLHttpRequest* an den Server geschickt und in einer Session gespeichert.

Mittels des *onLoad* Events wird in beiden Fällen die JavaScript Funktion *onloadGetPos(url, servletUrl)* aufgerufen. Der Parameter *URL* dient zur Identifizierung der Session. *ServletUrl* gibt Aufschluss über den Ort der Servlets. *onloadGetPos()* durchläuft alle Elemente der Seite. Wird ein vom Parser gesetzter Klassenname *wusab\_id\_{ID}* gefunden, werden mit den JavaScript Methoden *offsetLeft* und *offsetTop* die Abstände der Elemente zum oberen bzw. linken Rand des Elternelementes ermittelt. Rekursiv werden die Abstände aller Elternelemente aufsummiert bis kein weiteres Element mehr gefunden wird, um so die relative Position zu erhalten. Die Attribute *width* und *height* liefern die Größenangaben der Seitenelemente. Diese Werte werden in einer Variablen gespeichert und an den Server geschickt.

Die Daten werden mittels der POST-Methode an den Server gesendet, da die GET-Methode nur eingeschränkte Datenlängen zulässt. Die JSP-Seite *addPosition.jsp* zerlegt die empfangenen Daten und speichert sie in den entsprechenden Objekten.

## 4.5 Konfiguration des Prototypen

Über die Positions- und Größenangaben hinaus werden zur Implementierung des Regelwerkes Grenzwerte benötigt. Diese Grenzwerte beschreiben die maximalen bzw. minimalen Ausmaße von Seitenbereichen. Dies geschieht durch die Konfiguration des Prototyps über eine XML-Datei. Abbildung 7 zeigt welche Werte konfigurierbar sind. Die Grenzwerte gelten für nachstehende Seitenelemente:

- Y4: Maximale Distanz der Oberkante des Inhaltes zur Seitenoberkante.
- Y3: Mindesthöhe der horizontalen Navigation, falls Überlappung mit anderen Seitenelementen vorhanden.
- YTOL: Toleranz bei Überlappung. Wird bei jedem Seitenbereich berücksichtigt.
- X2: Mindestbreite der vertikalen Navigation.
- X4: Mindestbreite des Inhaltes.
- X5: Maximale Breite des Bereiches für zusätzlichen Inhalt.

Außerdem können die Angaben der zulässigen Seitengrößen, in Prozent, und der Grenzwert für die maximale Breite für die Überprüfung der Anpassungsfähigkeit festgelegt werden.

Die konfigurierbaren Werte beschränken sich auf die wichtigsten Seitenbereiche. Sie decken

nicht alle Sonderfälle ab und dienen eher der Feststellung der Eignung dieses Verfahrens. Des Weiteren werden für Modelle keine Sonderregelungen benötigt, da bei der Erstellung auf die überlappungsfreie Anordnung geachtet werden kann.

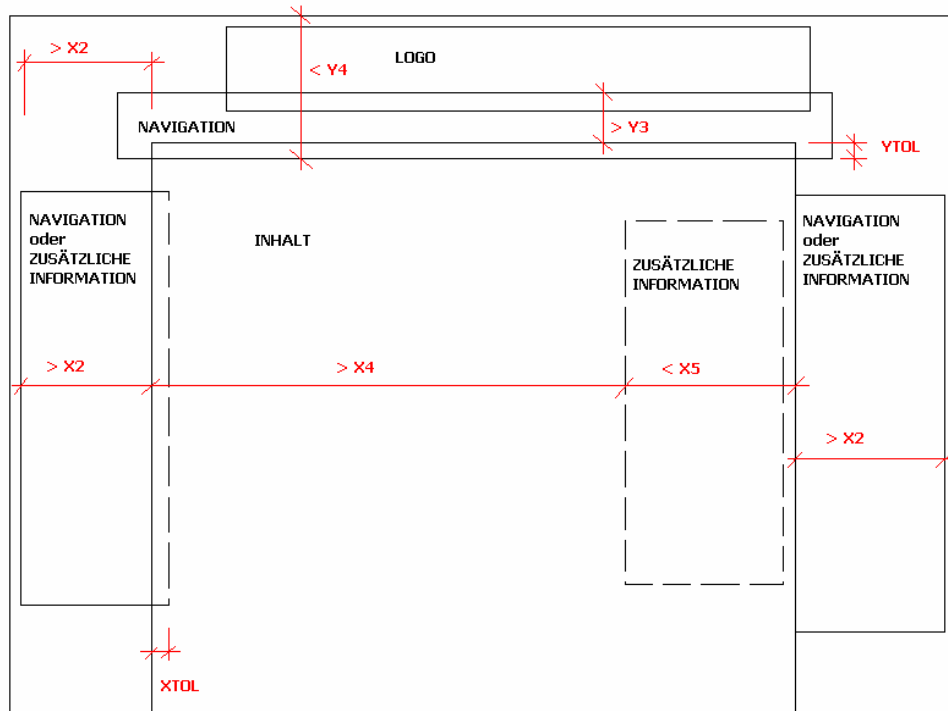


Abbildung 7: Konfigurierbare Grenzwerte.

## 4.6 Auswertung der Daten

Wurden alle für die Überprüfung der Seite notwendigen Daten gesammelt werden Tests zur Ermittlung der Usability durchgeführt und Meldungen für den Benutzer generiert. Es werden drei Arten von Meldungen unterschieden:

- OK: Entspricht dem de-facto-Standard.
- WARNING: Entspricht zwar nicht dem Standard, kann aber auch nicht als Fehlgestaltung gewertet werden und bedarf der Überprüfung durch den Benutzer.
- ERROR: Wurde als Fehlgestaltung erkannt und sollte geändert werden.

Insgesamt werden drei Tests durchgeführt die im Anschluss genauer beschrieben werden.

