Kontextsensitiver Guideline-Review als Methode der automatisierten User-Interface-Evaluierung

Susanne Reindl

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juli 2011

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 25. Juni 2011

Susanne Reindl

Inhaltsverzeichnis

Kurzfassung

Usability-Evaluierung ist ein wichtiger Teil bei der Erstellung von User-Interfaces. Unterschiedliche Methoden helfen Designern Probleme zu identifizieren und folglich die Usability ihres Produktes zu verbessern. Da diese Methoden einen gewissen zeitlichen und finanziellen Aufwand mit sich bringen, existieren als Ergänzung oder Alternative dazu Tools, welche die Evaluierung von User-Interfaces automatisieren. Bei der Automatisierung eines Guideline-Reviews evaluiert eine Software ein User-Interface gegen die Einhaltung von Usability-Guidelines und teilt dem Designer die gefundenen Verstöße mit.

Diese Arbeit analysiert bestehende Tools aus diesem Bereich und stellt die Idee eines automatisierten, kontextsensitiven Guideline-Reviews vor. Dazu wurde im praktischen Teil der Arbeit ein Prototyp entwickelt, durch den die Visualisierung der Idee ermöglicht wurde. Neben der Darstellung des entwickelten User-Interface, wird sowohl das Interaktionskonzept als auch die Struktur der Applikation erläutert. Um die Alltagstauglichkeit und die Usability der umgesetzten Lösung zu bewerten, wurde eine empirische Evaluierung durchgeführt.

Abstract

Usability evaluation is an important part in the creation of user interfaces. Different methods help designers to identify problems and hence to improve the usability of their product. Since these methods can be expensive and time-consuming, there are tools that automate the evaluation of user interfaces as a complement or alternative. In the automation of a guideline review, a software product evaluates compliance of a user interface with usability guidelines and reports the violations found.

This thesis analyzes existing tools in this field and introduces the idea of an automated, context-sensitive guideline review. For the visualization of the idea, a prototype has been developed in the practical part of the work. Beside the presentation of the developed user interface, both – the interaction concept and the structure of the application – are explained. In order to assess the practicality and the usability of the developed solution, an empirical evaluation was performed.

Kapitel 1

Einleitung

1.1 Motivation

Diese Arbeit steht ganz im Zeichen der Usability. Usability-Evaluierung hilft den Designern Probleme zu identifizieren, diese zu verstehen und folglich die Usability des Produktes zu verbessern. Dafür existieren unterschiedliche analytische und empirische Methoden, mit deren Hilfe Personen die Usability eines interaktiven Systems oder eines entsprechenden Prototypen beurteilen. Diese Methoden bringen jedoch gewisse Nachteile mit sich. Neben einem zeitlichen und finanziellen Aufwand, kann durch die Subjektivität der Gutachter, die Anwendung derselben Methode zu unterschiedlichen Ergebnissen führen.

Als Alternative oder Ergänzung dazu existieren Tools, welche die Evaluierung der Usability eines User-Interface automatisieren. Tools zur Automatisierung eines Guideline-Reviews assistieren dem Entwickler durch automatisches Finden und Erläutern von Usability-Verstößen und machen teilweise Vorschläge um diese zu beheben. Die Evaluierung wird von einer Software vorgenommen, muss jedoch von einer Person angestoßen werden. Je später der Zeitpunkt der Durchführung, desto mehr Fehler können sich in das Design des User-Interface eingeschlichen haben.

Besser wäre ein automatisierter Guideline-Review, bei welchem der Entwickler sofort zum Zeitpunkt der Entstehung eines Usability-Verstoßes, einen Hinweis darauf erhält. Es stellt sich die Frage, ob und in welchem Ausmaß eine solche kontextsensitive Evaluierung automatisiert möglich ist, ohne dass diese vom Entwickler selbst angestoßen werden muss.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, herauszufinden, ob die Durchführung eines automatisierten Guideline-Reviews während der Erstellungsphase eines User-Interface, kontextsensitiv möglich ist.

Zu diesem Zweck, soll ermittelt werden, wie ein Guideline-Review bei be-

reits bestehenden Lösungen automatisiert durchgeführt wird und welche Vor- und Nachteile diese Tools mit sich bringen. Das Aufzeigen von unterschiedlichen Methoden zur Usability-Evaluierung soll ein allgemeines Verständnis für die Problematik schaffen.

Auf diesen Grundlagen aufbauend, wird durch das zu Grunde liegende technische Projekt, eine Möglichkeit des automatisierten, kontextsensitiven Guideline-Reviews in Form eines Prototypen, dargestellt. Die Umsetzung des Prototypen visualisiert zum einen die eigene Lösungsidee, zum anderen zeigt sie technische Herausforderungen in diesem Zusammenhang. Die empirische Evaluierung des User-Interface soll die Frage nach der Alltagstauglichkeit einer solchen Lösung und den dahinter liegenden Ideen und Konzepten beantworten.

Die Ergebnisse aus der Analyse von bestehenden Lösungen, der Umsetzung einer eigenen Lösung und der empirischen Evaluierung, sollen dem Leser die Frage beantworten, auf welche Weise ein kontextsensitiver Guideline-Review, während der Erstellungsphase eines User-Interface, automatisiert durchgeführt werden kann.

1.3 Aufbau der Arbeit

Nachdem in diesem Kapitel die Motivation und Zielsetzung der vorliegenden Arbeit erläutert wurden, folgt in *Kapitel 2* die Darstellung unterschiedlicher Typen und Quellen von Usability-Guidelines.

In *Kapitel 3* wird auf unterschiedliche Möglichkeiten zur Usability-Evaluierung eingegangen. Unterschieden wird dabei zwischen empirischen und analytischen Methoden.

Kapitel 4 beschreibt den *State of the Art* der automatisierten User-Interface-Evaluierung. Es werden Tools zur Automatisierung unterschiedlicher Methoden vorgestellt, wobei der Schwerpunkt auf die Automatisierung eines Guideline-Reviews gelegt wird.

Die Darstellung eines eigenen Lösungsansatzes zur Automatisierung eines kontextsensitiven Guideline-Reviews erfolgt in *Kapitel 5*. Das Kapitel umfasst eine Beschreibung des Interaktionskonzepts, die Darstellung des entwickelten Prototypen und erläutert die Struktur der Applikation.

Zur Überprüfung der Alltagstauglichkeit, wurde eine empirische Evaluierung der entwickelten Lösung durchgeführt. In *Kapitel 6* werden die durchgeführte Untersuchung und die daraus resultierenden Ergebnisse beschrieben.

Das abschließende *Kapitel 7* fasst die erreichten Ergebnisse dieser Arbeit zusammen und nennt Erweiterungs- und Verbesserungsmöglichkeiten. Zur Diskussion wird auf die in *Kapitel 4* beschriebenen existierenden Lösungen Bezug genommen.

Kapitel 2

Usability-Guidelines

Usability, im Deutschen oft übersetzt mit dem Wort Benutzerfreundlichkeit, wird von Jakob Nielsen und Hoa Loranger wie folgt definiert [28]:

> „Usability ist ein Qualitätsmerkmal, wie einfach etwas zu benutzen ist. Es geht genauer gesagt darum, wie schnell Menschen die Benutzung eines Gegenstands erlernen können, wie effizient sie während seiner Benutzung sind, wie leicht sie sich diese merken können, wie fehleranfällig der Gegenstand ist und wie er den Nutzern gefällt. Wenn die Nutzer einen Gegenstand weder nutzen möchten noch können, bräuchte er eigentlich gar nicht zu existieren."

Der vermehrte Einsatz des Computers am Arbeitsplatz in den 1980er Jahren, führte zur Entwicklung von Hardware- und Software-Produkten, welche die einfache Bedienung durch den Benutzer zum Ziel hatten. Realisiert wurde dies durch sogenannte GUIs (Graphical-User-Interface), welche dem Laien die Interkation mit der Software erleichterten [2, 19]. Mit der dadurch steigenden Präsenz von PCs in der Gesellschaft, stieg auch das Interesse an den Bedürfnissen und Anforderungen der Endbenutzer und damit die Bedeutung der Usability von User-Interfaces [33].

Das Wissen zur Erreichung einer guten Usability, kann in Form von Usability-Guidelines festgehalten werden. Im Bezug auf die Usability eines User-Interface beschreibt Jean Vanderdonckt in [42] eine Usability-Guideline als einen Design- und/oder Evaluierungsgrundsatz, welcher beachtet werden muss, um die Usability eines User-Interface für eine bestimmte Benutzergruppe, zur Ausführung einer bestimmten interaktiven Aufgabe, gewährleisten zu können.

Usability-Guidelines existieren in unterschiedlichen Formen und können in unterschiedlichen Quellen gefunden werden. In Anlehnung an [19] und [31] lassen sie sich, wie Abbildung 2.1 zeigt, anhand ihres Ursprungs und dem Grad ihrer Genauigkeit (Typ) klassifizieren.

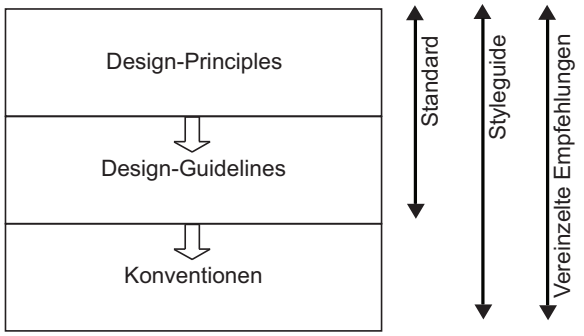


Abbildung 2.1: Usability-Guidelines eingeteilt nach deren Typ und Ursprung, vgl. [19].

Im weiteren Verlauf des Kapitels folgt eine Beschreibung der jeweiligen Typen und möglichen Quellen von Usability-Guidelines.

2.1 Typen von Guidelines

Die vorgenommene Einteilung in diesem Abschnitt betrifft den Grad der Genauigkeit von Usability-Guidelines. Die Spanne reicht von sehr allgemein gehaltenen Guidelines bis zu sehr spezifischen Guidelines. Abbildung 2.1 unterscheidet zwischen sehr allgemein definierten *Design-Principles*, auf Grundsätzen basierenden *Design-Guidelines* und sehr spezifischen Empfehlungen und *Konventionen*. Diese Begriffe werden in der Literatur oft synonym verwendet. Eine strikte Trennung und eindeutige Definition der Begriffe ist daher nicht möglich.

2.1.1 Design-Principles

Principles, Leitsätze oder Grundsätze, bilden die Basis zur Erreichung eines guten Designs. Sie enthalten Wissen über die Wahrnehmung und das Verhalten von Benutzern und können aufgrund dieser allgemein gehaltenen Informationen, bei jeder Erstellung eines User-Interface zum Einsatz kommen [19, 31]. Jakob Nielsen definiert in [27] zehn Principles, auch Heuristiken genannt, zur Erreichung eines guten User-Interface-Designs. Als Beispiel hier, wird eine seiner Heuristiken, betreffend die Konsistenz und das Einhalten von Konventionen innerhalb einer Plattform, angeführt:

> „Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.“

2.1.2 Design-Guidelines

Bezüglich des Grades der Genauigkeit liegen Design-Guidelines zwischen den sehr allgemein gehaltenen Design-Principles und den sehr spezifischen Konventionen. Sie basieren auf Design-Principles und beziehen sich auf einen bestimmten Bereich des Designs [19, 31]. Als Beispiel folgt eine Guideline zur richtigen Verwendung von GUI-Komponenten aus [10]:

> „Restrict the number of buttons on a window to six or fewer."

2.1.3 Konventionen

Konventionen oder Empfehlungen beziehen sich, ebenfalls wie Design-Guidelines, auf einen bestimmten Bereich des Designs und zielen auf die Verwendung innerhalb einer bestimmten Organisation ab. Sie enthalten sehr klare, präzise Anweisungen und lassen keinen Platz für Interpretationen. Solche Empfehlungen werden aus existierenden Design-Principles und Design-Guidelines, unter der Berücksichtigung spezieller Anforderungen, abgeleitet [19, 31]. Ein Besipiel dafür sind die *Windows User Experience Interaction Guidelines* des *Microsoft Developer Network* [21]. Diese Sammlung von Guidelines ist maßgeschneidert für die Erstellung von Windows-basierten Applikationen.

2.2 Quellen von Guidelines

Als mögliche Quellen für Usability-Guidelines wird auf Abbildung 2.1 zwischen *Standards*, *Styleguides* und *vereinzelten Empfehlungen* von Designern unterschieden. Es folgt eine Erläuterung dieser Begrifflichkeiten.

2.2.1 Standards

Standards sind offizielle und öffentlich zugängliche Dokumente mit Design-Principles oder Design-Guidelines zur Standardisierung unterschiedlicher Bereiche. Herausgeber von Standards können sowohl nationale als auch internationale Organisationen sein [19, 39]. Eine dieser Organisation ist die *International Organization for Standardization*[1]. Ein Beispiel für den Bereich Human-Computer-Interaction (HCI) und Usability ist der Standard *EN ISO 9241: Ergonomics of human-system interaction:* Dieser Standard beschreibt Richtlinien der Interaktion zwischen Mensch und Computer. Im Bezug auf User-Interfaces, enthält der Standard Teile mit Leitlinien zur Gestaltung interaktiver Systeme oder zur Gestaltung von Benutzerschnittstellen für das World-Wide-Web.

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

Das *World Wide Web Consortium (W3C)*[2] ist eine internationale Organisation zur Standardisierung von Techniken, betreffend das World-Wide-Web. Richtlinien für die barrierefreie Gestaltung von User-Interfaces bieten die *Web Content Accessibility Guidelines (WCAG)* [44].

2.2.2 Styleguides

Ein Styleguide beinhaltet Design-Principles, Design-Guidelines und Konventionen zur Definition eines einheitlichen Look&Feel, maßgeschneidert auf ein bestimmtes Produkt oder eine bestimmte Organisation.

Viele Styleguides finden sich online auf Websites von Firmen oder Organisationen [10,39]. Ein Beispiel dafür sind die *Windows User Experience Interaction Guidelines* [21] zur einheitlichen Gestaltung von Windows-basierten Applikationen.

2.2.3 Vereinzelte Empfehlungen

Design-Principles, Design-Guidelines oder Konventionen, welche sich inhaltlich mit dem selben Bereich befassen, werden häufig zusammengefasst dargestellt. Empfehlungen von Designern oder Entwicklern beinhalten empirisch getestete Guidelines, welche sich meist auf einen bestimmten Bereich der Usability beziehen. Solche Empfehlungen werden in Form von Conference-Proceedings, als einzelne Artikel, als Bücher oder auch als Standards veröffentlicht [19].

Das Wesentliche beim Designen von User-Interfaces im Web ist die Navigation und die Präsentation von Informationen. Dementsprechend existieren Guidelines, welche sich inhaltlich speziell diesen Bereichen widmen. Accessibility-Richtlinien beziehen sich nicht ausschließlich auf eine benutzerfreundliche optische Gestaltung einer Anwendung, sondern sind Empfehlungen um Webinhalte barrierefreier darzustellen. Durch die Einhaltung dieser Empfehlungen können Inhalte für beeinträchtigte Menschen barrierefrei gestaltet werden. Ein Beispiel für eine Guideline aus den *Web Content Accessibility Guidelines (WCAG) 2.0* ist folgende [44]:

> „Stellen Sie Textalternativen für alle Nicht-Text-Inhalte zur Verfügung, so dass diese in andere vom Benutzer benötigte Formen geändert werden können, wie zum Beispiel Großschrift, Braille, Symbole oder einfachere Sprache."

Vereinzelte Empfehlungen existieren für viele unterschiedliche Bereiche der Usability. Ein weiteres Beispiel sind die Empfehlungen *Guidelines for Multimedia on the Web* [25] von Jakob Nielsen.

[2] http://www.w3.org/

Kapitel 3

Usability-Evaluierung

Gemäß [33] und [37] hilft Usability-Evaluierung dem Designer Usability-Probleme zu identifizieren, diese Probleme und dahinter liegende Ursachen zu verstehen und zeigt Möglichkeiten zu deren Behebung. Dies umfasst alle analytischen oder empirischen Methoden zur Beurteilung der Usability eines interaktiven Systems oder eines entsprechenden Prototypen. Es empfiehlt sich, analytische Methoden durchgehend während des Designprozesses anzuwenden. Die Ergebnisse daraus dienen zum einen der frühzeitigen Identifikation von Usability-Problemen und zum anderen als Grundlage für die Durchführung von empirischen Evaluierungen.

In Anlehnung an Michael Scriven [36] erfolgt in [33] eine Unterscheidung zwischen formativer und summativer Evaluierung. Eine formative Evaluierung ist eine entwicklungsbegleitende Analyse mit dem Ziel der laufenden Verbesserung des Produktes. Zu unterschiedlichen Zeitpunkten des Entwicklungsprozesses werden Prototypen erstellt und evaluiert. Eine summative Evaluierung beantwortet Fragen wie: „Erfüllt das Produkt die spezifizierten Ziele?“ oder „Ist das Produkt besser als seine Vorgänger oder Mitbewerber?“ und dient als zusammenfassende Bewertung eines Produktes nach dessen Fertigstellung. Eine typische summative Evaluierung wäre das Messen von Performance-Zeiten und Fehlerraten.