### 4.6.1 Testen der Anordnung der Seitenelemente

In der Session sind alle Elemente als Objekte (*PageArea*) gespeichert, die als Seitenbereich in Frage kommen könnten. Es wurden für jedes Element Position und Größe ermittelt. Es ist jedoch wahrscheinlich, dass nicht jeder Seitenbereich genutzt bzw. annotiert wird. Deswegen werden alle in der Session gespeicherten Objekte durchlaufen und nur relevante, die ein Attribut für den Verwendungszweck besitzen, in einem Vektor der Testklasse, falls mehr als ein Bereich dieses Typs zulässig ist, gesammelt. Ist nur ein Seitenbereich erlaubt so wird dieses Objekt gespeichert.

Der Inhalt stellt den Bezugspunkt der Tests dar. Er ist, neben mindestens einem Navigationsbereich, ein obligatorisches Seitenelement. Wird keine Navigation oder Inhalt

gefunden, kann die Analyse des Layouts nicht durchgeführt werden. Es sind ein Inhalts-, drei Navigations-, ein Such-, ein Logo- und beliebig viele Werbebereiche, sowie Bereiche für den zusätzlichen Inhalt zulässig.

Da alle Seitenelemente relativ zum Inhalt überprüft werden, wird für ihn nur getestet, ob er im oberen Seitenbereich beginnt.

Für die Überprüfung der Navigation wird der mit den Seitenbereichen dieses Typs gefüllte Vektor durchlaufen. Jeder Navigationsbereich wird überprüft ob er über bzw. links des Inhaltes liegt. Ist dies der Fall, wird dem Benutzer eine OK-Meldung ausgegeben, die das Layout bestätigt. Wird die Navigation rechts des Inhaltes angeordnet, wird eine Warnung zurückgegeben. Bei allen anderen Platzierungen wird das Layout als Fehlgestaltung bewertet. Anbei das Aktivitätsdiagramm:

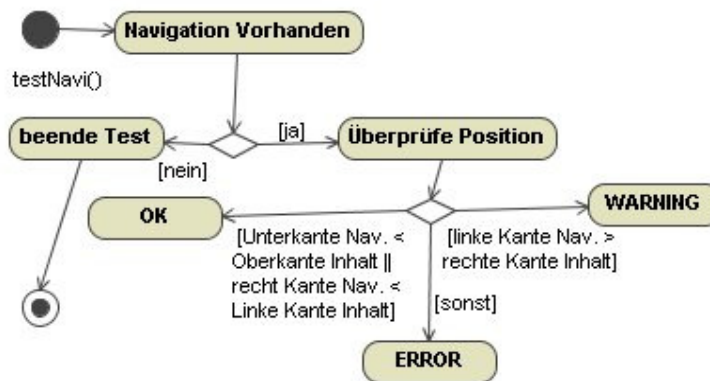


Abbildung 8: Aktivitätsdiagramm der Überprüfung der Navigation.

Bei DIV-Layouts kann es jedoch vorkommen, dass sich Seitenbereiche überschneiden und trotzdem richtig angeordnet sind. Deshalb wird eine Toleranz von 10 Pixel im Falle einer Überlappung der Seitenbereiche mit eingerechnet (z.B. Position der rechten Navigationskante - 10 Pixel < Position der linken Kante des Inhaltes). Ferner wurden für einige Seitenbereiche Ausnahmeregelungen implementiert (siehe 4.5).

Für das Logo gilt, dass es über allen anderen Seitenbereichen liegen muss, mit Ausnahme der Navigation und des Werbebereichs. Trifft das zu, so wird eine OK-Meldung ausgegeben. Bei allen anderen Anordnungen wird eine Fehlanordnung gemeldet. Für das Logo sind keine Sonderfälle implementiert worden.

Der Werbebereich der Seite muss über bzw. rechts aller anderen Seitenbereiche liegen. Wird er an der Oberkante des Layouts angeordnet, kann das Logo links der Werbung angesiedelt sein. Alle anderen Anordnungen werden als Fehler betrachtet.

Zusätzlicher Inhalt kann rechts oder links des eigentlichen Inhaltes platziert werden, wenn keine Navigation auf der Seite umgesetzt wurde. Des Weiteren ist er, neben der Suchfunktion, das einzige Element das sich innerhalb des Inhaltes befinden darf. Es muss jedoch darauf geachtet werden, dass genügend Platz für den eigentlichen Inhalt zur Verfügung steht.

Die Suchfunktion sollte entlang des oberen Bereiches des Layouts realisiert werden. Sie kann sich auch innerhalb eines Navigationsbereichs befinden. Alle anderen Anordnungen sind ungültig.

## 4.6.2 Testen der Größe der Seitenelemente

Für den Größentest der Seitenelemente werden in der initialen Phase, wie beim Test der Anordnung, alle Objekte gesammelt, die ein Attribut für den Verwendungszweck des Seitenbereiches besitzen. Für diesen Test werden jedoch nur Navigation, Inhalt und Werbung

beachtet. Die Flächen dieser Seite werden addiert und der prozentuale Anteil jedes einzelnen berechnet.

Ist der Inhalt kleiner 50% wird eine Warnmeldung zurückgegeben. Da nicht ausgeschlossen werden kann, dass es sich bei der zu überprüfenden Seite um eine Startseite handelt, wird erst bei 30% ein Fehler festgestellt. Alles andere wird als Fehlgestaltung gewertet. Beträgt der Werbereich der Seite mehr als 20% wird ein Fehler ausgegeben. Bei einer Navigationsfläche von 20 bis 30% wird der Benutzer gewarnt. Beträgt sie mehr als 30% wird ein Fehler zurückgegeben.

### 4.6.3 Testen der Anpassungsfähigkeit der Webseite

Um die Werte zur Überprüfung der Anpassungsfähigkeit einer Seite zu erhalten, wird die Seite von *onloadGetPos()* drei Mal durchlaufen. Nach jedem Durchlauf wird die Größe des Frames verändert, in dem die Seite geladen wird. Beim ersten Mal werden die Positionen und Größe der Elemente für die Auflösung des clientseitigen Bildschirms überprüft. Im zweiten Durchgang wird die Breite des Frames auf 1024 Pixel verändert und im dritten auf 800 Pixel. Nachdem alle Daten erhoben wurden, schickt das Skript die Daten an den Server, der sie in den betreffenden Objekten in der Session speichert.

## 4.7 Architektur und Klassen

Der Prototyp besteht aus einem Container für Java-basierte Web-Anwendungen (Tomcat) zur Ausführung von JSP-Seiten und Java-Klassen. Welche JSP-Seiten verwendet werden und wie sie zusammenhängen zeigt das Webdiagramm in Abbildung 9. Ein Diagramm der wichtigsten Klassen präsentiert Abbildung 10.

Im Anschluss wird beschrieben, welche JSP-Seiten die verschiedenen Java-Klassen verwenden.

Nachdem die URL der Webseite an die JSP-Seite *annotate.jsp* geschickt wurde, baut diese eine *HttpConnection* zur URL auf und übergibt den so erhaltenen *InputStream* dem Parser. Ein Visitor gibt dem Dokument eine Struktur und wandelt die HTML-Elemente in Objekte um. Das so veränderte Dokument wird in der *PageSession* gespeichert. Ist dieser Vorgang abgeschlossen werden drei Seiten in verschiedenen Framset geladen.

Im obersten, für den Benutzer unsichtbaren Frame wird die Seite *positionSetter.jsp* angezeigt. Sie ist für die Ermittlung der Größen- und Positionsdaten zuständig. Unter Verwendung des Visitors *PositionSetter.java* wird das Dokument zur Größen- und Positionsermittlung angepasst. Das Skript ermittelt die Werte und sendet sie an den Server der diese im *PageSession*-Objekt speichert (siehe 4.4). Ferner ist *PositionSetter.java* für die Speicherung der relevanten HTML-Elemente als *PageArea*-Objekte in der *PageSession* zuständig. Die Klasse *PageArea.java* enthält verschiedene Operationen und Variablen zum Speichern der Positions- und Größenangaben sowie der Bereichstypen der Seitenelemente. Dazu werden sie in der *PageSession* in einem Vektor, der den Zugriff auf die *PageArea*-Objekte mittels ID erlaubt, und einem Hashtable, welcher den Zugriff mittels des HTML-Objektes zulässt, abgelegt. Zur Speicherung der verschiedenen Größen und Positionen bei unterschiedlichen Auflösungen legt *PageSession.java* zwei weitere Kopien der *PageArea*-Objekte in Vektoren und Hashtables an.

*addPosition.jsp* übernimmt die Speicherung der Positionsdaten nachdem sie vom Skript empfangen wurden. Für die drei zu überprüfenden Auflösungen wird jeweils eine Variable mit den entsprechenden Positions- und Größenangaben übermittelt und im passenden *PageArea*-Objekt abgelegt.

Im zweiten Frame wird *regionsprop.jsp* angezeigt. Diese Seite ist für die Annotierung durch den Benutzer verantwortlich. Dazu ist sie auf Werte von *regions.jsp* angewiesen, die im dritten Frame angezeigt wird. *regions.jsp* unterstützt die Selektion der Seitenbereiche. Dafür wird das in der *PageSession* gespeicherte Dokument vom Visitor *RegionSelDumper.java* um den Auswahlmechanismus erweitert. Er platziert in jedem expandierten `<div>`-, `<table>`- oder `<td>`-

Tag ein Icon zur Selektion. Klickt der Benutzer auf eines dieser Icons wird ein *onclick*-Event ausgelöst und mit *parent.frames['regionprop'].location.replace(url?id={ID})* dem zweiten Frame mit der *regionsprop.jsp* Seite, die ID des selektierten Seitenelementes geschickt. *regionsprop.jsp* zeigt darauf hin ein Drop-down-Menü an, mittels dem der Benutzer dem Seitenbereich einen Typen zuweisen kann, sofern er die Auswahl mit dem OK-Button bestätigt. Die Zuweisung des Typen an ein *PageArea*-Objekt übernimmt ebenso *regionsprop.jsp*, da es ID und Typ der *PageArea* kennt. Wurden die Seitenelemente annotiert und bestätigt kann die Auswertung der Daten stattfinden.