Der folgende Abschnitt befasst sich mit Methoden zur empirischen Evaluierung der Usability eines Produktes. Darauf folgt eine Erläuterung unterschiedlicher Möglichkeiten zur analytischen Evaluierung.

3.1 Empirische Evaluierung

Die empirische Evaluierung beschreibt benutzerorientierte Methoden, bei denen künftige, reale Benutzer bei der Interaktion mit einem Produkt oder Prototypen beobachtet werden. Zur Gewinnung von empirischen Daten existieren relativ formlose Methoden, wie die Beobachtung von potentiellen Benutzern während der Verwendung eines Prototypen oder systematisch straffe

Methoden zur Messung der Performance oder der Fehlerraten. Der Vorteil dieser Methoden ist die Evaluierung eines Systems basierend auf empirischen Daten und nicht, wie bei der analytischen Evaluierung, basierend auf Annahmen oder Vorschlägen von Experten [33, 37].

3.1.1 Usability-Test

Ein Usability-Test wird durchgeführt, um die Nützlichkeit und Gebrauchstauglichkeit eines Produktes zu überprüfen. Das Ziel ist, herauszufinden, ob das entwickelte Produkt von der künftigen Benutzergruppe verwendbar ist und ob es den vorgesehenen Zweck erfüllt. Vorrangig eingesetzt wird diese Methode zum Testen von Desktop-basierten Anwendungen. Zur Durchführung werden häufig kontrollierte Umgebungen, wie z.B. Usability-Labore verwendet. Als geeignet erachtet wird die Anzahl von 5-12 Personen zur Durchführung eines Tests. Bei zeitlichen oder budgetären Begrenzungen, kann schnelles Feedback auch von nur zwei bis drei Benutzern eingeholt werden.

Die empirischen Daten werden durch eine Kombination von Methoden gewonnen. Die Zeit, die der Teilnehmer benötigt um eine Aufgabe fertigzustellen (z.B. das Finden einer Website) und die Anzahl der Fehler (z.B. die Wahl einer falschen Option im Menü) sind die zwei wichtigsten Maße der quantitativen Performance-Messung. Die Aufzeichnung der Performance wird durch das Logging von Tastenanschlägen und Mausbewegungen, zusammen mit Videoaufzeichnungen ermöglicht.

Durch viele Usability-Tests werden auch subjektive Eindrücke der Testpersonen gewonnen. Ein Fragebogen kann dazu verwendet werden, um die Zufriedenheit der Benutzer bei der Verwendung des Produktes oder einer speziellen Aufgabe herauszufinden. Eine weitere Möglichkeit ist das Führen von strukturierten oder halbstrukturierten Interviews. Bei der Methode *Lautes Denken* werden die Testpersonen gebeten, ihre Eindrücke und Ziele während der Bearbeitung der Aufgaben zu verbalisieren. Durch die Analyse der Aufzeichnungen ist es möglich, Rückschlüsse auf Eindrücke, Wünsche oder Probleme bei den jeweiligen Aufgaben zu ziehen.

Nach der Durchführung eines Usability-Tests werden auf Basis der Aufzeichnungen Auswertungen erstellt und Schwachstellen analysiert [24,33,37].

3.1.2 Feldstudie

Bei einer Feldstudie erfolgt die Untersuchung der Produktnutzung unter natürlichen/realistischen Gegebenheiten. Das Ziel ist, herauszufinden, wie ein Produkt oder Prototyp von der Zielgruppe angenommen und im Arbeits- und Alltagsleben verwendet wird. Dies unterscheidet sich sehr stark von der kontrollierten Umgebung eines Usability-Tests, bei dem die Aufgaben im Vorfeld ganz genau festgelegt werden. Es wird davon ausgegangen, dass sich

die Interaktion zwischen Mensch und Produkt im alltäglichen Leben von der in einem Usability-Labor unterscheidet. Durch eine Feldstudie kann demnach ermittelt werden, wie erfolgreich das Produkt in der realen Welt sein wird. Ein Nachteil ist, dass das Testen eines Produktes oder die Überprüfung einer Hypothese nicht mit derselben Genauigkeit wie bei einem Usability-Test möglich ist. Dies erschwert präzise Aussagen über die Usability eines Produktes oder das Verhalten von Testpersonen.

Die Dauer einer Feldstudie streckt sich von ein paar Minuten bis hin zu ein paar Monaten oder einem Jahr. Daten werden primär durch Beobachtung und das Führen von Interviews gewonnen. Video- und Audioaufzeichnungen dienen zum Festhalten der Abläufe in der ausgewählten Umgebung [33, 37].

3.2 Analytische Evaluierung

Zur Bewertung der Usability eines Produktes ist das Involvieren von tatsächlichen Benutzern nicht zwingend erforderlich. Der Ansatz der analytischen Evaluierung beinhaltet expertenorientierte Methoden zur Usability-Inspektion und Methoden zur Vorhersage oder Simulation des mentalen Modells eines Benutzers. Eine Motivation für die Anwendung von analytischen Methoden ist, dass diese relativ schnell und einfach in allen Phasen der Produktentwicklung angewandt werden können. Des Weiteren ist die analytische Evaluierung eines Systems, aufgrund ihrer einfachen und schnellen Durchführbarkeit, oft kostengünstiger als Usability-Tests oder Feldstudien, wofür kontrollierte Umgebungen und die Integration von realen Benutzern erforderlich ist [33, 37].

3.2.1 Usability-Inspektion

Usability-Tests und Feldstudien wenden sich direkt an mögliche Benutzer und liefern daher grundlegende Resultate im Bezug auf die Usability eines Produktes. Diese Methoden sind jedoch mit einem gewissen zeitlichen und finanziellen Aufwand verbunden. In manchen Phasen der Produktentwicklung kann es wichtiger sein, schnell und kostengünstig, ohne hohen Aufwand, Feedback zu erhalten. In einer solchen Situation kann Usability-Inspektion eine gute Alternative bieten. Usability-Inspektion ist der Oberbegriff für Methoden, bei dem Experten oder Gutachter ein Interface eines interaktiven Produktes nach gewissen Kriterien beurteilen und so mögliche Usability-Probleme identifizieren. „Experte“ ist ein weit dehnbarer Begriff, bezeichnet im Wesentlichen jedoch eine Person mit Erfahrung bei der Durchführung von Evaluierungsmethoden und Wissen im Bereich HCI. Der Einsatz erfolgt formativ und ermöglicht das frühzeitige Erkennen von Schwachstellen. Bei frühen Entwürfen eines Produktes z.B. in Form von Skizzen, ist jedoch mit eingeschränkten Ergebnissen zu rechnen. Je mehr die Benutzeroberfläche dem fertigen Produkt entspricht, desto aussagekräftiger sind die erzielten

Ergebnisse [37, 40].

In den 1990er Jahren entwickelten sich als Alternative zu Usability-Tests unterschiedliche Inspektions-Methoden [37], welche in den folgenden Abschnitten beschrieben werden.

Heuristische Evaluierung

Jakob Nielsen sieht die Methode der heuristischen Evaluierung als Ergänzung zum klassischen Usability-Test und bezeichnet diese als eine *Discount Usability Engineering Method*. Ziel ist die Überprüfung der benutzerfreundlichen Gestaltung eines Produktes gegen einen Satz von allgemein gültigen Usability-Principles durch einen Gutachter. Im Kontext der Evaluierung nennt man diese auch Heuristiken. Aus der Analyse von 249 Usability-Problemen, entwickelte Jakob Nielsen zehn Heuristiken [27] als Referenz für die Durchführung einer heuristischen Evaluierung. In Anbetracht der sehr allgemein gehaltenen Prinzipien empfiehlt Nielsen als Ergänzung die Entwicklung von produktspezifischen Heuristiken [29, 30].

Im Bezug auf die Kosten stellt sich Entwicklern die Frage, wie viele Experten benötigt werden um ein gutes Ergebnis zu erzielen. Während es Usability-Probleme gibt, die sehr einfach zu finden sind, können manche Probleme nur von wenigen Experten identifiziert werden. Aus diesem Grund empfiehlt es sich, drei bis fünf Gutachter in eine heuristische Evaluierung zu involvieren. Wie Abbildung 3.1 zeigt, sollten diese rund 75 % der vorhandenen Usability-Probleme identifizieren [30].

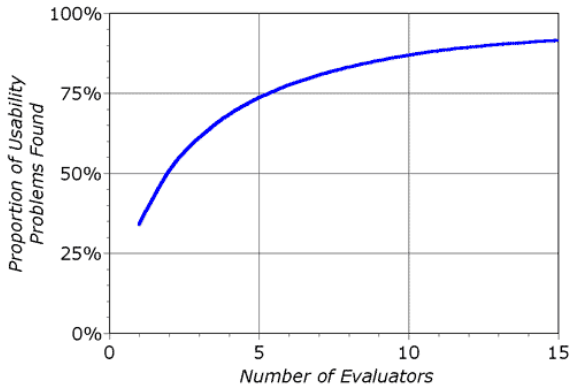


Abbildung 3.1: Die Kurve zeigt den Anteil gefundener Usability-Probleme in einem Interface durch heuristische Evaluierungen mit einer bestimmten Anzahl von Gutachtern [30].

Da dieses Verfahren kostengünstig und nicht sehr aufwändig in seiner Durchführung ist, ist es unter den Entwicklern weit verbreitet und sehr beliebt. Ein Nachteil ist, dass die Methode keine empirisch belegten, sondern

regelbasierte Hinweise auf Usability-Probleme liefert [37]. Daher sollte gemäß [30] die heuristische Evaluierung nicht als Ersatz, sondern als Ergänzung zum klassischen Usability-Test gesehen werden.

Cognitive-Walkthrough

Eine weitere Methode der Usability-Inspektion ist der Cognitive-Walkthrough, bei dem sich ein Usability-Experte in die Rolle eines möglichen Benutzers versetzt und dabei im Vorfeld definierte Handlungsabläufe analysiert. Dabei ist es wichtig, dass die Experten Aufzeichnungen darüber führen, was im System gut funktioniert hat und was nicht. Auf dieser Basis können Benutzungsalternativen vorgeschlagen werden [37, 40].

Das Verfahren wurde in den 90er Jahren von Wharton et. al . [43] entwickelt und basiert auf der Theorie des explorierenden Lernens aus der Kognitionspsychologie. Der Kernpunkt ist die Evaluierung der leichten Erlernbarkeit des Designs eines Produktes, mit dem Ziel dem Designer oder Entwickler zu erklären, ob und warum das Design die Interaktion mit einem Benutzer beeinträchtigen wird.

Eine Variante des Cognitive-Walkthrough ist der Pluralistic-Walkthrough, bei dem Benutzer, Entwickler und Usability-Experten gemeinsam ein Szenario durchlaufen und Usability-Probleme diskutieren [4].

Der Cognitive-Walkthrough ist wie die heuristische Evaluierung schnell, einfach und mit geringem Kostenaufwand durchzuführen. Außerdem ist ein Vorteil, dass das Verfahren die Evaluierung der Handhabung des Produktes zum Ziel hat, während andere Methoden ausschließlich auf die Identifikation von Usability-Problemen abzielen [37].

Checkliste

Checklisten sind zum Abhaken zusammengefasster Aufzählungen, welche sich an industriellen oder systemspezifischen Standards und Guidelines orientieren, geeignet. Bei dieser Art der Inspektion evaluieren Experten – ähnlich wie bei der heuristischen Evaluierung – ihr Design nach den angeführten Guidelines. Die Messung der Usability durch Checklisten empfiehlt sich, wenn mehrere Produkte nach den gleichen Maßstäben beurteilt und verglichen werden sollen [10, 37].

3.2.2 Modellbasierte Analyse

Ähnlich wie bei der Usability-Inspektion werden hier keine Benutzer zur Evaluierung benötigt. Bei Methoden, welche unter diesem Begriff zusammengefasst werden, verwenden Experten Modelle und Formeln um Prognosen über die Effizienz von unterschiedlichen Systemen und über die Performance von Benutzern erstellen zu können [37].

GOMS

Das GOMS-Modell wurde in den 1980er Jahren von Stu Card, Tom Moran und Anan Newell [5] entwickelt. Durch die Methode werden das Wissen eines Benutzers und der kognitive Prozess bei einer Interaktion mit einem System modelliert. Das Wort *GOMS* ist ein Akronym und setzt sich zusammen aus *goals*, *operators*, *methods* und *selection rules* [5, 37]:

- *Goals* sind Ziele oder Zustände, die der Benutzer erreichen möchte, z.B. das Finden einer Website mit guten Kochrezepten.
- *Operators* sind Aktionen, die der Benutzer zur Erreichung der Ziele setzen darf. Sie werden durch das System festgelegt, z.B. die Bewegung der Maus zum Eingabefeld oder das Drücken der Entertaste.
- *Methods* beziehen sich auf kognitive Prozesse und physische Aktionen des Benutzers zur Erreichung der Ziele, z.B. die Erinnerung daran, dass sich der Befehl *Kopieren* im Menü *Bearbeiten* befindet.
- *Selection Rules* werden vom Benutzer eingesetzt, um sich für eine mögliche Methode zu entscheiden, z.B. die Entscheidung, ob zum Kopieren eines Textes der Befehl *Kopieren* im Menü oder der Shortcut *Strg + C* verwendet wird.

Durch die Zerlegung der Interaktion mit einem System in die oben angeführten elementaren Aktionen, ermöglicht GOMS die modellhafte Evaluierung eines Systems. Es existieren verschiedene Varianten des GOMS-Modells, welche die Evaluierung unterschiedlicher Aspekte eines Systems ermöglichen. Allen Varianten liegt jedoch die Definition, der soeben beschriebenen Konzepte, zugrunde [37].

Fitts' Gesetzt

Mit dem Fitts' Gesetz, entwickelt 1954 von Paul Fitts, kann die benötigte Zeit zum Klicken eines Bildschirmelements mit einem Zeigegerät mathematisch berechnet werden. Das Gesetzt bezieht sich auf die Distanz und Größe der Elemente. Demnach wird für ein weit entferntes Element mehr Zeit zum Anklicken benötigt. Diese Zeit erhöht sich jedoch ziemlich langsam, da die Anwender ihre Bewegungen beschleunigen. Je weiter das Objekt entfernt und je kleiner das Objekt ist, desto mehr Zeit wird benötigt dieses anzuklicken [28]. Die Formel 3.1 besagt, dass die benötigte Zeit zum Klicken eines Objekts wie folgt berechnet werden kann [17]:

$$T = a + b \ \log_2(1 + \frac{D}{W}), \tag{3.1}$$

wobei

- T die berechnete Zeit zum Anklicken eines Elements ergibt.
- a und b empirisch ermittelte Konstanten sind und sich auf die Aktivierungs- und Arbeitszeit des Systems beziehen.

- D die Distanz zwischen dem Zeigegerät und dem Mittelpunkt des Ziel beschreibt.
- W für die Breite des Ziels steht.

Fitts' Gesetz kann hilfreich zur Evaluierung von Systemen im Hinblick auf die benötigte Zeit zur Lokalisierung von Objekten eingesetzt werden. Das Gesetz hilft Designern bei der Platzierung von Elementen in Relation zu anderen Elementen auf dem Bildschirm [37].

Hick-Hyman-Gesetz

Das Hick-Hyman-Gesetz, benannt nach den Psychologen William Edmund Hick und Ray Hyman, beschreibt die Zeit die eine Person in Abhängigkeit auf die Anzahl der Wahlmöglichkeiten benötigt, um eine Entscheidung zu treffen [40]. Die durchschnittliche Reaktionszeit zum Auswählen zwischen einer Anzahl von Auswahlmöglichkeiten, lässt sich wie folgt ermitteln [17]:

$$T = a + b \ \log_2(n), \tag{3.2}$$

wobei

- T für die berechnete Reaktionszeit steht.
- a und b empirisch ermittelte Werte sind, welche abhängig vom Interaktionsgerät und der Erfahrung des Anwenders mit der spezifischen Selektion sind.
- n für die Anzahl der Auswahlmöglichkeiten steht.

Im Bezug auf die Usability von interaktiven Systemen findet das Gesetzt Anwendung in der Vorhersage der benötigten Zeit zum Auswählen einer Option in einem hierarchischen Menü oder einer Navigation [17].

Kapitel 4

State of the Art: Automatisierte User-Interface-Evaluierung

Im Bereich der Usability-Evaluierung existiert eine große Anzahl unterschiedlicher Untersuchungsmethoden, welche anhand verschiedener Kriterien kategorisiert werden können. In Kapitel 3 wurden analytische und empirische Methoden zur Evaluierung eines User-Interface beschrieben. Diese Methoden bringen gewisse Nachteile mit sich. Unterschiedliche Ergebnisse bei Anwendung derselben Methode können auf die Subjektivität der Untersucher oder auf die fehlende Systematik bei der Durchführung zurückgeführt werden. Molich et. al [22] ließen in einer Studie zwei User-Interfaces von zwei unterschiedlichen Teams evaluieren. Die Evaluierung ergab eine Überschneidung der Ergebnisse von nur 1 %. Aus diesem Grund empfehlen Experten die Kombination von mehreren Techniken [24].

Eine Alternative zur manuellen Durchführung der Techniken ist die Automatisierung der Evaluierung durch eine Software. Gemäß [1] kann die automatisierte Evaluierung eines User-Interface, Evaluierungen, welche von Personen durchgeführt werden, zwar nicht ersetzen, aber durchaus ergänzen. Durch das automatische Erkennen von Usability-Problemen wird der Entwickler oder Designer bei der Verbesserung der Usability seines Interface unterstützt, ohne aufwändige Tests durchführen zu müssen oder über ein Expertenwissen zu verfügen.

In [12] wird die Unterteilung in Anlehnung an Balbo [35] zwischen automatisierten Ansätzen aufgegriffen und adaptiert. Diese folgende Unterteilung dient der Spezifizierung und beschreibt welcher Aspekt einer Methode zur Evaluierung der Usability automatisiert wurde:

- Nicht-Automatisiert: Der Untersucher führt alle Aspekte der Evaluierung selbst durch.
- Erfassung: Die Software erfasst automatisch Usability-Daten.

- Analyse: Die Software erfasst Usability-Daten und identifiziert Usability-Probleme.
- Kritik: Die Software automatisiert die Analyse und macht Vorschläge zur Verbesserung.

Tools zur Automatisierung der Usability-Evaluierung zogen in den letzten Jahren viel Interesse auf sich [3]. Durch die große Anzahl von Usability- und Accessability-Richtlinien, galt großes Interesse der automatischen Erkennung von Verstößen gegen Guidelines [15].

Dieses Kapitel versucht einen aktuellen Überblick über die unterschiedlichen Methoden der automatisierten User-Interface-Evaluierung zu geben. Der nachfolgende Abschnitt umfasst eine ausführliche Beschreibung existierender Tools zur Automatisierung von Inspektionsmethoden. Danach erfolgt eine grobe Beschreibung weiterer automatisierbarer Methoden zur Evaluierung der Usability einer Benutzeroberfläche.

4.1 Automatisierte Inspektionsmethoden

Gemäß Abschnitt 3.2.1 bedeutet Usability-Inspektion die Evaluierung eines Interface durch eine Person im Hinblick auf die Einhaltung von Guidelines. Zu den Inspektionsmethoden zählen Techniken wie *Cognitive-Walkthrough*, *Heuristische Evaluierung* oder *Checklisten*.

Vielen Entwicklern und Designern ist die Wichtigkeit von Usability zwar durchaus bewusst, es mangelt jedoch an Wissen, wie die Usability im eigenen Design umgesetzt werden kann. Gerade bei einer großen Menge an Usability-Guidelines besteht die Schwierigkeit diese auch anzuwenden [12]. Um der Problematik entgegen zu wirken, wurden Tools zur automatisierten Inspektion von User-Interfaces entwickelt. Gemäß [12] existiert im Bereich der User-Interface-Inspektion automatisierte Unterstützung für die oben genannten Techniken. Die folgenden Abschnitte erläutern wie bestehende Methoden zur Usability-Inspektion automatisiert eingesetzt werden können. Der Schwerpunkt dabei liegt auf dem Punkt Guideline-Review, da die im Rahmen dieser Arbeit entwickelte eigene Lösung (siehe Kapitel 5) unter diesem Punkt einzuordnen ist.

4.1.1 Cognitive-Walkthrough

Bei einem Cognitive-Walkthrough simuliert ein Gutachter einen potentiellen Benutzer durch die Durchführung von selbst definierten Aufgaben (siehe dazu auch Abschnitt 3.2.1). Dabei stellt der Gutachter fest, ob diese definierten Aufgaben für den Benutzer lösbar wären oder nicht. Da diese Technik einer aufwändigen Dokumentation bedarf, gab es gemäß [12], frühe Ansätze zur Automatisierung dieser Technik. Um den Aufwand dieser Technik zu reduzieren, entwickelten [32] ein System, welches die durchführende Person mit

Fragen konfrontiert. Diese Fragen beziehen sich auf die vom Benutzer selbst festgelegten Aktionen zur Erreichung der Aufgaben und veranlassen den Gutachter zur Durchführung dieser Aktionen. Nachdem der Walkthrough beendet ist, werden die Daten in eine textuelle Zusammenfassung konvertiert. Diese Möglichkeit zur Automatisierung wurde von den Anwendern jedoch als sehr mühsam und zeitaufwändig empfunden [32].

4.1.2 Guideline-Review

Guideline-Review ist eine Bezeichnung für die zugrundeliegende Methodik bei der Durchführung von Usability-Inspektionen mit Hilfe von Checklisten sowie heuristischen Evaluierungen. Bei der Automatisierung eines Guideline-Reviews übernimmt eine Software die Evaluierung eines User-Interface gegen die Einhaltung von Usability-Guidelines. Einen Überblick über unterschiedliche Typen und Quellen von Usability-Guidelines bietet das Kapitel 2.

Tools, welche automatisiert einen Guideline-Review vornehmen, assistieren dem Entwickler durch automatisches Finden und Erläutern von Usability-Verstößen und machen teilweise Vorschläge um diese zu beheben. Manche sind für die Evaluierung von fertigen Benutzeroberflächen konzipiert, während andere für den Gebrauch bereits während der Produktentwicklung gedacht sind [3]. Bezogen auf die in der Einleitung dieses Kapitels beschriebene Unterteilung der Automatisierung, existieren in diesem Bereich Tools zur automatisierten Analyse und Kritik [12].

In der Literatur wird unterschieden zwischen Tools zur Evaluierung von Desktop-Applikationen oder zur Evaluierung von webbasierten Interfaces [3, 12]. Web-Applikationen bieten tendenziell weniger Funktionalität als Desktop-Applikationen. Mit dem primären Ziel, dem Benutzer Informationen anzubieten, beschränkt sich die Funktionalität von Web-Applikationen oft auf die Navigation oder das Ausfüllen von Formularen [12]. Aufgrund neuer Technologien wie *PHP*, *AJAX*, *Flash* oder *Silverlight* existieren dennoch viele funktionale Web-Anwendungen, welche die Unterscheidung zwischen den eben genannten Kategorien erschweren. Da sich die entwickelten Tools dennoch meist auf eine der beiden Kategorien beschränken, erfolgt die Unterscheidung zwischen Desktop- und Web-Anwendungen im folgenden Abschnitt.

Desktop-Anwendungen

Methoden zur automatisierten Evaluierung von Desktop-Anwendungen sind sehr effektiv für Guidelines, welche operationalisiert werden können. Dies beinhaltet quantitative Maße wie die Größe, den Abstand oder die Anzahl von GUI-Komponenten, sowie die Überprüfung der Konsistenz innerhalb einer Anwendung. Nicht gemessen werden können qualitative Aspekte des User-Interface, wie z.B. ob ein Benutzer die Beschriftung eines Labels verstehen wird [12]. In [7] wird gezeigt, dass von einer Menge an etablierten

Guidelines im besten Fall 78 % der Guidelines und im schlechtesten Fall nur 44 % operationalisierbar sind. Automatisierte Ansätze zur Kritik eines Interface, insbesondere jene, welche das Interface modifizieren, stellen das höchste Level zur Unterstützung bei der Einhaltung von Guidelines dar [12].

Im Folgenden werden Methoden und Funktionalitäten von Programmen zur Durchführung von automatisierten Guideline-Reviews für Desktop-Anwendungen dargestellt. Es wurden Tools ausgewählt, deren Ziele und Funktionalitäten ähnlich derer des eigenen Lösungsansatzes (siehe Kapitel 5) sind.

KRI/AG

Bereits im Jahre 1992 entwickelten [16] ein Tool namens *KRI/AG*, welches Interfaces, erstellt mit *TeleUSE*[1] *UIMS*[2], im Hinblick auf die Einhaltung von Guidelines evaluiert. KRI/AG verfügt über eine Wissensbasis bestehend aus allgemeinen und plattformabhängigen Guidelines. Es verwendet die Wissensbasis zum automatisierten Evaluieren und Kritisieren eines User-Interface. Das Design muss als *UIL*[3]-Repräsentation gespeichert werden, um von KRI/AG interpretiert werden zu können. Nachdem die Wissensbasis auf das User-Interface angewendet wurde, werden Kommentare über mögliche Fehler im Design generiert. Das System begutachtet das Interface also nur dann, wenn der Designer ausdrücklich Kommentare dazu wünscht. Beispielhafte Kommentare des Systems sehen wie folgt aus [16]:

- „The [text field at the bottom] does not have a label. There should be a label or header above or the left of it.“
- „The text fields in [the dialog boxes which appear when the user selects Save As or Open] do not have default values.“
- „There is no help menu in the menu bar.“

Den Beispielkommentaren zu entnehmen, analysiert das System das Interface, deckt Probleme auf und macht Vorschläge um diese zu beheben.

Sherlock

Sherlock [18] ist ein Tool zur automatisierten Analyse von Windows-Interfaces. Das Tool evaluiert alle User-Interfaces, welche in ein kanonisches Format[4] übersetzt wurden. Diese übersetzte Datei enthält eine GUI-Objekt-Beschreibung, welche von Sherlock interpretiert werden kann. Zum Zeitpunkt der Entstehung dieser Anwendung existierten Programme zur Übersetzung von Ressourcedateien für *Microsoft Visual Basic 3.0* und *Microsoft Visual*

[1] TeleUSE from Aonix: http://www.aonix.com/teleuse.html
[2] User-Interface-Management-System
[3] User-Interface-Language
[4] Eine eindeutige Darstellungen von Daten

C++ 4.0. Sherlock ist fokussiert auf die Überprüfung der Konsistenz innerhalb eines oder mehrerer User-Interfaces. Das Programm evaluiert Layout-Eigenschaften wie die Größe von Dialogfenstern oder Buttons, die Platzierung von ähnlichen Elementen, die Ausrichtung der Elemente oder die Dichte und Balance der gesamten Oberfläche. Evaluiert wird auch die Konsistenz von Schriftarten, Schriftgrößen oder Hintergrundfarben. Ein weiterer Punkt ist die Evaluierung von inkonsistenten Ausdrucksweisen, wie unterschiedliche Groß- und Kleinschreibung oder Rechtschreibfehler in Beschriftungen von Buttons oder Labels. Die Ausgabe der Kommentare nach erfolgter Analyse wird durch Ergebnistabellen dargestellt. Abbildung 4.1 zeigt als Beispiel für eine Ausgabe von Sherlock die Inkonsistenz bei der Verwendung von Buttons.

BUTTON CONCORDANCE

BUTTON LABEL	DIALOG BOX	BUTTON TYPEFACE	BUTTON FG_COLOR	BUTTON (H, W)	BUTTON POSITION LEFT	RIGHT	TOP
Exit	attapp94	1	2	56, 120	464	56	240
	cover	2	2	57, 153	680	182	536
	coveraf	3	2	41, 89	448	0	392
	coveruf	3	2	41, 89	448	85	392
	frmhand	4	3	49, 105	384	31	136
	frmlogin	4	3	41, 97	248	96	256
	winstat	6	1	49, 113	368	70	424
EXIT	delete	1	1	41, 97	280	45	352
	syllabus	1	1	33, 137	856	7	512
Left SAVE	feed			33, 81	272	647	448
Left Save	omp			33, 97	112	796	456
Right SAVE	feed			33, 89	648	263	448
Right Save	omp			33, 97	544	364	456
SAVE Left	mulq			33, 97	256	653	488
SAVE Right	mulq			33, 97	504	405	488

DISTINCT TYPEFACES IN BUTTONS:
1 = MS Sans Serif 8.25 Bold
2 = MS Sans Serif 18
3 = MS Sans Serif 13.5
4 = MS Sans Serif 12 Bold
5 = MS Sans Serif 9175 Bold
6 = MS Serif 12 Bold

DISTINCT FOREGROUND COLORS IN BUTTONS:
1 = Default Color
2 = ffffffff80000005
3 = 0
4 = ff0000

Abbildung 4.1: Die Auswertungstabelle von Sherlock zeigt die Inkonsistenz der verwendeten Buttons innerhalb eines User-Interface [18].

Die abgebildete Auswertung in Abbildung 4.1 zeigt Folgendes:

- Designer verwendeten sechs unterschiedliche Schriftarten bei der Beschriftung der Button-Labels.
- Die Größe der Buttons war inkonsistent in der gesamten Anwendung. Die Höhe des Buttons *Exit* schwankt zwischen 33 und 57 Pixel.
- In manchen Dialogfenstern verwendeten die Designer die Labels *Save Left* und *Save Right*. In anderen *Left Save* und *Right Save*.

Das Tool Sherlock analysiert sehr viele unterschiedliche Eigenschaften einer Benutzeroberfläche und gibt diese in einer tabellarischen Form für den Benutzer aus. Interpretationen der Ergebnisse oder Vorschläge für die Umsetzung der erkannten Probleme werden jedoch nicht gemacht.

Ergoval

Zur Vereinfachung der Anwendung von Guidelines, entwickelten [8] eine Methode zur Strukturierung von Guidelines und ein System zu deren Evaluierung mit dem Namen *Ergoval*. Ergoval ist für die Evaluierung von in Windows entwickelten grafischen Benutzeroberflächen geeignet. Nähere Angaben dazu werden in der Literatur nicht gemacht.

Um den Benutzer mit leicht handhabbaren Informationen, welche in den sehr abstrakten Guidelines enthalten sind, zu versorgen, wurden diese in ergonomische Regeln umformuliert und danach strukturiert. Die heruntergebrochenen Regeln wurden mit User-Interface-Elementen in Bezug gebracht. Durch Gruppieren von Objekten, welche von denselben Regeln betroffen waren, wurde eine Typologie[5] erstellt. Abbildung 4.2 zeigt einen Auszug aus der erstellten Typologie. Die abgebildeten Typen erben die Eigenschaften des Elterntyps und dessen zugeordneten Regeln. Die User-Interface-Objekte bilden die Blätter dieser Klassifizierung. Ein Objekt kann auch zu mehreren Typen gehören. Ein Button beispielsweise, würde den Rubriken *Command* und *Choice Of State* zugeordnet werden und erbt demnach die Attribute und Regeln beider Rubriken. Die gebildeten Regeln mussten aufgrund der Vererbung nur einmal am höchsten Level der Abstraktion implementiert werden. Durch diese Typisierung wurde die Anzahl der Regeln reduziert und damit die Wartbarkeit des Regelwerks erhöht.

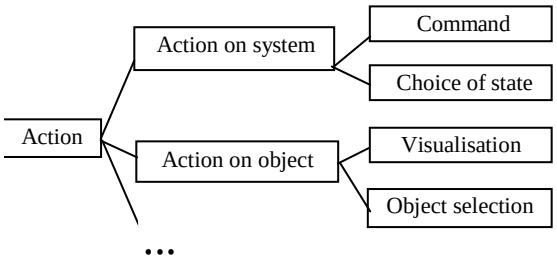


Abbildung 4.2: Auszug der erstellten Typologie für Guidelines [8].

Um das User-Interface zu evaluieren, wird für jedes Element in der Wissensbasis nach Referenzwerten der jeweiligen Elementeigenschaft (z.B. Größe oder Anzahl) gesucht und mit dem tatsächlichen Wert, gefunden im Interface, verglichen. Diese Referenzwerte sind numerischer, boolescher oder symbolischer Natur. Beispiele für die textuelle Beschreibung von Regeln sind folgende [8]:

- „In a set of radio buttons, a radio button must be default preselected.“

[5] Gruppenzuordnung aufgrund der Gesamtheit der Merkmale

- „When a push button is pre-selected, this pre-selection must be graphically displayed.“
- „If the data to be entered (typed or selected) consists in measures, units must be displayed.“

Da eine präzise Evaluierung eines User-Interface bezüglich Guidelines als sehr zeitaufwändig einzustufen ist, versuchten [8] durch Ergoval diesen Prozess zu automatisieren. Eine der Hauptschwierigkeiten dabei war, die Werte der Eigenschaften der grafischen Elemente aus dem Code zu extrahieren. Eine weitere Einschränkung der Automatisierung war, dass nur ein Teil der definierten Regeln in Ergoval angewandt werden konnte. Qualitative oder aufgabenbezogene Informationen konnten nicht automatisiert überprüft werden [8].

Ergoval ist eine Methode, welche viel Wert auf die Strukturierung und Wartbarkeit der gebildeten Richtlinien legt. Diese Richtlinien werden den Interface-Elementen zugeordnet und in Form einer Typologie dargestellt. Der Benutzer erhält durch das System Informationen über nicht eingehaltene Guidelines in seinem Design. Dabei werden die Guidelines in textueller Form dargestellt, jedoch nicht begründet.

HUI Analyzer

Der *HUI (Handheld User Interface) Analyzer* [3] ist ein Tool, welches entwickelt wurde, um Usability-Probleme in User-Interfaces zu finden. Automatisiert werden das Erfassen, Analysieren und Auswerten von Usability-Daten. Obwohl das Tool ursprünglich zur Evaluierung von GUIs auf mobilen Geräten entwickelt wurde, ist es allgemein anwendbar auf GUIs und kompatibel mit allen Applikationen entwickelt mit dem *Microsoft .NET Compact Framework*.