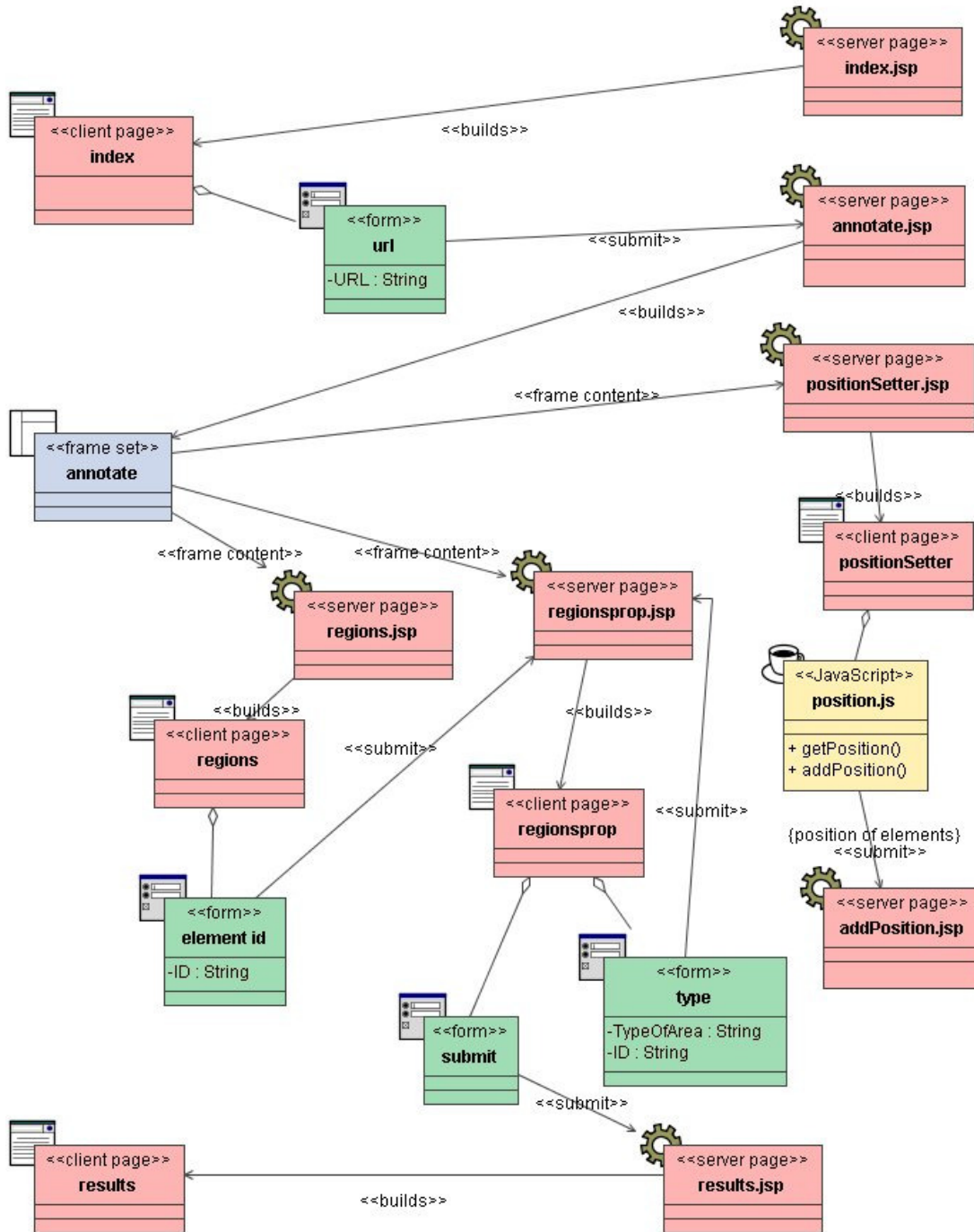


Abbildung 9: Webdiagramm des Prototypen.

Dazu wird *results.jsp* aufgerufen. Sie enthält die durchzuführenden Test. Bisher wurden drei implementiert: *LayoutPositionTest.java*, *ScalabilityTest.java* und *AreaSizeTest.java*. Um für die Ausgabe und den Zugriff auf die Ergebnisse der Tests eine einheitliche Schnittstelle zu erhalten, implementieren alle Tests das Interface *UsabilityTest.java*.

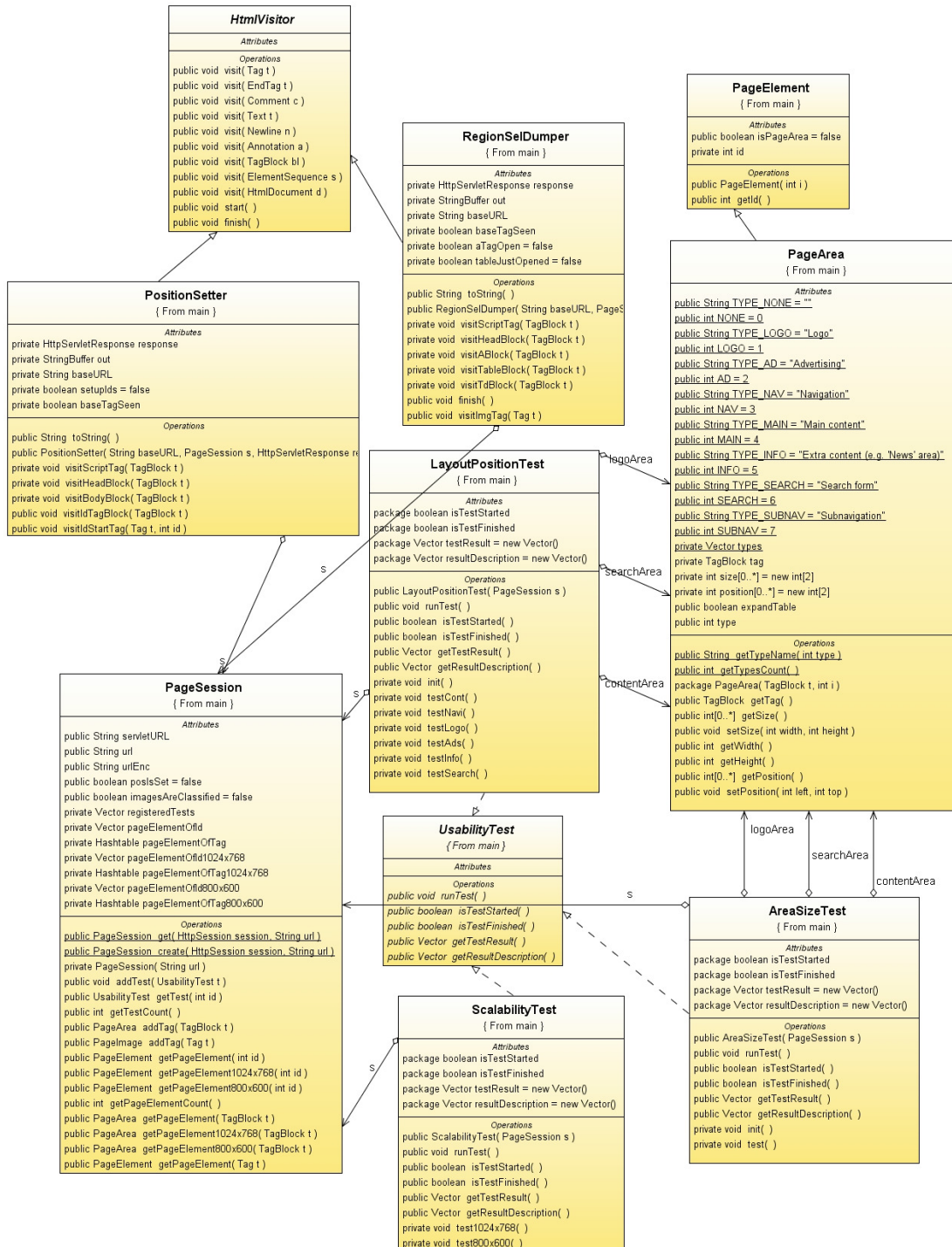


Abbildung 10: Klassendiagramm des Prototypen.