Der HUI Analyzer bietet im Groben zwei Funktionalitäten zur automatisierten Evaluierung. Zum einen bietet das Tool Unterstützung durch die Evaluierung von Eigenschaften einer GUI im Bezug auf eine Menge von konfigurierbaren Usability-Heuristiken. Zum anderen automatisiert das Tool Usability-Tests und reduziert somit den Aufwand, welcher durch derartige Tests entsteht. Im folgenden Text wird auf die erst genannte Funktionalität des HUI Analyzer genauer eingegangen.

Das Erkennen von Usability-Problemen im User-Interface, wird durch einen vordefinierten Satz von sogenannten Usability-Metriken[6] realisiert. Diese Metriken sind definierte Grenzen, welche sich auf Usability-Guidelines beziehen. Abbildung 4.3 zeigt die inkludierten Metriken mit den dazugehörigen Standardwerten. Überprüft werden Eigenschaften wie Größe und Anzahl von GUI-Elementen, Schriftgröße, Menütiefe, Hintergrundfarbe oder Weißraum.

[6] Messbare Werte

Total number of buttons	4
Total number of text input controls	2
Button height	40
Button width	80
Font types count	2
Font size	10
Selection controls per group/container count	5
ListBox item count	5
ListView item count	5
Menu items count	5
Menu depth	3
Percentage of free space	20%
Number of background colors used	2

Abbildung 4.3: Auflistung von definierten Metriken des HUI Analyzer [3].

HUI Analyzer beinhaltet vordefinierte Werte und ermöglicht dem Benutzer zusätzlich die Definition von eigenen Metriken. Die Evaluierung operiert auf dem kompilierten Code des User-Interface und sollte während der Entwicklung iterativ durchgeführt werden. Inspiriert durch moderne Unit-Testing-Tools, ergeben die Evaluierungen gegen diese Metriken entweder true oder false. Diese Funktionalität wird von [3] als *Form Assertion* bezeichnet.

Hotspot Analysis ist eine weitere Funktion des HUI Analyzer, welche durch die farbliche Darstellung die Häufigkeit der Verwendung von User-Interface-Elementen illustriert. Dunkle Farben illustrieren Komponenten welche häufiger verwendet werden. Abbildung 4.4 zeigt, dass die Textarea sowie die Drop-Down-Box links unten sehr häufig verwendet wurden. Der Reset-Button ist weiß und wurde demnach noch nie geklickt.

Die zwei beschriebenen Funktionalitäten des Tools HUI Analyzer, *Form Assertion* und *Hotspot Analysis*, dienen der automatisierten Evaluierung eines Interface bezüglich festgelegter Metriken. Diese Metriken, vorgegeben durch das Tool oder den Entwickler selbst, beziehen sich auf Guidelines, welche es einzuhalten gilt. Die Darstellung der Ergebnisse, resultierend aus der Analyse, erfolgt durch boolesche Werte. Der Entwickler der GUI erkennt demnach, ob er die durch die Metriken festgelegten Grenzen einhält. Er erhält jedoch keine Information über die dahinter liegende Guideline oder eine Begründung für die festgelegten Grenzwerte.

Web-Anwendungen

Die Usability und Accessability von Websites sind wesentliche Faktoren, welche für den Erfolg dieser entscheidend sein können. Zur Erleichterung der

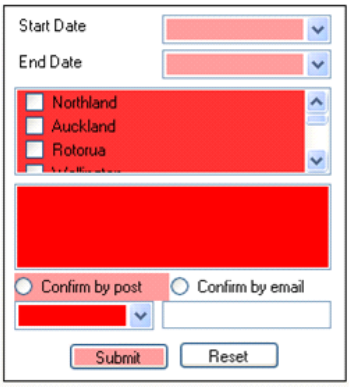


Abbildung 4.4: Hotspot Analyzer. Je dunkler die Elemente eingefärbt sind, desto öfter werden diese von den Benutzern verwendet [3].

Einhaltung von Usability- und Accessability-Guidelines, existieren Tools, welche eine automatisierte Evaluierung einer Website vornehmen. Gemäß [1] formalisieren diese Programme Usability-Regeln, wenden diese auf den HTML-Code an und produzieren einen Report um dem Benutzer die gefundenen Probleme mitzuteilen. Üblicherweise werden diese Anwendungen auch selbst in Form von Web-Applikationen umgesetzt. Nach dem Eingeben einer URL in ein Input-Feld, wird die entsprechende Seite geladen, der HTML-Code analysiert und eine Ausgabe über mögliche Usability-Probleme erstellt [1]. Ivory und Hearst nennen in [12] Tools zur automatisierten Analyse und Kritik von Websites. Zum Zeitpunkt der Recherche für diese Arbeit existiert eine große Anzahl von Programmen zur Evaluierung der Usability, Accessability oder Ästhetik einer Website.

Der folgende Abschnitt bietet einen Querschnitt dieser Tools, um eine Vorstellung über deren Arbeitsweise, Funktionalitäten und Ziele zu erhalten.

The Rating Game

The Rating Game[7] ist ein Analyse-Tool von Lincoln D. Stein, welches die Qualität von Websites anhand einiger messbarer Werte automatisiert evaluiert. Um eine Evaluierung vornehmen zu können, muss eine URL einer beliebigen Seite in ein Textfeld eingegeben werden. Nach der Evaluierung erfolgt eine tabellarische Ausgabe der Werte: z.B. Anzahl der Grafiken, Anzahl der Frames und Applets oder Gesamtanzahl der Wörter. Diese Werte werden dem Benutzer mitgeteilt, ohne ihn über deren Bedeutung zu informieren.

[7] http://stein.cshl.org/ lstein/rater/

Magenta

Magenta[8] ist ein Tool, welches die Evaluierung von Accessability- und Usability-Guidelines unterstützt. Magenta bietet dem Designer einen Editor zur Erfassung und Spezifizierung von neuen oder existierenden Guidelines in einem XML-basierten Format *Guideline Abstraction Language (GAL)*. Das Tool ermöglicht die Überprüfung jeder Guideline, welche in diesem Format erfasst wurde. Die Evaluierung ist beschränkt auf Dokumente implementiert in X/HTML oder CSS [15]. Möchte der Designer seine HTML-Seiten gegen die definierten Guidelines evaluieren, müssen sowohl diese, als auch die Guidelines von ihm ausgewählt werden. Der Report zeigt eine Auflistung der gefundenen Probleme und ermöglicht dem Designer eine Code-Korrektur durch das Tool vornehmen zu lassen. Magenta ermöglicht die Definition folgender Arten von Guidelines:

- Guidelines entsprechend dem italienischen Gesetz für Accessability
- W3C WCAG 1.0 Guidelines
- Accessability- und Usability-Guidelines für visuelle Beeinträchtigungen

Der große Vorteil von Magenta, gemäß eigenen Angaben in [15], ist die Flexibilität in der Verwaltung der Guidelines. Bei vielen bestehenden Tools sind die Guidelines in der eigentlichen Implementierung erfasst, was die Wartung dieser erheblich erschwert. Durch die XML-basierte Sprache können bei Magenta neue Guidelines in externen Files abgelegt werden, ohne die Implementierung ändern zu müssen.

Wusab

Wusab ist eine von Richard Atterer [1] mit Java umgesetzte Web-Applikation zur Evaluierung der Usability und Accessability einer Website. Ebenso wie bei dem Tool *The Rating Game*, wird dem Benutzer des Programms die Möglichkeit zur Eingabe einer URL geboten. Daraufhin wird der HTML-Code der Seite von Wusab geladen und analysiert. Die Implementierung produziert einen Usability-Report als Ausgabe in einer eigenen HTML-Seite. Abbildung 4.5 zeigt exemplarisch einen generierten Output.

Der Prototyp implementiert die folgenden Usability-Kriterien:

- Text complexity: Durch die Verwendung von Wortlisten und unterschiedlichen Algorithmen kann der Validator feststellen, ob die Texte der Website einfach zu verstehen und zu lesen sind.
- Text/background contrast: Durch die Analyse der CSS-Eigenschaften, kann der Farbkontrast der Seite ermittelt werden.
- Alternative text for images: Es erfolgt eine Überprüfung der Alt-Attribute der <img>-Tags.

[8] http://giove.isti.cnr.it/accessibility/magenta/

Results of image alternative text analysis

RESULT	DESCRIPTION
ERROR	3 images on the analysed page have wrong ALT-text. If an ornamental image (a layout image with no relevant content) has a non-empty ALT-text, this will be confusing for people using screen readers or other assistive technology.
OK	8 images on the analysed page have correct ALT-text

Results of layout position analysis

RESULT	DESCRIPTION
OK	One navigation area is located to the left of the main content
OK	Logo is located on top of the vertical navigation
OK	No advertising found
OK	Found additional content on the right
OK	Placement of search area is OK

Results of area size analysis

RESULT	DESCRIPTION
OK	Navigation areas are less than 20%
ERROR	Content area is less than 30%
OK	Advertising areas are less than 20%

Results of scalability analysis

RESULT	DESCRIPTION
OK	Your website fits into a screen with a resolution of 1024 x 768 pixels
ERROR	Your website does not fit into a screen with a resolution of 800 x 600 pixels. Users will have to scroll horizontally!

Abbildung 4.5: Auszug der Ergebnisse eines Guideline-Tests durchgeführt mit Wusab [1].

- Layout analysis: Durch den Abgleich mit de-facto Standards erfolgt die Überprüfung des Layouts.
- Area size analysis: Die relative Größe, angegeben in Pixel, eines Seitenbereiches, sollte innerhalb gewisser Grenzen liegen.

Richhard Atterer erläutert einen Ansatz bei dem nicht nur der HTML-Code analysiert wird, sondern dem Programm auch zusätzliche Information in Form von *Metamodels* zugänglich ist. Die *Metamodels* bieten dem Validator Benutzer-, Plattform- und Umgebungsinformationen (z.B. Lärmpegel in der verwendeten Umgebung). Integriert werden könnten diese Informationen durch die in Web-Engineering-Lösungen, wie z.B. in UWE[9], erzeugten Modelle. Auch andere Quellen zur Bereitstellung der Informationen sind denkbar. Manueller Input durch den Benutzer, Erweiterungen im HTML-Code oder Logdaten können in die Analyse einfließen. Zur semantischen Erweiterung bettet Wusab Informationen durch die Verwendung von class="..." Attributen in den HTML-Code ein. Zusätzlich dazu können die Benutzer durch ein webbasiertes Interface Seitenbereiche definieren.

Des Weiteren betont Atterer den Vorteil einer Integration des Evaluierungsprozesses in eine IDE[10]. Durch diesen Ansatz kann der Entwickler zu einem frühen Zeitpunkt der Entwicklung auf Probleme hingewiesen werden.

Aesthetic Measurement Application (AMA)

Zur Messung der Ästhetik einer Web-Applikation wurde eine *Aesthetic Measurement Application (AMA)* [45] entwickelt. In [23] wurden 14 Charakteristika für ästhetisch ansprechende grafische Benutzeroberflächen identifiziert. Die Ästhetik bezieht sich auf die Attraktivität und visuelle Erscheinung einer Benutzeroberfläche. In [45] wurden sechs der 14 Charakteristika aufgegriffen: Balance, Equilibrium, Symmetrie, Sequence und Rhythmus und Komplexität (Durchschnitt aller Werte). Die Berechnungen erfolgen gemäß den Formeln

[9] UML-basiertes Web-Engineering-Tool: http://uwe.pst.ifi.lmu.de/index.html
[10] Integrated Development Environment

beschrieben in [23].

Zur Erläuterung eines Charakteristikums wird als Beispiel die Balance aufgegriffen. Die Balance entspricht dem optischen Gleichgewicht eines Bildes. In einem Screendesign wird sie durch die gleichmäßige horizontale und vertikale Verteilung von Screen-Elementen erreicht. Berechnet wird die Balance durch folgende Formel:

$$BM = 1 - \frac{|BM_{vertical}| + |BM_{horizontal}|}{2} \tag{4.1}$$

$BM_{vertical}$ und $BM_{horizontal}$ sind jeweils die ermittelten Werte für die vertikale und horizontale Balance. In [45] wurde die Ästhetik einer Website zur Erlernung der Sprache Mandarin gemessen. Abbildung 4.6 zeigt die Applikation AMA und die gemessenen Werte für die Hauptseite der Mandarin-Website zwischen 0 (am Schlechtesten) und 1 (am Besten). Zur Evaluierung muss nach dem Start der Applikation ein Interface von dem Benutzer ausgewählt werden. Die Berechnung wird durch einen Klick auf den Button *Count Aesthetic Values* gestartet.

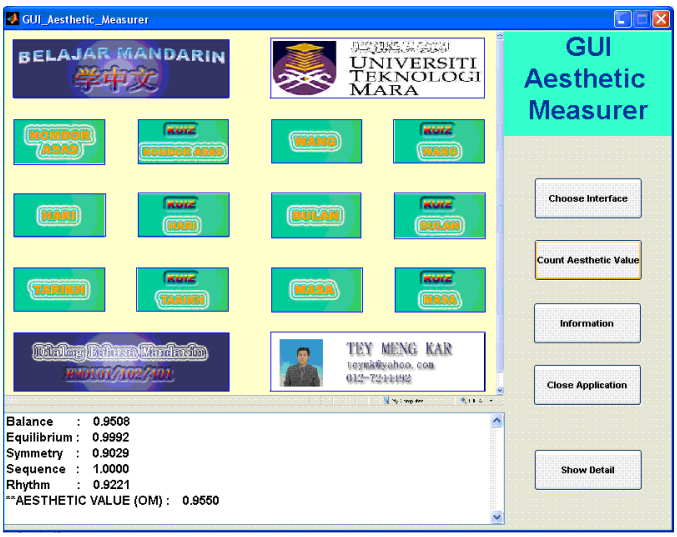


Abbildung 4.6: Ermittelte ästhetische Werte einer Webpage durch AMA [23].

Die Evaluierung eines Interface durch AMA beschränkt sich auf die Ermittlung von sechs ästhetischen Werten. Andere Guidelines werden bei der Evaluierung nicht berücksichtigt.

Online-Validatoren

Eine weitere Möglichkeit zur Evaluierung einer Website ist die Überprüfung durch Online-Validatoren. Ein Beispiel ist der *W3C HTML Validator*[11] welcher die Markup-Validität von Webdokumenten, erstellt in *HTML*, *XHTML*, *SMIL* oder *MathML*, überprüft.

4.2 Weitere automatisierte Methoden

Bisher erfolgte eine ausführliche Beschreibung existierender Tools zur Automatisierung von Inspektionsmethoden. Es existieren weitere Methoden zur Usability-Evaluierung, welche ebenfalls bis zu einem gewissen Grad automatisiert werden können.

4.2.1 Usability-Test

Bei einem Usability-Test wird ein Anwender während der Durchführung vordefinierter Aufgaben beobachtet (siehe dazu auch Abschnitt 3.1.1). Das Festhalten der Aktionen des Benutzers erfolgt live durch Notizen des Untersuchungsleiters oder durch Audio- oder Videoaufnahmen. Unterstützung dabei bieten automatisierte Methoden zum Festhalten von nützlichen Daten wie Remote-Testing, Server- oder Client-Logs. Remote-Testing ermöglicht die Durchführung von Usability-Tests während Tester und Untersuchungsleiter nicht am selben Ort sind. Durchgeführt wird diese Methode durch eine Software, die den Tester durch die Testing-Session leitet und die dabei entstehenden Resultate aufzeichnet [12]. Bei einem Server-Log erzeugen laufende Programme am Server Logfiles mit Daten wie z.B. IP Adresse des zugreifenden Computers, Zeitpunkt des Zugriffes oder die URL der Website die der Benutzer zuvor besucht hat. Server-Logs sind im Hinblick auf die Usability schwierig zu interpretieren oder wenig aussagekräftig. Fehlerhafte Daten sind durch Caching der Browser oder Proxy-Server möglich [11,12]. Client-seitige Logs ermöglichen die Aufzeichnung von exakteren und umfangreicheren Informationen. Ermöglicht werden solche Logs z.B. durch Modifizierung des Quellcodes, spezielle Proxyserver oder Cookies.

Der Vorteil von Analysen basierend auf Logfiles ist, dass große Mengen an Informationen im Hintergrund aufgezeichnet werden können. Der Benutzer der Software oder Website fühlt sich dabei nicht gestört und verhält sich dadurch natürlich. Nachteilig ist die Interpretation solcher Daten, da der Gutachter praktisch keine Hintergrundinformation zu den entstandenen Daten besitzt [12].