Bei der Initialisierung wird jedem Test die *PageSession* übergeben. Somit können sie auf die *PageArea*-Objekte zugreifen und diese überprüfen.

Die generierten Fehlermeldungen und Beschreibungen werden in zwei Vektoren gespeichert. Zur Visualisierung der Meldungen durchläuft *results.jsp*, nach Abarbeitung der Tests, die zwei Vektoren und gibt sie dem Benutzer in Tabellenform, für jeden Test separat, aus.

Das oben gezeigte Klassendiagramm repräsentiert Klassen aus zwei Paketen, dem *main*- und *parser*-Paket. Die Klassen des *main*-Paketes sind durch *{from main}* unter dem Klassennamen zu erkennen. Alle Anderen sind aus dem *parser*-Paket.

Weitere zum Parsen der Webseite verwendete Klassen sind *HtmlParser.java*, eine von JavaCC erzeugte Klasse die mit Hilfe der in *HtmlParser.jj* definierten Grammatik einen flachen Parsbaum des HTML-Dokumentes erzeugt. *HtmlDocument.java* speichert diese als Sequenz von HTML-Elementen. Der Visitor *HtmlCollector.java* gibt diesem Parsbaum eine Struktur. Er weißt beispielsweise jedem Starttag einen Endtag zu.

Ferner werden noch einige Klassen im *main*-Paket verwendet, welche keine Erwähnung im Klassendiagramm finden. Dazu gehört der *Conf.java* der unter Verwendung der Klasse *XMLFileReader.java* die Konfigurationsdatei einliest und die dort festgelegten Werte als Konstanten speichert. Die Klasse *Download.java* baut eine *HttpConnection* zur gewünschten URL auf. Sie dient dem Download des Quellcodes.





## 5 Testen des Prototypen

Zur Feststellung der Tauglichkeit von WUSAB wurde das Layout verschiedener Webseiten getestet. Es wurde darauf geachtet, DIV- sowie TABLE- Layouts zu überprüfen. Im Folgenden werden die Ergebnisse von drei Webauftritten vorgestellt.

Die Webseite der Embedded Interaction Group ist eine statische Informationsseite und wurde mit einem TABLE-Layout umgesetzt. Sie ist eine der wenigen Webauftritte mit zwei Navigationsleisten auf der linken Seite. Ferner enthält die Startseite einen relativ großen News-Bereich. Abbildung 11 zeigt welche Seitenbereiche annotiert wurden.

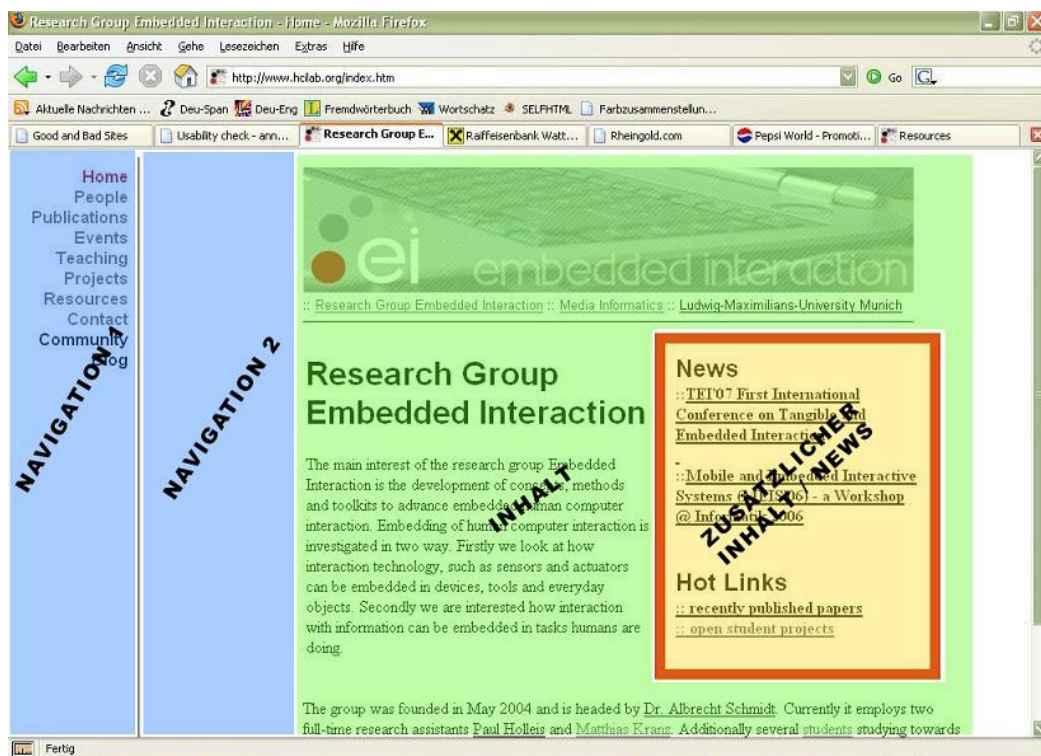


Abbildung 11: Annotierung der Seite <http://www.hcilab.org>.

Zur Annotierung ist zu sagen, dass der zweite Navigationsbereich von links auf der Homepage des Webauftrittes nicht genutzt wird. Im weiteren Verlauf ist jedoch die Subnavigation darin zu finden und wurde deshalb als Navigation definiert.

Das Ergebnis einer Überprüfung auf die korrekte Anordnung der Seitenelemente ergab eine Bestätigung des Layouts. Nur der Hinweis auf die Abwesenheit des Logos, welches sich nicht in einem eigenen Bereich befindet und somit nicht annotiert werden konnte, wurde in Form einer Warnung ausgegeben.