Neben Methoden zur reinen Erfassung von Benutzereingaben, existieren auch Applikationen zur automatisierten Analyse entstandener Logdateien. Beispiele für Ergebnisse aus solchen Analyseverfahren sind die Zeit die der

[11] http://validator.w3.org/

Benutzer für die Fertigstellung einer Aufgabe benötigt, die Anzahl der fertiggestellten Aufgaben innerhalb eines Zeitrahmens, das Verhältnis zwischen erfolgreichen und fehlerhaften Interaktionen, die Anzahl der Fehler, die Anzahl der Features, die nie verwendet wurden, die Häufigkeit der Verwendung der Hilfe oder die verbrachte Zeit mit der Hilfe. Quantitative Performance-Messung ist eine weitere Bezeichnung für diese Methode der User-Interface-Evaluierung [24].

4.2.2 Erhebungsmethoden

Das Ziel dieser Methoden ist die Erfassung von subjektiven Eindrücken mehrerer Personen bezüglich verschiedener Aspekte eines User-Interface. Diese Daten werden durch Fragebögen oder Interviews erfasst. Software-Tools unterstützen den Gutachter bei der Erfassung, Verarbeitung und Auswertung der Daten. Realisiert wird diese Methode beispielsweise durch den Einbau von kontextspezifischen Fragen in einem User-Interface. Die Anzeige der Fragen wird ausgelöst durch Eingaben (Fehler oder spezielle Befehle) des Benutzers während der Durchführung eines Usability-Tests. Eine weitere Möglichkeit ist das Ausfüllen eines Fragebogens direkt im Anschluss an die Produktnutzung [12]. Einen Fragebogen online automatisiert zu erstellen, zu verarbeiten und auszuwerten, ermöglicht *Google Docs* durch das Anlegen von Formularen[12].

4.2.3 Simulationsmethoden

Bei diesem Ansatz wird die Interaktion und das Verhalten eines Benutzers durch eine Software simuliert, um die Usability eines Interface automatisch testen zu können. Die Ergebnisse daraus werden beispielsweise in Form von Performance-Messungen dargestellt. Die Simulation von Benutzern eines User-Interface kann zur Erreichung unterschiedlicher Ziele eingesetzt werden. Zur Ermittlung von optimalen Navigationszeiten werden idealisierte Benutzermodelle verwendet, welche einen expliziten, vordefinierten Navigationspfad folgen. Dabei werden diverse metrische Werte, wie Lade- oder Navigationszeiten, ermittelt und festgehalten [13]. Ein Ziel ist auch die automatisierte Erzeugung von synthetischen Daten zur unterstützten Analyse von Logfiles. In [6] wird ein entwickelter Simulationsansatz zur automatisierten Generierung von Navigationswegen einer Website beschrieben. Dabei wird ein Modell einer bestimmten Website erstellt, welches Daten über inhaltliche Ähnlichkeiten der einzelnen Seiten und über die Strukturierung der Links enthält. Die Simulation modelliert einen hypothetischen Benutzer, welcher die Website beginnend bei einem Startpunkt bis zur Erreichung des definierten Ziels durchschreitet. Die gespeicherten Navigationswege des Simulators dienen dem Vergleich mit Navigationswegen von realen Benutzern.

[12] docs.google.com

Kapitel 5

Kontextsensitiver Guideline-Review

Im Zuge dieser Arbeit wurde aufgrund der gewonnenen Erkenntnisse, ein User-Interface automatisiert auf Guideline-Verstöße inspizieren zu können, eine eigene Lösungsidee entwickelt und in Form eines Prototypen umgesetzt. Das Ziel der Entwicklung lag auf der Visualisierung der Lösungsidee durch die Erstellung eines User-Interface, welches durch Benutzer evaluiert werden sollte (siehe Kapitel 6). Die Umsetzung des Prototypen orientierte sich an der Erreichung dieses Ziels und beschränkte sich auf die Evaluierung von einem Satz ausgewählter Guidelines. In diesem Kapitel werden die Ideen und Konzepte im Rahmen des Lösungsansatzes, sowie das entstandene User-Interface und die Umsetzung des Prototypen dargestellt. Zur technischen Umsetzung der Lösungsidee mussten einige Arbeitsschritte und Überlegungen zur Auswahl, Kategorisierung und technischen Abbildung der Guidelines vorgenommen werden.

5.1 Beschreibung des Interaktionskonzepts

Die Beschreibung des Interaktionskonzepts bildet einen zentralen Punkt des Lösungsansatzes und stellt die Ausgangssituation und Vorarbeit für die praktische Umsetzung dar. Es beinhaltet die Beschreibung der Lösungsidee, der Zielgruppe, sowie die Vorgehensweise bei der Auswahl und Klassifizierung der Guidelines. Ebenfalls für die Umsetzung der Lösungsidee relevant, war die Analyse moderner GUI-Builder bezüglich der Integration von Usability-Richtlinien. Der folgende Abschnitt beschreibt die Idee einer Lösungsmöglichkeit zur automatisierten Inspektion eines User-Interface während dessen Erstellungsphase.

5.1.1 Lösungsidee

Einen der wesentlichsten Punkte der Lösungsidee stellt die automatisierte Evaluierung eines User-Interface während dessen Erstellungsphase in einer integrierten Entwicklungsumgebung dar. Als Entwicklungsumgebung wurden GUI-Builder mit der Möglichkeit einer Designansicht und der Verwendung von GUI-Elementen (Radiobuttons, Checkboxes usw.) per Drag & Drop in Betracht gezogen (siehe Abschnitt 5.1.5). Der Entwickler bzw. Designer einer GUI soll während der Verwendung der GUI-Elemente und der Erstellung des Interface in der Designansicht, auf Verstöße gegen Usability-Guidelines automatisiert hingewiesen werden und Verbesserungsvorschläge erhalten. Ziel ist es, durch den kontextsensitiven Guideline-Review, den Benutzer auf Designfehler zum Zeitpunkt der Entstehung hinzuweisen und Wiederholungsfehler dadurch zu vermeiden. Als kontextsensitiv wird eine Hilfe bezeichnet, wenn diese dem Anwender unmittelbar zur Verfügung steht und sich auf ein bestimmtes Element der Oberfläche bezieht [41].

Überprüft werden soll nur die Einhaltung von Guidelines, welche im direkten Bezug mit den verwendeten GUI-Elementen stehen (siehe Abschnitt 5.1.3). Wie auch in Abschnitt 4.1.2 erwähnt, ist es aufgrund der Beschaffenheit der Guidelines nicht immer möglich, diese automatisiert evaluieren zu können. Gemäß [8] sind im besten Fall 78 % der Guidelines und im schlechtesten Fall nur 44 % operationalisierbar. Operationalisierbar ist eine Guideline, wenn die Einhaltung dieser durch eine Software automatisiert erkannt werden kann. Die Vorgehensweise bei der Unterteilung in operationalisierbare und nicht-operationalisierbare Guidelines wird in Abschnitt 5.1.4 erläutert. Um dennoch dem Benutzer die Möglichkeit bieten zu können, ein fehlerfreies Interface durch die Einhaltung aller Guidelines zu gestalten, sollen nicht-operationalisierbare Guidelines in Form von Checklisten zur Evaluierung durch den Benutzer selbst zur Verfügung stehen. Der gesamte Prozess ist somit aufgeteilt in eine automatisierte Inspektion durch die Software und eine manuelle Evaluierung durch den Benutzer selbst.

5.1.2 Zielgruppe

Die Zielgruppe der bereits beschriebenen Lösungsidee, sind all jene, die privat oder beruflich einen GUI-Builder wie *Adobe Flash Builder* oder *Microsoft Visual Studio*, zur Erstellung von grafischen Benutzeroberflächen verwenden.

Um das Wort Anwender zu spezifizieren und um eine Vorstellung von einem typischen Anwender zu bekommen, wurden vor der Erstellung des Prototypen Personas definiert. Personas sind dokumentierte Personen, welche als Anwender des Produkts in Frage kommen. Sie verfolgen ähnliche Ziele, mit ähnlicher Motivation und ähnlichem Verhalten. Die Definition der Personas basiert auf den Unterschieden dieser Charakteristika: Was machen diese Personen und warum machen sie es (Ziele und Motivation). Es emp-

fiehlt sich die Anzahl der Personas klein zu halten. Je größer die Gruppe, desto mehr unterschiedliche Ziele und Wünsche der Anwender gilt es mit dem Produkt abzudecken [34].

Zur besseren Vorstellung der Design-Konzepte wurden Szenarios erstellt und die entwickelten Personas in einen Kontext gebracht. Gemäß [34] sind Szenarios in Worte gefasste Prototypen zur Vermittlung einer Vorstellung bezüglich des Gebrauchs des entstehenden Produkts. In Anhang A werden je drei Personas und die dazugehörigen Szenarios beschrieben.

5.1.3 Auswahl der Guidelines

Evaluiert werden soll das User-Interface gegen eine Wissensbasis an Guidelines. Diese Guidelines beziehen sich ausschließlich auf die zur Verfügung stehenden GUI-Elemente. Jegliche andere Arten und Bereiche von Usability-Guidelines wurden nicht miteinbezogen. Bei der Auswahl der Guidelines wurden insgesamt vier Quellen verwendet:

- **The Essential Guide to User Interface Design** [10]: Ein Buch von Wilbert O. Galitz über gutes Screen- und Interface-Design. Die Guidelines wurden einem Kapitel, welches sich zur Gänze mit der richtigen Verwendung von GUI-Elementen befasst, entnommen.
- **GUI Bloopers** [14]: Jeff Johnson zeigt in seinem Buch häufig vorkommende Designfehler und gibt Ratschläge, wie diese vermieden werden können. Diesen Empfehlungen wurden die Guidelines für diesen Lösungsansatz entnommen.
- **Sun Web Application Guidelines – Version 4.1** [41]: Diese Sammlung von Richtlinien wurde von User-Interface-Designern und Usability-Experten zur Schaffung eines guten und konsistenten Look & Feel innerhalb einer Web-Applikation erstellt. Entstanden sind die Richtlinien ursprünglich für den Gebrauch innerhalb von *Sun Microsystems*. Gemäß den eigenen Angaben von Sun, können die Richtlinien jedoch von allen Personen bei der Verwendung einer Web-Applikation verwendet werden.
- **Windows User Experience Interaction Guidelines** [21]: Das Ziel der *Windows User Experience Interaction Guidelines* ist eine qualitativ hochwertige und konsistente Basis an Richtlinien für Windows-basierte Applikationen zu schaffen. Die Guidelines, speziell für Windows-Bedienelemente, können auch generalisiert bei der Erstellung von grafischen Benutzeroberflächen angewandt werden.

Die Wahl der vier genannten Quellen für Styleguides lässt sich zum einen mit ihrer Aktualität begründen, zum anderen beinhalten alle einen ausführlichen Teil mit Guidelines über die richtige Verwendung von GUI-Komponenten.

Für die Aufnahme einer Guideline in die Wissensbasis musste mindestens eine von zwei Voraussetzungen erfüllt sein. Die eine Voraussetzung war das Vorkommen einer Guideline in mindestens zwei der vier oben aufgelisteten Quellen. Wurde diese Voraussetzung nicht erfüllt, musste die Guideline durch die Angabe einer Quelle, welche ihre Existenz rechtfertigt, belegbar sein. Da es sich bei der im Zuge dieser Arbeit implementierten Lösung jedoch um einen Prototyp handelt, wurde nicht jede einzelne Guideline der vier genannten Quellen ganz genau analysiert, sondern es wurde eine erste repräsentative Auswahl getroffen.

Für die Erstellung des Prototypen wurden Guidelines für vier GUI-Elemente ausgewählt und kategorisiert. Dazu zählten Radiobutton, Checkbox, Button und Drop-Down-Liste.

5.1.4 Klassifizierung der Guidelines

Der erste Schritt der Klassifizierung umfasste die Zuordnung der Guidelines zu den jeweiligen GUI-Elementen. Im nächsten Schritt wurden die Guidelines je GUI-Element in messbare und und nicht-messbare eingeteilt. Die Tabelle 5.1 zeigt eine vorgenommene Kategorisierung der Guidelines für das Element Radiobutton.

Tabelle 5.1: Kategorisierung der Guidelines für Radiobuttons.

		Measureable								Immeasureable		
	Guidelines	Available	Default	Alignment	Distance	Number	Size	Action	Sorted	Right Element	Sorted	Text/ Meaning
Radiobutton	Every group of radiobuttons needs to be labeled.	x										
	You should have one radiobutton selected by default.		x									
	Prefer to align radiobuttons vertically instead of horizontally.			x								
	Provide adequate separation between choices.				x							
	Never use a single radiobutton.					x						
	Don´t use the selection of a radiobutton to perform a command.							x				
	It is recommendable not to use more than 8 options in a group.					x						
	Radiobuttons should be used to choose only one choice from a list.									x		
	The data should be small and fixed in number.									x		
	The options should be important enough to be a good use of the screen space.									x		
	The options of a group are listed in a logical order.										x	
	All labels should be written as a word or phrase, not as a sentence.											x

Zur Klassifizierung wurden anhand der Eigenschaften der Guidelines Unterkategorien gebildet. Wurde eine Unterkategorie als messbar eingestuft, d.h. sie kann durch die Software evaluiert werden, wurde sie der entsprechenden Hauptkategorie *Measureable* zugeteilt. Die Bewertung, ob eine Guideline als messbar eingestuft werden konnte, erfolgte demnach durch die Zuordnung zu einer Unterkategorie. Diesem Schema entsprechend wurde jede einzelne Guideline einem GUI-Element und einer der beiden Hauptkategorien zugeordnet. Die Überschriften *Measureable* und *Immeasureable* der Tabelle 5.1,

bilden die beiden Hauptkategorien ab. Darunter befinden sich die jeweiligen Unterkategorien. Die Zuordnung einer Guideline in eine Unterkategorie wird durch ein Kreuz dargestellt. In Anhang B wird die vorgenommene Kategorisierung aller vier Elemente in tabellarischer Form dargestellt.

Im Bezug auf die Aussage bezüglich der Operationalisierbarkeit von Guidelines in [8], konnten für die in die Wissensbasis aufgenommenen Guidelines der vier GUI-Elemente, die gezeigten Werte in Tabelle 5.2 ermittelt werden.

Tabelle 5.2: Angabe der Prozentwerte nach erfolgter Kategorisierung.

	Anzahl	Messbar	Nicht-Messbar
Radiobutton	12	58%	42%
Drop-Down-Liste	10	60%	40%
Button	8	63%	37%
Checkbox	8	50%	50%
Gesamt	**38**	**58%**	**42%**

Diese Werte repräsentieren die vorgenommene Kategorisierung im Zuge dieser Arbeit und stellen aufgrund der prototypischen Auswahl der Guidelines nur Richtwerte dar.

5.1.5 Analyse moderner GUI-Builder

Wie in Abschnitt 5.1.1 bereits erklärt, ist der Kernpunkt der Lösungsidee die automatisierte Evaluierung eines User-Interface während der Erstellungsphase in der Designansicht einer Entwicklungsumgebung. In Betracht gezogen wurden Entwicklungsumgebungen mit integrierten Editoren für grafische Benutzeroberflächen, d.h. mit der Möglichkeit der Verwendung von GUI-Elementen (Radiobuttons, Checkboxes usw.) per Drag & Drop. Solche grafischen Editoren werden auch oft als WYSIWYG[1]-Designer bezeichnet. Als gut geeignet für die Integration der entwickelten Lösungsidee, wurden folgende Entwicklungsumgebungen identifiziert:

- *Adobe Flash Builder*[2]: Der *Adobe Flash Builder* ist eine Entwicklungsumgebung für die Erstellung von Mobile-, Web- und Desktop-Applikationen durch die Programmiersprache *ActionScript 3.0* und dem Opens-Source-Framework *Flex*.
- *Microsoft Visual Studio*[3]: *Visual Studio* ist eine von *Microsoft* entwickelte integrierte Entwicklungsumgebung für verschiedene Hochsprachen. Die Entwicklungsumgebung bietet dem Benutzer, bei der Erstellung von Projekten mit einer grafischen Benutzeroberfläche, Unterstützung durch einen dementsprechenden Editor.

[1] What You See Is What You Get
[2] http://www.adobe.com/products/flash-builder.html
[3] http://www.microsoft.com/germany/VisualStudio/

Jeff Johnson erläutert in [14] die Bedeutung von Entwicklungsumgebungen bei der Erstellung von User-Interfaces, welche mit generellen oder plattformspezifischen Standards und Usability-Guidelines konform sind. Dabei betont er auch bei der Auswahl einer Entwicklungsumgebung darauf zu achten, ob der Benutzer bei der Einhaltung von Standards und Guidelines während der Erstellung einer GUI Unterstützung erhält.

Die beiden genannten Entwicklungsumgebungen wurden bezüglich der Unterstützung eines Designers oder Programmierers bei der Einhaltung von Richtlinien analysiert. Als Referenz für bestehende Guidelines wurden die vier Quellen aufgelistet in Abschnitt 5.1.3 herangezogen.

Hilfestellung bei der Einhaltung von Standards erhalten die Benutzer vor allem durch das festgelegte Aussehen der zur Verfügung stehenden grafischen Komponenten. Gerade *Visual Studio* ermöglicht durch das Grafik-Framework *WPF*[4] die Erstellung von Applikationen im Windows-Look & Feel. Erreicht wird dieser Look & Feel beispielsweise durch vorgefertigte *Message Boxes* oder zahlreiche unterschiedliche *Dialog Boxes* (siehe Abbildung 5.1)[5].

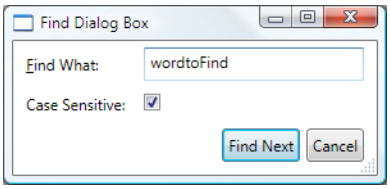


Abbildung 5.1: Eine *Dialog Box* erstellt mit der Grafikbibliothek *WPF*.

Auch das Aussehen von einfachen Steuerelementen wie z.B. von Checkboxes, unterstützt die Einhaltung von Usability-Guidelines und Standards. Bei den Checkboxes befindet sich das Label gemäß Usability-Guidelines, bereits vordefiniert immer rechts neben dem Icon. Auch bei der Begutachtung von Buttons konnten integrierte Guidelines festgestellt werden. So wird bei der Vergabe eines Labels, welches die Standardgröße eines Buttons übersteigt, die Breite des Buttons automatisch angepasst, so dass der gesamte Text lesbar bleibt. Dabei wird auch ein konsistenter Abstand zwischen dem Text und dem Button-Rand eingehalten. Bei Vergabe einer extrem kurzen Beschriftung wird im *Adobe Flash Builder* die Standardbreite von 70 Pixel jedoch nicht unterschritten. Ebenfalls Unterstützung erhalten die Benutzer bei der Ausrichtung und Anordnung von Komponenten durch Abstandhalter und das Erscheinen von Hilfslinien.

Zusammenfassend kann gesagt werden, dass die analysierten Entwicklungsumgebungen die Einhaltung von Standards und Guidelines durch das vordefinierte Aussehen der GUI-Komponenten und durch die Hilfestellung

[4]Windows Presentation Foundation
[5]Entnommen aus: http://msdn.microsoft.com/en-us/library/aa969773.aspx

bei deren Ausrichtung unterstützen. Darüber hinaus konnte keine Unterstützung oder Hilfestellung festgestellt werden. Auffällig war, dass in der Designansicht beider Entwicklungsumgebungen keine Beschreibung der GUI-Komponenten oder über deren richtige Verwendung gefunden werden konnte.

5.2 Darstellung des Interaktionsprototypen

Um die Integration eines automatisierten Guideline-Reviews in eine Entwicklungsumgebung veranschaulichen zu können, wurde ein Prototyp implementiert, welcher die wichtigsten Funktionalitäten eines grafischen Editors einer wie in Abschnitt 5.1.5 beschriebenen Entwicklungsumgebung beinhaltet. Die Funktionalität beschränkt sich auf folgende Komponenten: Der Prototyp beinhaltet GUI-Komponenten, welche per Drag & Drop auf die Bühne gezogen werden können. Werden auf der Bühne liegende Elemente markiert, können Eigenschaften für die jeweils markierten Elemente gesetzt werden.

Die Darstellung der entwickelten Lösungsidee und folglich die Benutzbarkeit der integrierten Hilfestellung, sollten durch die Umsetzung des Prototypen erreicht werden. Dazu wurde der zuvor ausgewählte und danach kategorisierte Pool an Guidelines herangezogen (siehe Abschnitt 5.1.3 und 5.1.4). Die Umsetzung der als messbar eingestuften Guidelines ist nicht vollständig und erfolgte exemplarisch.

Der weitere Verlauf des Kapitels zeigt das fertige Ergebnis des User-Interface und beinhaltet Überlegungen und Begründungen, die zu diesem Ergebnis führten. Abbildung 5.2 zeigt den prototypisch umgesetzten grafischen Editor einer Entwicklungsumgebung mit integrierter Hilfestellung zur Einhaltung von Usability-Guidelines.

5.2.1 Automatisierter Guideline-Review

Im Folgenden wird der Funktionsumfang der Hilfestellung zur Durchführung eines automatisierten Guideline-Reviews erläutert. Zu diesem Zweck wurde beispielhaft ein User-Interface mit Verstößen gegen Usability-Guidelines erstellt (siehe Abbildung 5.2).

Die gelben Dreiecke mit Rufzeichen im erstellten User-Interface geben dem Benutzer den Hinweis, dass Verstöße gegen Usability-Guidelines in seinem Design vorliegen. Während der Benutzer sein Interface erstellt, führt die Software im Hintergrund automatisch einen Guideline-Review durch. Der Guideline-Review erfolgt gegen einen Teil der in Abschnitt 5.1.4 als messbar eingestuften Guidelines. Sobald ein Verstoß – ausgelöst durch Aktionen des Benutzers (siehe Abschnitt 5.2.4) – festgestellt werden kann, erscheint das Symbol und wird dem jeweiligen Element oder der Gruppe von Elementen zugeordnet. Die Wahl des Symbols begründet sich durch vorhandene Guidelines für *Warning Messages* in [21].

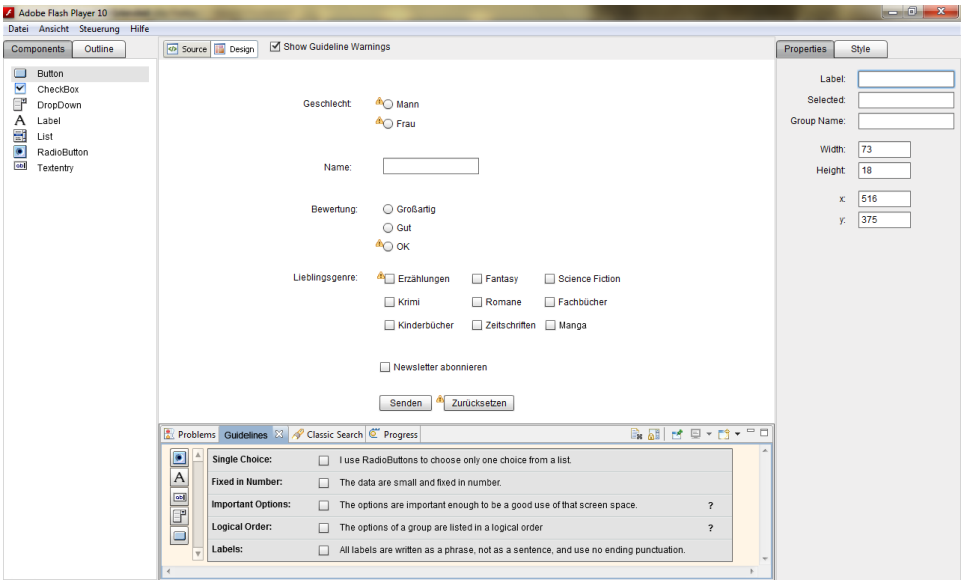


Abbildung 5.2: Prototypisch umgesetzter grafischer Editor mit integrierter Hilfestellung zur Einhaltung von Usability-Guidelines.

Durch den Klick auf ein Rufzeichen, öffnet sich ein Fenster mit einer Auflistung gebrochener Guidelines durch das jeweilige Element oder die jeweilige Gruppe. Bei der Radiobutton-Gruppe auf Abbildung 5.3 konnten zwei Verstöße festgestellt werden.

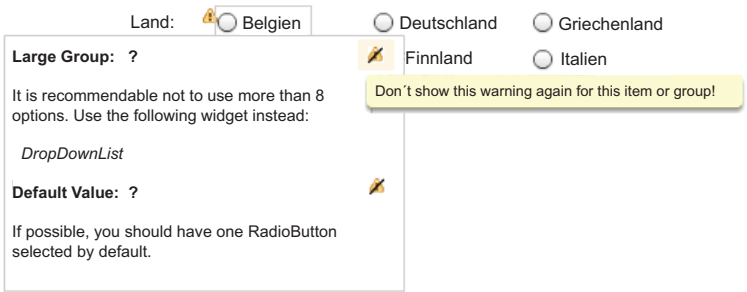


Abbildung 5.3: Auflistung von zwei gebrochenen Guidelines nach durchgeführtem automatisierten Guideline-Review.

Zum einen wurde diese Gruppe als *Large Group* identifiziert, was bedeutet, dass die Gruppe aus mehr als acht Optionen besteht. Zum anderen erkannte die Software, dass vom Benutzer keine Default-Option gesetzt wurde. Die gebrochene Guideline wird dabei textuell als Empfehlung formuliert. Durch die Verwendung des Fragezeichens besteht die Möglichkeit, in einer

Detailansicht noch weitere Informationen zu dieser Guideline zu erhalten. Entschließt sich der Benutzer der Empfehlung einen Default-Wert zu setzten nachzukommen, verschwindet dieser Hinweis nach durchgeführter Aktion aus der Liste. Wird allen Empfehlungen aus der Liste nachgekommen, sind somit keine Verstöße gegen Guidelines, welche automatisch erkannt werden können, vorhanden und das gelbe Dreieck verschwindet.

Individuelle Überlegungen des Benutzers zu seinem Design können dazu führen, dass er sich gegen die Einhaltung einer empfohlenen Guideline entscheidet. Diese Entscheidung wird durch das durchgestrichene Dreieck unterstützt. Durch einen Klick auf dieses Symbol, verschwindet die entsprechende Guideline aus der Liste. Der Benutzer legt dadurch ausdrücklich fest, dass er zukünftig diese Guideline für diese Gruppe nicht mehr angezeigt haben möchte. Wird demnach ein Verstoß gegen diese weggeklickte Guideline festgestellt, wird diese auch nicht wieder in die Liste aufgenommen.

Die Hilfestellung bietet neben den Empfehlungen zur Einhaltung einer Guideline, auch die Möglichkeit das Interface automatisch zu modifizieren. Auf Abbildung 5.3 erhält der Benutzer die Empfehlung anstatt der großen Gruppe mit Radiobuttons eine Drop-Down-Liste zu verwenden. Durch einen Klick auf die Schaltfläche wird die Radiobutton-Gruppe automatisch in eine Drop-Down-Liste umgewandelt. Dazu wird eine neue Drop-Down-Liste mit den Beschriftungen der einzelnen Radiobuttons als Datenanbinder erzeugt (siehe Abbildung 5.4).

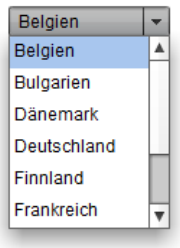


Abbildung 5.4: In eine Drop-Down-Liste umgewandelte Radiobutton-Gruppe von Abbildung 5.3.

Die Checkbox *Show Guideline Warnings* auf Abbildung 5.2 ermöglicht das Ein- und Ausblenden der gelben Warnungen. Durch diese Funktion kann der Benutzer bei Bedarf die Warnungen ausblenden und sein Design ohne Unterbrechungen erstellen.

5.2.2 Checkliste

Bei der im Vorfeld durchgeführten Klassifizierung der Guidelines (Abschnitt 5.1.4), wurde eine Guideline als nicht-messbar eingestuft, wenn ein Verstoß gegen sie nicht automatisch von einer Software erkannt werden kann. Jene

Guidelines werden pro Element in Form einer Checkliste abgebildet. Abbildung 5.5 zeigt die Checkliste für das Element Radiobutton.

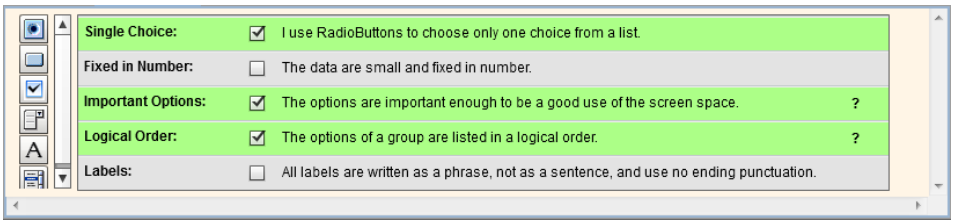


Abbildung 5.5: Checkliste für die Durchführung eines manuellen Guideline-Reviews durch den Benutzer.

Mit Hilfe der Checkliste muss der Benutzer selbst einen Guideline-Review durchführen. Dabei sollte das erstellte Interface im Hinblick auf die Einhaltung jeder Guideline in der Checkliste inspiziert werden. Wird kein Bruch gegen eine Guideline festgestellt, so kann diese in der Liste abgehakt werden. Durch die farbliche Änderung von grau auf grün soll auf einen Blick erkennbar sein, für welche Guidelines die Inspektion bereits erfolgte und für welche noch nicht. Eine vollständig grüne Checkliste signalisiert, dass für das entsprechende Element alle Guidelines im User-Interface eingehalten werden.

5.2.3 Detailansicht

Die Detailansicht einer Guideline dient deren ausführlichen Erläuterung. In die Detailansicht kann durch einen Klick auf das zu einer Guideline gehörige Fragezeichen gelangt werden (siehe Abbildung 5.3 und Abbildung 5.5). Dieses ist vorhanden für Guidelines mit Erklärungsbedarf oder für Guidelines, welche durch weitere verfeinerte Guidelines genauer spezifiziert werden.

Das Ziel ist, dem Benutzer auf Wunsch weitere Informationen zu bieten und beim Verständnis einer Guideline zu unterstützen. Die Darstellung der Informationen erfolgt textuell und wird durch Grafiken unterstützt. Abbildung 5.6 zeigt eine Detailansicht einer Guideline mit der Kurzbezeichnung *Important Options*.

5.2.4 Automatisierte Fehlererkennung

Der Anstoß zur Überprüfung eines Elements oder einer Gruppe von Elementen bezüglich der Einhaltung von Guidelines, kann durch unterschiedlich gesetzte Aktionen des Benutzers ausgelöst werden:

- Die Erzeugung gewisser Elemente: Dies geschieht entweder indem der Benutzer ein Element per Drag & Drop auf die Bühne zieht oder durch die Transformation eines bereits auf der Bühne liegenden Elements in ein anderes Element.

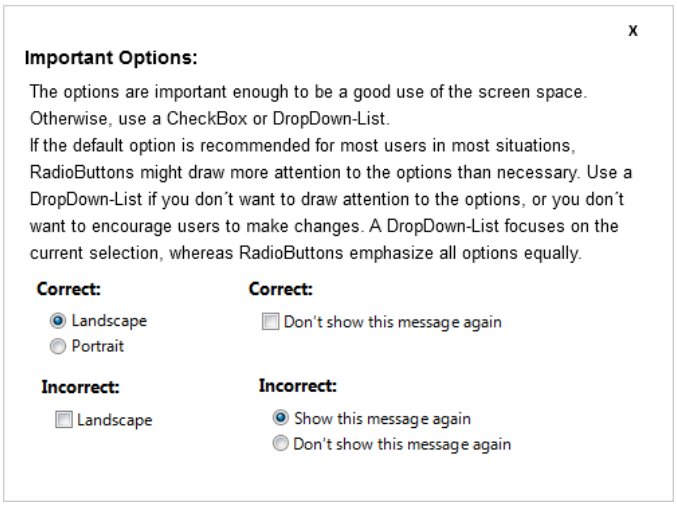


Abbildung 5.6: Detailansicht einer Guideline.

- Das Setzen oder Ändern von Eigenschaften eines Elements: Dies beinhaltet alle Eigenschaften die, durch die in Abbildung 5.2 gezeigte Registerkarte *Properties*, gesetzt werden können. Beispiele dafür sind:
  - Breite, Höhe
  - X-Position, Y-Position
  - Vergabe eines Labels
  - Setzen von Default-Werten
  - Erstellen einer Gruppe
  - Setzen von Events
  - Anbindung eines Datenanbieters
- Die Bewegung eines Elements durch den Benutzer: Bewegt der Benutzer ein Element mit der Maus, ändert sich die X- und Y-Position.

Wird eine dieser Aktionen vom Benutzer ausgeführt, führt die Software für das betreffende Element bzw. für die betreffende Gruppe an Elementen einen Guideline-Review durch. Wird ein Verstoß gegen eine Guideline erkannt, so erscheint dem Element zugeordnet ein gelbes Dreieck.

5.3 Struktur der Applikation

Die Umsetzung des Prototypen erfolgte mit *Adobe Flash Builder* und *Flex SDK* Version 4. Der Aufbau erfolgte in Anlehnung an das Architekturmuster *Model-View-Presenter (MVP)*, was eine einfache Erweiterbarkeit der Applikation ermöglichen soll. *MVP* geht aus dem Entwurfsmuster *Model-View-Controller (MVC)* hervor und beschreibt einen Ansatz, bei dem das Model

und die View komplett voneinander getrennt und über einen Presenter verbunden sind. Abbildung 5.7 zeigt den Ausschnitt des Aufbaus der Applikation, welcher für die Umsetzung der integrierten Hilfestellung zur Einhaltung von Guidelines relevant war und beschränkt sich auf die GUI-Komponenten Radiobuttons und Checkboxes. Für alle weiteren GUI-Komponenten ist der Aufbau ident. Die beinhalteten Klassen sowie das verwendete Architekturmuster, werden im weiteren Verlauf dieses Abschnitts beschrieben.