Die Auswertung der Größenverteilung der Seitenbereiche ergab, dass die Fläche der Navigation zwischen 20 und 30% beträgt und deshalb mit einer Warnung zu bewerten ist. Wie Abbildung 11 zeigt, wird zwar nicht der gesamte annotierte Bereich für die Navigation benutzt, ist jedoch für die Navigation reserviert und kann somit von keinem anderen Seitenbereich verwendet werden. Hier stellt sich die Frage, welche Größenangaben für die Flächenermittlung verwendet werden. In dieser Arbeit werden die tatsächlichen Größen der Seitenbereiche zur Berechnung der Fläche verwendet, da theoretisch die Möglichkeit besteht, den ganzen Bereich für die Navigation zu nutzen. Des Weiteren konnte die Fläche des Logos nicht von der Inhaltsfläche abgezogen werden. Dies stellt eine Verzerrung des Ergebnisses dar und könnte durch eine Erweiterung des Auswahlbereiches auf Bilder oder Grafiken behoben werden.

Bei der Ermittlung der Anpassungsfähigkeit der Seitenelemente wurde festgestellt, dass sich das Layout nicht für eine Auflösung von 800 x 600 Pixel eignet. Will man den gesamten Inhalt der Seite lesen muss man den horizontalen Scrollbar betätigen. Dies ist aus Sicht der Benutzerfreundlichkeit nicht zulässig und wird mit einem Fehler bewertet.

Als nächste Seite wurde wetter.at mittels WUSAB analysiert. Sie ist ein Beispiel für die Analyse von DIV-Layouts. Abbildung 12 zeigt die Annotierung der Seite:

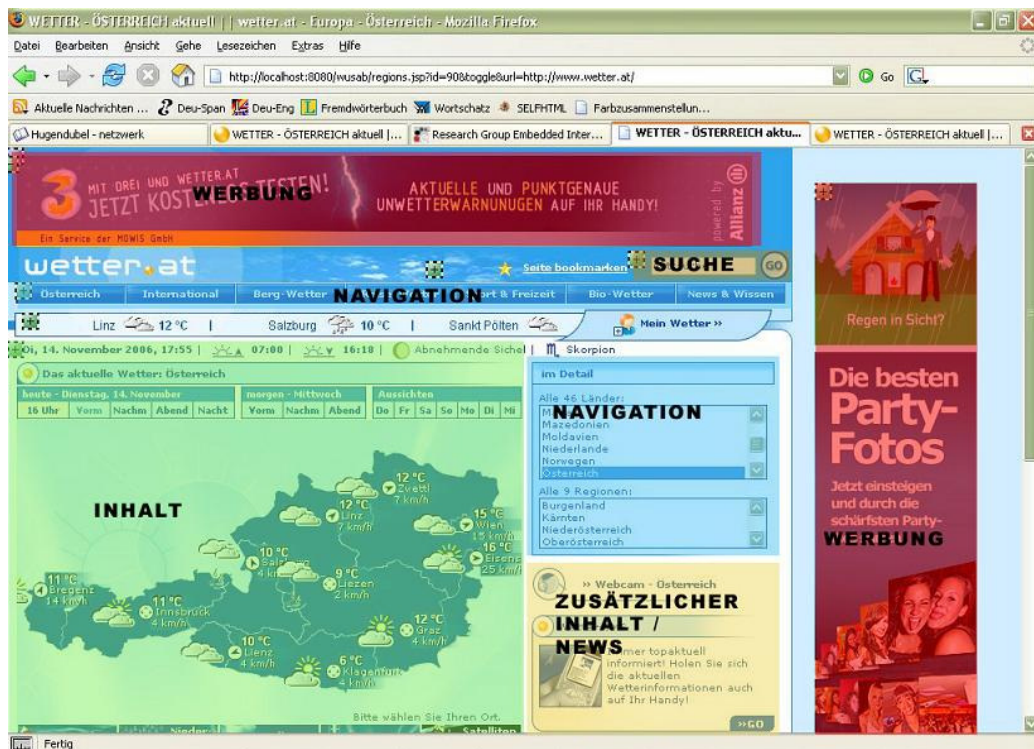


Abbildung 12: Annotierung von wetter.at.

Der Anordnungstest der Seitenelemente ergab zwei Warn- und eine Fehlermeldung. Die Navigation rechts des Inhaltes anzuordnen ist nicht üblich, deshalb wird der Benutzer darauf hingewiesen. Ferner konnte auch hier das Logo nicht annotiert werden, da es mit der Werbung in einem Seitenbereich liegt. Auch darauf wird der Benutzer mit einer Warnung hingewiesen. Den zusätzlichen Inhalt rechts unter der Navigation anzuordnen stuft WUSAB noch als Fehler ein. Da diese Position sehr häufig für zusätzlichen Inhalt eingesetzt wird, sollte eine Regel für die Bestätigung dieser Anordnung implementiert werden.

Der Test der Größenverteilung wurde ohne Warnung oder Fehlermeldung bestanden. Es fällt

jedoch auf, dass im Vergleich zum vorherigen Test weniger Inhalt auf der Seite zu sehen ist. Die positive Bewertung der Analyse ist darauf zurückzuführen, dass auch der nicht sichtbare Inhalt in die Flächenberechnung mit einbezogen wird. Richtig wäre die Überprüfung des sichtbaren Inhalts. Die Umsetzung der entsprechenden Regel wird von einer Auflösung von 1024 x 768 Pixel ausgehen, da Webseiten für diese Auflösung optimiert werden sollten, und nur den tatsächlich sichtbaren Seitenbereich berücksichtigen.

Die Überprüfung der Anpassungsfähigkeit ergab eine Bestätigung für eine Auflösung von 1024 x 768 Pixel. Dagegen wurden ein Fehler für 800 x 600 Pixel ausgegeben. Wird dies kontrolliert, kann festgestellt werden, dass zwar auch mit dieser Auflösung die gesamte Seite sichtbar ist, der rechte Werbebereich aber aus dem Blickfeld geschoben wurde. Die Fehlermeldung begründet sich dadurch, dass ein horizontaler Scrollbar erscheint. Der Benutzer könnte den Eindruck haben, dass mehr Inhalt vorhanden ist, als er angezeigt bekommt. Bedient er deshalb den Scrollbar, bekommt er „nur“ Werbung zu sehen.