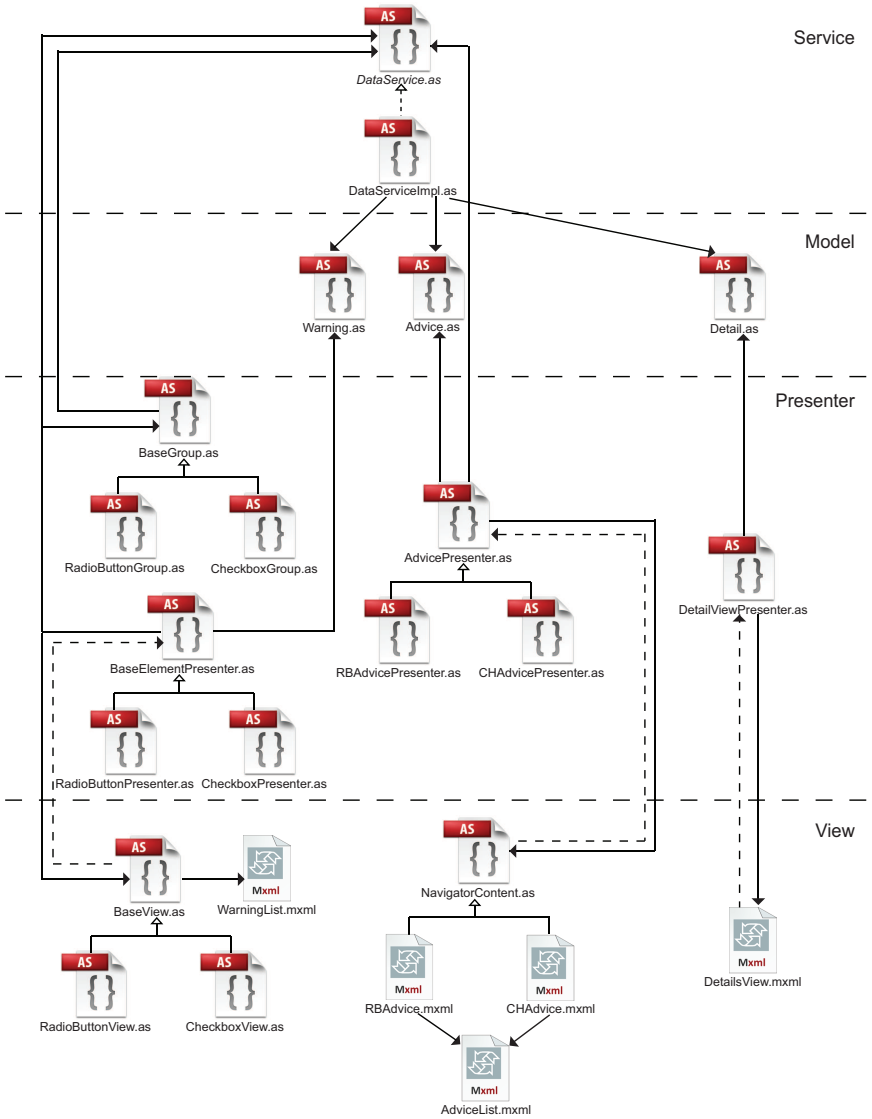


Abbildung 5.7: Der Ausschnitt des Aufbaus der erstellten Applikation zeigt den für die integrierte Hilfestellung relevanten Teil der Umsetzung.

5.3.1 Service

Um mit den für den Prototypen ausgewählten Guidelines arbeiten zu können, musste die in Abschnitt 5.1.4 beschriebene Klassifizierung technisch abgebildet werden. Zur Haltung und Verwaltung der Daten der Guidelines wurden die folgenden zwei Klassen implementiert.

DataService.as

Dieses Interface definiert die erforderlichen Methoden zur Bereitstellung der Daten.

DataServiceImpl.as

Je Element (Radiobutton, usw.) existiert eine Liste zur Haltung der messbaren Guidelines (Objekte des Typs Warning) und eine zweite Liste zur Haltung der nicht-messbaren Guidelines (Objekte des Typs Advice). Durch einen initialen Methodenaufruf werden die Objekte angelegt und die beiden Listen befüllt. Die Klasse implementiert das Interface DataService.as und enthält je Element eine Methode zur Rückgabe der gesamten nicht-messbaren Guidelines und eine Methode zur Rückgabe einer bestimmten messbaren Guideline. Jede Guideline besitzt einen eindeutigen Namen, welcher als Schlüssel zur Identifikation dient. Die Daten Name, Kurz- und Detailbeschreibung der Guidelines sind direkt in den Programmcode eingebettet. Zur Darstellung einer Detailansicht, enthält die Klasse eine Instanz von Detail.as.

5.3.2 Model

Die folgenden Klassen dienen der Haltung von Daten zur Laufzeit, welche von der entsprechenden View angezeigt werden. Die Steuerung des jeweiligen Models erfolgt allein vom Presenter. Das Model enthält keine Geschäftslogik und kennt weder die View noch den Presenter.

Warning.as

In einem Objekt dieser Klasse werden im Wesentlichen der Name sowie eine Kurz- und Detailbeschreibung einer als messbar eingestuften Guideline gespeichert.

Advice.as

Ein Objekt der Klasse Advice.as speichert Name, Kurz- und Detailbeschreibung einer als nicht-messbar eingestuften Guideline.

Detail.as

Fordert ein Benutzer eine Detailansicht einer Guideline durch einen Klick auf ein Fragezeichen an, werden in einer Instanz dieser Klasse die relevanten Daten gespeichert.

5.3.3 Presenter

Die Klassen, welche auf Abbildung 5.7 unter dem Begriff Presenter zusammengefasst wurden, beinhalten die gesamte Logik der Applikation. Presenter sind das Bindeglied zwischen Model und View und steuern die Abläufe zwischen den beiden Schichten. Sie reagieren auf Aktionen des Anwenders, Verwalten die Model-Klassen und steuern die Anzeige der View.

BaseElementPresenter.as

Von BaseElementPresenter.as sind alle Klassen abgeleitet, welche für die Verwaltung einer bestimmten GUI-Komponente zuständig sind. Diese abgeleiteten Klassen bilden das Kernstück des automatisierten Guideline-Reviews. BaseElementPresenter.as enthält die gesamte Funktionalität zur Steuerung der Anzeige für den automatisierten Guideline-Review (siehe Abbildung 5.3). Dies beinhaltet folgende Aufgaben:

- Ein- und Ausblenden des Warnung-Symbols und der Liste mit Warnungen (dargestellt durch WarningList.mxml): Sobald die Liste eine Warnung enthält, wird durch die Klasse BaseView.as das Symbol eingeblendet.
- Verwaltung der Warnungen: Dies beinhaltet das Hinzufügen einer Warnung zur Liste bei einem erkannten Verstoß, sowie das Entfernen einer Warnung aus der Liste bei erfolgreicher Einhaltung der Guideline.
- Speichern der Warnungen: Warnungen, welche auf Wunsch des Anwenders nicht mehr angezeigt werden sollen, werden gespeichert.
- Bilden von Gruppen: Definiert der Anwender die Zugehörigkeit eines Elements zu einer Gruppe, wird ein Objekt des statischen Typs BaseGroup erzeugt. Das jeweilige Element wird dieser Gruppe zugewiesen.
- Anstoß zum Erzeugen einer Detailansicht: Erfolgt ein Klick auf ein Fragezeichen, wird die entsprechende Methode im DataService aufgerufen.
- Setzen von Eigenschaften der jeweiligen View (RadioButtonView.as, CheckboxView.as usw.): Dies betrifft Eigenschaften, wie Position und Größe, welche alle Elemente besitzen.

RadioButtonPresenter.as

Jeder erzeugte Radiobutton auf der Bühne (RadioButtonView.as) besitzt seinen eigenen RadioButtonPresenter, welcher die Anzeige des Elements und

die Anzeige der Hilfestellung übernimmt. Diese Klasse, abgeleitet von BaseElementPresenter.as, enthält Methoden zur Überprüfung, ob die für Radiobuttons relevanten Guidelines von der View eingehalten werden. Kann ein Verstoß gegen eine Guideline festgestellt werden, wird die entsprechende Guideline von der Klasse DataServiceImpl.as durch einen eindeutigen Schlüssel angefordert. Die Rückgabe, ein Objekt der Klasse Warning.as, wird in der Liste mit Warnungen gehalten. Enthält eine Guideline metrische Grenzwerte, werden solche – als Konstanten definiert – im Programmcode gehalten. Weitere Aufgaben der Klasse sind die Transformation des Elements in ein anderes Element und das Setzen von elementspezifischen Eigenschaften der ihr zugehörigen View.

BaseGroup.as

BaseGroup.as beinhaltet den Namen der erzeugten Gruppe und eine Liste aller zur Gruppe gehörigen Elemente. Die Klasse ermöglicht das Hinzufügen zu einer Gruppe und das Entfernen eines Elements aus einer Gruppe. Zu einer Gruppe zusammengefasst werden können alle Elemente, für die diese Funktionalität standardmäßig vorgesehen ist oder Elemente, für die gruppenspezifische Guidelines existieren.

RadioButtonGroup.as

Über die Funktionalität der Elternklasse BaseGroup.as hinaus, ermöglicht RadioButtonGroup.as die Evaluierung einer Gruppe von Radiobuttons gegen die Einhaltung von Guidelines. Metrische Grenzwerte einer Guideline werden als Konstante definiert in der Klasse gehalten. Bei einem festgestellten Bruch einer Guideline, ist der folgende Ablauf gleich dem in der Klasse RadioButtonPresenter.as.

AdvicePresenter.as

Zieht der Benutzer ein GUI-Element erstmalig auf die Bühne, wird eine Checkliste mit Guidelines für das entsprechende Element der View hinzugefügt. Werden alle Elemente eines bestimmten Typs von der Bühne gelöscht, verschwindet auch die entsprechende Checkliste. AdvicePresenter.as übernimmt als Basisklasse diese Funktionalität und stößt, durch den Aufruf einer Methode im DataService, die Anzeige einer Detailansicht an.

RBAdvicePresenter.as

Diese Klasse ist abgeleitet von AdvicePresenter.as und übernimmt die Befüllung der Checkliste, angezeigt durch RBAdvice.mxml, mit Daten. Dazu wird eine Liste mit Objekten der Klasse Advice.as von DataService.as angefordert und gespeichert.

DetailViewPresenter.as

DetailViewPresenter.as steuert die Anzeige der Detailansicht einer Guideline. Die Klasse fängt die Events, ausgesendet von Detail.as, in einer Methode auf und zeigt die gespeicherten Daten des Models in der DetailsView.mxml an.

5.3.4 View

Klassen, welche der Anzeige von Daten oder Elementen dienen, werden hier unter dem Begriff View zusammengefasst. Die View enthält keinerlei Geschäftslogik und hat keinen Zugriff auf das Model. Die gesamte Steuerung der Ansicht in der View erfolgt vom Presenter. Da es sich hierbei um Klassen ohne enthaltende Logik handelt, wird auf eine textuelle Beschreibung in Folge verzichtet. Welche Daten und Elemente durch die jeweiligen Klassen der View angezeigt werden, kann aus den Abbildungen des Abschnitts 5.2 und Beschreibungen der anderen Klassen entnommen werden.

Kapitel 6

Empirische Evaluierung

Nach der Darstellung des eigenen Lösungsansatzes beschreibt dieses Kapitel die Evaluierung des Prototypen. Zur Evaluierung der Alltagstauglichkeit und der umgesetzten Konzepte wurde eine Studie durchgeführt. Im Folgenden wird zunächst die durchgeführte Untersuchung zur Evaluierung des Prototypen und dessen Konzepte beschrieben. Die Ergebnisse resultierend aus dieser Untersuchung werden anschließend dargestellt und interpretiert.

6.1 Ziele

Das Hauptziel dieser Studie war die Evaluierung der im Prototypen praktisch umgesetzten Ideen der Designphase und die dahinterliegenden Konzepte. Fragen die in diesem Zusammenhang beantwortet werden sollten, sind folgende:

- Werden die implementierten Konzepte von den Benutzern verwendet?
- Wie werden diese von den Benutzern verwendet?
- Werden die Konzepte von den Benutzern verstanden?
- Erfüllen die Konzepte ihren vorgesehenen Zweck?

Ein wichtiger Aspekt dabei war die Überprüfung der Alltagstauglichkeit und des Mehrwerts der Konzepte. Die Ergebnisse sollen belegen, ob ein solches Produkt von der Zielgruppe verwendet werden würde und welche Stärken und Schwächen sie in den einzelnen Funktionen sehen. Die Herausarbeitung dieser Vor- und Nachteile dient als Grundlage für anschließende Empfehlungen für die Entwicklung solcher Produkte.

Die Ermittlung von zusätzlich erwünschten Funktionalitäten war ein weiteres Ziel dieser Studie. Nicht im Vordergrund stand die Evaluierung der grafischen Präsentation des Prototypen.

6.2 Methode

Zur Durchführung der Studie wurde die Methode *Lautes Denken* gewählt. Bei dieser Methode verbalisiert eine Testperson ihre Gedanken während der Verwendung eines Produktes. Indem die Benutzer ihre Gedanken laut aussprechen, ist es möglich, Rückschlüsse auf Eindrücke, Wünsche oder Probleme zu ziehen. Die Probanden werden gebeten, eine Aufgabe zur Evaluierung der Mensch-Computer-Schnittstelle zu bearbeiten [24]. Wie oben beschrieben, war das Ziel der Studie die Evaluierung der Konzepte, Überprüfung der Alltagstauglichkeit und Eruierung von Wünschen und Bedürfnissen der Benutzer. Da auch [9] und [24] den Einsatz der Methode als Erprobung der Gebrauchstauglichkeit der bereits erstellten Materialien beschreiben, schien die Wahl dieser Methode als geeignet. Während der Durchführung werden die Benutzer durch gezieltes Fragen des Testleiters angeregt ihre Gedanken zu äußern. Solche Fragen können sein: „Was denken Sie gerade?“ oder „Was denken Sie, was das bedeutet?“ [24].

Um die geäußerten Gedanken festzuhalten, ist es empfehlenswert, diese per Video- oder Audiomitschnitt aufzuzeichnen. Durch die Aufzeichnung muss sich der Testleiter keine Notizen während der Durchführung machen und kann sich so besser auf das Geschehen und die Motivation der Probanden konzentrieren. Um Themenfelder zu vertiefen, kann im Anschluss an die Durchführung anhand eines Leitfadens ein Interview geführt werden. Zur Auswertung und Dokumentation des Tonmaterials ist es empfehlenswert, diese schriftlich zu erfassen. Die Transkripte dienen als Auswertungsgrundlage, werden zusammengefasst und in einer Übersicht dargestellt [9]. In Abschnitt 6.4 wird unter dem Punkt *Auswertung* genauer auf die Methode der qualitativen Inhaltsanalyse eingegangen.

6.3 Teilnehmer

Für die Studie wurden fünf Testpersonen (drei männlich, zwei weiblich) rekrutiert. Im Bezug auf die Anzahl der benötigten Probanden, können gemäß [26] und [38] mit vier bis sechs Personen verlässliche Ergebnisse erzielt werden. Die ausgewählten Personen sind zum Zeitpunkt der Studie zwischen 22 und 28 Jahre alt und studieren an der Fachhochschule Hagenberg. Im Hinblick auf die Zielgruppe und die entwickelten Personas des Produktes (siehe Abschnitt 5.1.2) wurde versucht, eine repräsentative leistungsheterogene Auswahl der Testpersonen zu treffen. Die Studenten kommen aus den Studienrichtungen *Software Engineering*, *Kommunikation Wissen & Medien* und *Interactive Media* und sind teilweise bereits berufstätig. Alle diese Personen verwenden privat oder beruflich Entwicklungsumgebungen wie *Adobe Flash Builder* oder *Microsoft Visual Studio* und kommen somit als potentielle Anwender der entwickelten Lösung in Frage.

Die Probanden wurden im Vorfeld der Studie gebeten, ihr Wissen über Design-Guidelines einzuschätzen. Abbildung 6.1 zeigt die Vorkenntnisse der Probanden gemäß eigenen Angaben. Die Buchstaben A-E repräsentieren die Testpersonen.

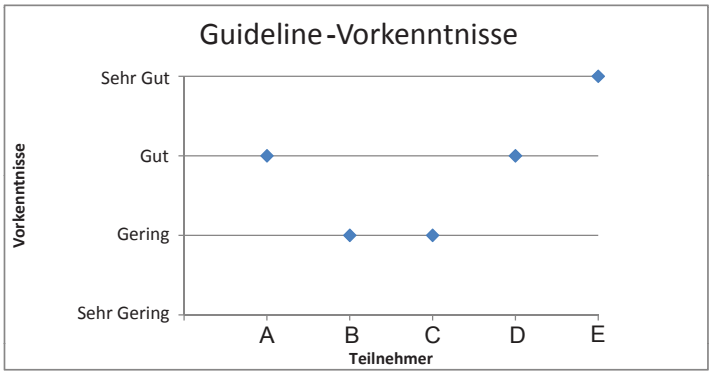


Abbildung 6.1: Eigene Einschätzung des Wissens der Testpersonen bezüglich Design-Guidelines vor Durchführung der Untersuchung.