Die Seite des Online-Marketing-Unternehmens [textlinkbrokers.com](http://www.textlinkbrokers.com) wurde mittels DIV- und TABLE-Tags umgesetzt. Die Seitenbereiche wurden wie folgt annotiert:

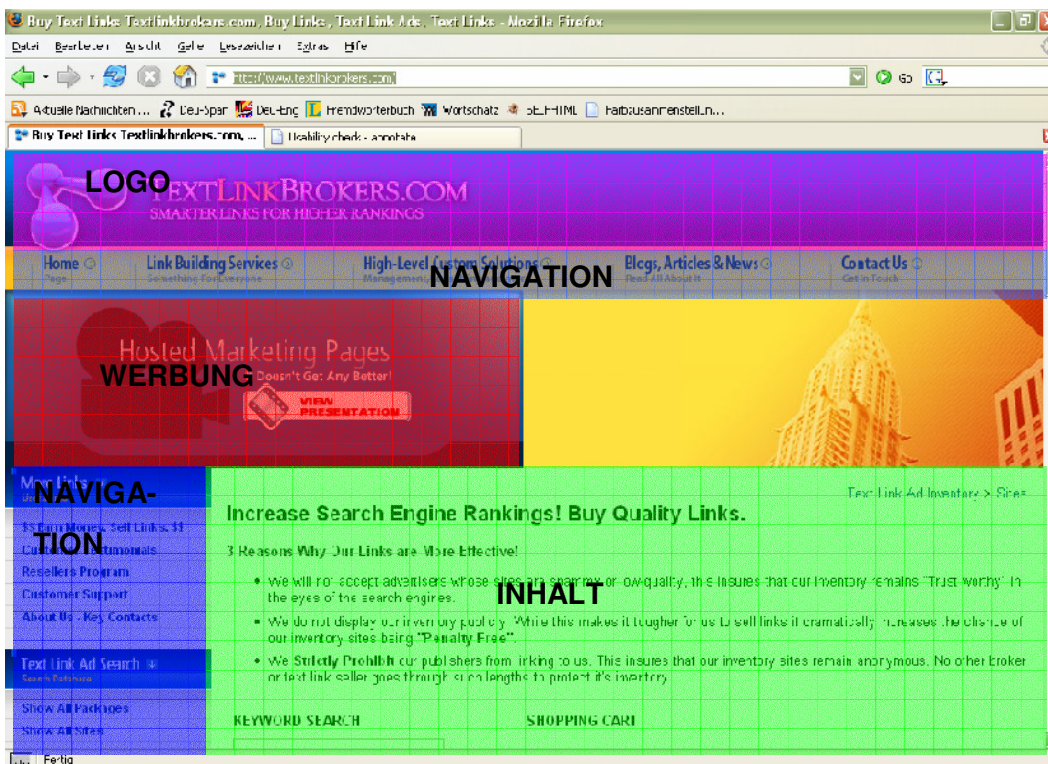


Abbildung 13: Annotierung von [www.textlinkbrokers.com](http://www.textlinkbrokers.com).

Zusätzlich zur gezeigten Annotierung wurde die Suchfunktion unterhalb der linken Tiefennavigation markiert. Bei der Werbung handelt es sich in diesem Fall um Eigenwerbung. Sie besteht aus einer bewegten Flashanimation, die dem Benutzer die Firmenphilosophie vermitteln soll. Für den Benutzer erscheint sie dennoch als Werbung und wurde deshalb als solche annotiert.

Die Analyse dieser Seite ergab zwei Fehlanordnungen der Seitenelemente. Der Inhalt beginnt erst in der zweiten Hälfte der Seite. Dadurch wird verhältnismäßig viel Zeit benötigt um den Inhalt zu finden. Außerdem befindet sich die Werbung zwischen Navigation und Inhalt. Auch

dies wurde als Fehler bewertet und mit einer entsprechenden Meldung versehen. Die Suchfunktion befindet sich zwar innerhalb der linken Tiefennavigation, kann aber nicht auf Anhieb gefunden werden. Der Nutzer muss vertikal scrollen um das Suchfeld zu finden. Da ein Drittel der Benutzer die Suche zur Navigation verwenden, sollte sie im sichtbaren Bereich der Seite angesiedelt werden. Diesen Aspekt berücksichtigt WUSAB noch nicht, wird aber um eine entsprechende Regel ergänzt werden.

Die Überprüfung der Anpassungsfähigkeit und der Größenverteilung der Seitenelemente fiel positiv aus. Jedoch fällt auch bei diesem Beispiel auf, dass die Flächenverteilung nicht dem Ideal entspricht. Wird aber, wie oben erwähnt, im Regelwerk berücksichtigt werden.

## 6 Resümee

Abgesehen vom Sammeln der Daten führt WUSAB alle Schritte der Evaluierung bis hin zur Auswertung automatisch durch. Wie gezeigt wurde, beschränkt sich das Tool auf die Untersuchung einiger Aspekte der Usability des Layouts von Webseiten. An dieser Stelle könnte argumentiert werden, dass diese de-facto-Standards jedem bekannt sind und keiner automatischen Überprüfung bedürfen. Werden jedoch die oben beschriebenen Tests betrachtet so wird klar, dass oft nicht alle Standards beim Design des Layouts beachtet werden. WUSAB weist den Designer einer Webseite darauf hin und schlägt alternative Gestaltungsmöglichkeiten vor. Da auch dieses Tool nur einen Teilbereich der Usability abdeckt, kann es einen Experten nicht ersetzen. Es eignet sich jedoch durchaus dazu, unerfahrene Entwickler auf Gestaltungsfehler hinzuweisen oder, wird von einem Web-Engineering-Tool ausgegangen, die Fehler bei der Konzeption des Layouts in einem frühen Stadium des Entwicklungsprozesses festzustellen.

Bei der Überprüfung bestehender Webseiten mit WUSAB findet ein wesentlicher Punkt der Usability des Layouts keine Beachtung. Die Konsistenz einer Webseite wird nicht berücksichtigt. Will man dies erreichen, müssten Webseiten-Designer dem Tool bekannte Klassen oder IDs verwenden. Zusätzlich muss WUSAB alle Seiten des Webauftrittes kennen, um diese auf ihre Konsistenz überprüfen zu können. Alternativ dazu wäre eine automatische Bestimmung der Seitenbereiche denkbar.

Ein weiterer Nachteil von WUSAB ist die feste Implementierung der Regeln. Es erlaubt keine Anpassung an neue Regelwerke oder Standards. Zwar besteht die Möglichkeit einige Grenzwerte festzulegen, die Evaluationslogik ist jedoch festgelegt und kann damit nicht beeinflusst werden. Änderungen einer Regel müssen direkt im Quellcode vorgenommen werden. Der Entwurf einer Beschreibungssprache dieser Richtlinien würde eine Trennung des Quellcodes und der Evaluationslogik bewirken. Dazu müsste WUSAB die Überführung der beschriebenen Standards in generische Methoden zur Überprüfung übernehmen. Somit könnten Regeln einfacher und schneller an neue Standards angepasst werden. Außerdem könnte die Auswertungslogik für den Benutzer dadurch transparenter und leicht verständlich beschrieben werden. Weiters ist eine Anpassung der Regeln durch den Benutzer denkbar.

Trotz dieser Nachteile zeigt WUSAB wie man die automatische Evaluierung der Benutzerfreundlichkeit von Webseiten verbessern kann. Es zeigt, dass noch nicht alle Möglichkeiten in diesem Gebiet ausgeschöpft sind und trägt dazu bei, Usability-Tests effizienter zu gestalten.





## Quellen:

- [1] International Standards Organization. ISO Norm 9241-11. Ergonomic requirements for visual display terminals.
- [2] Beirekdar, A. Vanderdonckt, J. Noirhomme-Fraiture, M. KWARESMI - Knowledge-based Web Automated Evaluation with REconfigurable guidelineS optiMization. In PreProceedings of 9th International Workshop on DSV-IS'2002, pp. 362-376.
- [3] Ivory, Melody Y., Hearst, Marti A. The State of the Art in Automating Usability Evaluation of User Interfaces. In ACM Computing Surveys 01. Vol. 33 Issue 4, p470, 47p
- [4] Nielsen, J. 1993. Usability Engineering. Boston, MA: Academic Press.
- [5] Correani, F. Leporini, B. Paternò, F. Supporting Web Usability for Vision Impaired Users. In Proceedings User Interface for All. Vienna 04. Lecture Notes Computer Science 3196, pp. 242-253.
- [6] Balbo, S. Automatic evaluation of user interface usability: Dream or reality. In S. Balbo, Ed., Proceedings of the Queensland Computer-Human Interaction Symposium. Queensland 95. Bond University.
- [7] Atterer, Richard. Schmidt, Albrecht. Adding Usability to Web Engineering Models and Tools. In Proceedings of the Fifth ICWE. Sydney 05. Springer LNCS 3579, pp. 36-41.
- [8] Chi, E.H. Pirolli, P. Pitkow, J.E. The scent of a site: a system for analyzing and predicting information scent, usage, and usability of a Web site. In Proceedings CHI 2000, pp. 161-168.
- [9] Manhartsberger, Martina. Musil, Sabine. Web usability: das Prinzip des Vertrauens. Galileo Press. 2001.
- [10] Spool, Jared M. Scanlon, Tara. Schroeder, Will. Snyder, Carolyn. DeAngelo, Terri. Web Site Usability: A Designer's Guide. San Francisco. CA 99.
- [11] Adkisson, Heidi. Identifying De-facto standards for E-Commerce Web Sites. Master's thesis 2002. Xerox Palo Alto Research Center.
- [12] Nielsen, J. Design Web Usability: the Practice of Simplicity. New Riders Publishing. Indianapolis 99.
- [13] Cugini, J. Scholtz, J. VISVIP: 3D Visualization of Paths through Web Sites. Proceedings of the International Workshop on Web-Based Information Visualization (WebVis'99), pp. 259-263
- [14] Scholtz, Jean. Laskowski, Sharon. Developing usability tools and techniques for designing and testing web sites. In Proceedings of the 4th Conference on Human Factors & the Web. 1998.





## Online-Quellen:

- [@1] Shawn Lawton Henry. Introduction to Web Accessibility. World Wide Web Consortium (W3C), Web Accessibility Initiative (WAI). 2005. URL: <http://www.w3.org/WAI/intro/accessibility.php>
- [@2] Lynch, Patrick. Horton, Sarah. Web Style Guide, 2<sup>nd</sup> edition. 2005. URL: <http://webstyleguide.com>
- [@3] Research-Based Web Design & Usability Guidelines. US Department of Health and Human Services. 2006. URL: <http://www.usability.gov/pdfs/updatedguidelines.html>
- [@4] Nielsen, Jakob. Alertbox: Current Issues in Web Usability. Bi-weekly column. URL: <http://www.useit.com>
- [@5] General Services Administration. Section 508. URL: <http://www.section508.gov/>
- [@6] <http://zing.ncsl.nist.gov/WebTools/WebVIP/overview.html>
- [@7] Advisor Media. Advisor Web Ad Opportunities. URL: <http://zones.advisor.com/MediaKit.nsf/w/AdvertSpecsWeb>
- [@8] Bernard, Michael. Criteria for optimal web design (designing for usability). 2003. URL: <http://psychology.wichita.edu/optimalweb/position.htm>
- [@9] Sun Microsystems. Java Compiler Compiler [tm] (JavaCC [tm]) - The Java Parser Generator. 2006. URL: <https://javacc.dev.java.net/>