6.4 Durchführung

Das Ablaufschema und die Durchführung der Studie erfolgte in Anlehnung an die Empfehlungen in [9] und [24] wie auch in Abschnitt 6.2 zusammengefasst beschrieben. In der folgenden Auflistung werden die einzelnen Phasen genau erläutert:

- **Einführung:** Die Einführungsphase diente vor allem dazu, die Testpersonen auf die bevorstehende Studie vorzubereiten und um ihnen Orientierung zu vermitteln. Im ersten Schritt wurden sie gebeten eine Einverständniserklärung zur Teilnahme an der Studie durchzulesen und zu unterschreiben. Mit dieser Unterschrift bestätigte der Proband die freiwillige Teilnahme an der Untersuchung. Des Weiteren wurde im Zuge dessen beschrieben, was die Testperson im Verlauf der Untersuchung erwartet und das alle erhobenen Daten vertraulich behandelt werden.

  Der Arbeitsplatz bestand aus einem Laptop auf dem der Prototyp ausgeführt wurde, einem Aufnahmegerät und den ausgedruckten Aufgabenstellungen. Die Testpersonen bekamen alle eine kurze inhaltliche Einführung über Ziel und Zweck des Prototypen. Eine Demonstration sollte das Verständnis und Gespür für das Produkt und dessen Ziele verstärken.

  Danach wurden dem Probanden die Ziele der Studie verdeutlicht. Das

Wissen der Probanden über die Ziele der Evaluierung und deren Schwerpunkte erschien als besonders wichtig. So können diese ihre Äußerungen und Aussagen an den Anwendungskontext anpassen.

Nachdem den Probanden der Prototyp und das Ziel der Studie bekannt war, wurde ihnen die Untersuchungsmethode *Lautes Denken* erläutert. Als Hilfestellung bekamen Sie einen Ausdruck mit beispielhaften Aussagen wie: „...jetzt überlege ich gerade...“, „...das ist aber viel zu lesen...“ oder „...das freut mich jetzt nicht...“.

Vor der Durchführung der Untersuchung wurde dem Probanden die durchzuführende Aufgabenstellung vorgelegt und eine kurze Einleitung dazu gegeben.

- **Aufgabenstellung:** Um den Prototyp testen zu können, wurden Aufgabenstellungen definiert, welche die Probanden umsetzen sollten (siehe Anhang C.1). Das Ergebnis dieser Aufgabenstellungen war ein Formular zur Bewertung von Büchern. Anhand eines Pilottests mit einer Testperson, welche nicht an der Untersuchung teilnahm, wurde die Effektivität der Aufgaben überprüft.
- **Interview:** Im anschließenden Interview wurden zunächst kontextspezifische Fragen geklärt. Ergaben sich aus der Beobachtung oder den Äußerungen des Probanden Unklarheiten, so wurde versucht, diese im direkten Anschluss an die Versuchsdurchführung zu klären. Danach folgten Fragen zur Klärung interessanter Aspekte, zu welchen der Proband während der Durchführung keine oder wenige Aussagen treffen konnte. Da das Ziel dieser Methode ist, Informationen durch das laute Denken der Probanden zu gewinnen, wurde bewusst auf die Kurzhaltung des Interviews geachtet.
- **Auswertung:** Zur Auswertung der Aussagen der Probanden, musste das Tonmaterial in geeigneter Form aufgearbeitet werden. Dazu wurden die Audiodateien gesichert und transkribiert. Die transkribierten Texte wurden dann zu einer thematischen Übersicht zusammengefasst.

  Orientiert wurde sich hier an den Methoden der qualitativen Inhaltsanalyse nach Philipp Mayring in [20] und an der Beschreibung zur Datenanalyse in [37]. Die in Abschnitt 6.5 dargestellten Ergebnisse entstanden durch eine zusammenfassende und inhaltlich strukturierte Auswertung nach [20].

  Das Wort *zusammenfassend* bedeutet, dass die Inhalte der Transkripte so abstrahiert und generalisiert werden, dass am Ende eine übersichtliche Zusammenfassung der wesentlichen Aussagen aller Untersuchungsteilnehmer entsteht. Während der Analyse der Transkripte wurden mehrere Analyseschritte zusammengefasst. So wurde bei der Durchsicht der Transkripte Paraphrasierung, Abstrahierung, Generalisierung und Reduktion durchgeführt. Die Textstellen wurden gleich auf das angestrebte Abstraktionsniveau transformiert [20]. Ziel der in-

haltlichen Strukturierung war es, die Inhalte der Transkripte nach definierten Kategorien einzuteilen. Die zusammengefassten, abstrahierten Textabschnitte wurden den entwickelten Hauptkategorien zugeteilt und bildeten gleichzeitig neue Unterkategorien.

Die Kodierung der Transkripte erfolgte gemäß [37]. Die entstandenen Unterkategorien wurden nummeriert und in den Transkripten referenziert. Dies geschah durch das Einschließen der Textstellen in eckige Klammern.

Um festlegen zu können, wann welcher Transkriptteil einer Unterkategorie zugeteilt werden sollte, wurden sogenannte Ankerbeispiele definiert. Ankerbeispiele sind konkrete Textstellen in den Transkripten, die einer Unterkategorie zugeteilt werden und als Beispiel für diese gelten [20].

Eine Kodiereinheit wird in [20] wie folgt beschrieben:

> „Die Kodiereinheit legt fest, welches der kleinste Materialbestandteil ist, der ausgewertet werden darf, was der minimale Textteil ist, der unter eine Kategorie fallen kann."

In diesem Fall ist die Kodiereinheit jede vollständige positive oder negative Aussage eines Teilnehmers über die verwendeten Konzepte des Prototypen. Jede vollständige Aussage eines Teilnehmers über gewünschte zusätzliche Funktionalitäten und jede Aussage über die Alltagstauglichkeit des Produktes.

6.5 Ergebnisse und Interpretation

Die Ergebnisse der Studie bilden im Wesentlichen die erstellten User-Interfaces, das aufgenommene Audiomaterial, die Transkripte und die daraus strukturierten und zusammengefassten Auswertungen in Form von Tabellen. Ein Ergebnis, welches sich nicht auf Papier darstellen lässt, ist die Beobachtung der Probanden und das dadurch entwickelte Gespür für die Nutzergruppe. Im Folgenden werden die gewonnenen Ergebnisse gezeigt und erläutert.

Abbildung 6.2 zeigt ein Interface, welches durch einen Probanden im Rahmen der Untersuchung durch die Durchführung der Aufgabenstellungen entstand.

Abbildung 6.3 zeigt einen Auszug eines kodierten Transkriptes. Die Vorgehensweise der Kodierung wird in Abschnitt 6.4 unter dem Punkt *Auswertung* genauer beschrieben.

Die Tabellen in Anhang C.2 stellen die Ergebnisse der durchgeführten Untersuchung zusammengefasst dar. Die Beschriftungen der Tabellen bilden die vordefinierten Hauptkategorien. Bis auf Kategorie 1, orientiert sich die Bildung der Hauptkategorien an den einzelnen Funktionalitäten des Prototypen.

Abbildung 6.2: Erstelltes Interface durch einen Probanden im Rahmen der Untersuchung.

[…] das genau meint. Also ich lese mir da jetzt das ganze durch was da steht beim Fragezeichen **2.1]** und unten bei den Informationen. Ja das Fragezeichen sagt, dass die Auswahl wichtige Möglichkeiten sein müssen zum Anhaken, schätze ich mal. **[**Ok, das lese ich jetzt nicht alles durch weil ich mir denke, dass ist jetzt eh nicht so schwer zum Anwenden, also schaue ich es nur grob durch. **2.2]** **[**Logical Order, ja, das betrifft es in dem Fall jetzt nicht, weil es nur zwei Optionen sind. Ich hake es jetzt nicht an. Ok, ich weiß jetzt noch nicht genau, was das jetzt bedeutet, ob das für alle Anwendungsfälle gilt. **3.1]** Dann gruppiere ich sie mal, ich nenne sie mal Group-Gender und beschrifte sie. Jetzt versuche ich die Gruppe auszurichten. Dann ziehe ich ein Label auf die Bühne. Dann schaue ich unten, was bei Label steht. Ah da kann man wechseln, das hätte ich jetzt nicht gesehen so spontan. Gut da steht jetzt nichts (Prototyp). **[**Dann probier ich es mal mit Show Guideline Warnings. Dann klicke ich auf das Rufzeichen, weil bei Herr zeigt es mir eine Warning.**4.1]** Aha, man soll einen Radiobutton als default selektieren. **[**Schauen wir mal, was da steht beim Fragezeichen. Gut das weiß ich eh, was ein Default-Wert ist, das lese ich jetzt nicht. **2.2]** **[**Dann gebe ich mal den Defaultwert beim Herrn ein.**1.2]** Jetzt möchte ich eine Namenseingabe machen. Es soll eine Namenseingabe geben, dann mache ich einen Textentry und gebe ein Label dazu. Dann schauen wir mal, ob da was steht unten (Checkliste). Mittlerweile habe ich das System verstanden. **[**Ah da steht ein Tooltip, Place all Elements before checking the Guidelines. Okay dann mache ich mal alles durch und dann checke ich die Guidelines. **1.3]** Gut dann soll ich eine Auswahl treffen zwischen 3 **[…]**

Abbildung 6.3: Auszug aus einem codierten Transkript.

Abbildung 6.4 zeigt einen Screenshot des Prototypen mit den nummerierten Funktionalitäten zur Veranschaulichung der gebildeten Hauptkategorien. *Kategorie 1*, Allgemein (Tabelle C.1), beinhaltet Aussagen der Benutzer, welche sich nicht direkt auf eine Funktionalität des Prototypen beziehen.
Kategorie 2, Detailansicht (Tabelle C.2), beinhaltet alle Aussagen der Benutzer zur erweiterten Erklärung der Guidelines. Durch einen Klick auf das Fragezeichen, erhält der Benutzer die Detailansicht.

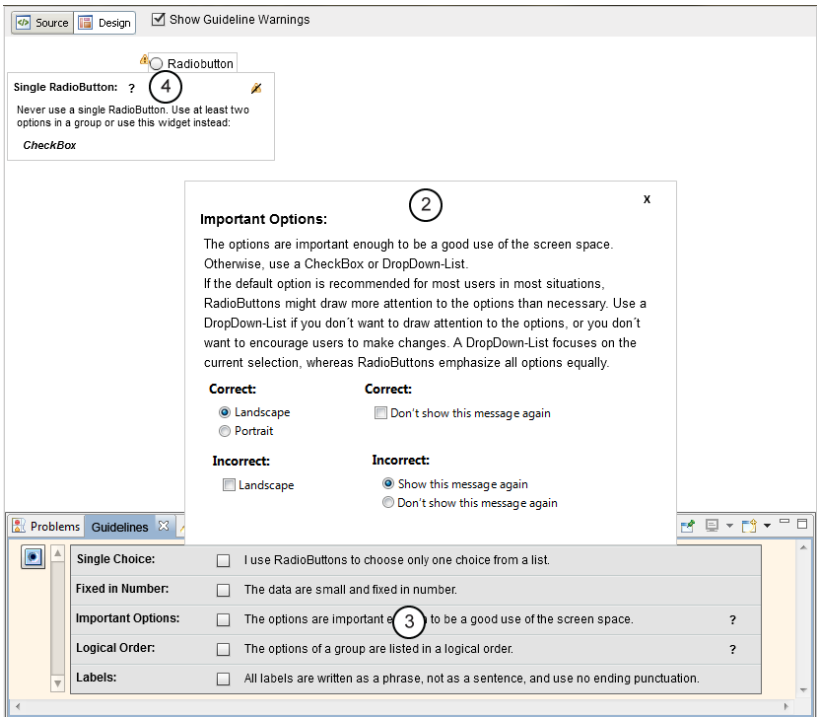


Abbildung 6.4: Abbildung der Funktionen des Prototypen. Die Nummerierung dient als Referenz zu den gebildeten Kategorien im Zuge der Auswertung der Untersuchung.

Kategorie 3, Checkliste (Tabelle C.3), beinhaltet alle Aussagen der Benutzer zu den Guidelines, dargestellt in Form von Checklisten.
Kategorie 4, Automatisierte Hilfe (Tabelle C.4), beinhaltet alle Aussagen der Benutzer zu den automatisierten, kontextsensitiven Warnungen.

Im Folgenden werden die in tabellarischer Form dargestellten Ergebnisse der durchgeführten Untersuchung, aber auch die daraus gewonnenen Eindrücke textuell zusammengefasst und interpretiert. Es soll beschrieben werden, wie diese Ergebnisse zu deuten sind und was diese für die Weiterentwicklung des Prototypen bedeuten. Die Einordnung der nun folgenden textuell beschriebenen Ergebnisse erfolgt in die bei der Auswertung gebildeten Hauptkategorien. Dabei werden die Generalisierungen der Tabellen in Anhang C.2 aufgegriffen, textuell beschrieben und interpretiert. Vorschläge für die Weiterentwicklung des Prototypen werden ebenfalls angeführt. Da der folgende Text voraussetzt die Funktionen und Konzepte der entwickelten Lösung zu kennen, empfiehlt es sich Kapitel 5 vorbereitend zu lesen.

6.5.1 Kategorie: Automatisierte Hilfe

Im Bezug auf die automatisierte Hilfestellung konnte beobachtet werden, dass alle Probanden diese während der gesamten Erstellungsphase der GUI aktiviert ließen. Die aufscheinenden Rufzeichen bei einem möglichen Bruch einer Designrichtlinie wurden bei Erscheinen sofort registriert. Ziel aller Probanden war es, ihr Design sofort von dem Rufzeichen zu befreien, indem sie entweder der dahinter liegenden Empfehlung nachkamen oder die Warnung ausblendeten. Diese Art der kontextsensitiven Hilfestellung wurde von fast allen Testpersonen als hilfreich, nicht als störend empfunden. Lediglich eine Person äußerte sich über eine mögliche Störung des Arbeitsflusses durch das aufscheinende Rufzeichen. Eine Lösung hierfür wäre, dem Benutzer eine Entscheidungsmöglichkeit über die Darstellung der Empfehlungen bzw. Warnungen zu bieten. So könnten neben der kontextspezifischen Darstellung durch das Rufzeichen, alle Warnungen in Form einer Auflistung dargestellt werden, die der Benutzer zu einem selbstgewählten Zeitpunkt abarbeiten könnte.

Wie eben erwähnt, bieten sich dem Benutzer folgende Möglichkeiten, um ein Rufzeichen aus seiner GUI zu entfernen: Eine Möglichkeit ist es, die hinter dem Rufzeichen liegende Empfehlung auszuführen und somit die Designrichtlinie zu befolgen. Die andere Möglichkeit ist die Empfehlung nicht zu beachten und die Warnung durch einen Klick auszublenden. Der Hintergedanke dieser Funktionalität, dem Anwender ein individuelles Entscheiden über die Einhaltung der Richtlinien zu ermöglichen, wurde von allen Testpersonen erkannt und als sehr positiv empfunden. So wurde es geschätzt Alternativen aufgezeigt zu bekommen, aber dennoch selbst die Entscheidung über die Einhaltung der Richtlinie treffen zu können.

Jedoch wäre es für vier der fünf Testpersonen wünschenswert, die weggeklickten Warnungen in irgendeiner Form wieder anzeigen zu können. Dies äußerte sich dadurch, dass die Testpersonen die Warnung unbeabsichtigt ausgeblendet haben oder im Nachhinein Warnungen oder Details dazu nachlesen wollten. Um dem entgegenzuwirken, könnte eine Art von History zur Darstellung ausgeblendeter Warnungen implementiert werden.

Teilweise konnte beobachtet werden, dass es für die Probanden verwirrend war, wenn eine bewusst ausgeblendete Warnung bei anderen Elementen oder Gruppen wieder erschien. Grund dafür war, dass diese Personen das Gefühl hatten, die Richtlinie nun bereits zu kennen und somit auch bei anderen Elementen keinen Hinweis zu dieser Richtlinie mehr benötigt hätten. Um auch diesem Wunsch entgegen zu kommen, könnte man dem Anwender auch hier wieder zwei Möglichkeiten bieten. Nämlich zum einen die Möglichkeit die Warnung nur für das betroffene Element ausblenden zu können und zum anderen für das gesamte Design und somit für alle Elemente der GUI.

Dass die Probanden die Richtlinien und Empfehlungen hinter den Rufzeichen teilweise im Gedächtnis behielten äußerte sich auch dadurch, dass

die Texte hinter den Rufzeichen bereits vermutet oder die Richtlinien nach einmaligem Lesen eingehalten wurden. Dies würde auch gegen eine, wie oben erwähnt, listenartige Darstellung der Designfehler sprechen. In diesem Fall würde der Benutzer eventuell erst zu einem sehr späten Zeitpunkt oder gar erst nach Fertigstellung der GUI auf seine Fehler hingewiesen werden. Ein und derselbe Fehler könnte so mehrmals im Design und müsste demnach auch im Nachhinein mehrmals behoben werden. Die Vermeidung von Wiederholungsfehlern der Probanden durch sofortiges Lesen der Empfehlungen beweist den Vorteil der kontextsensitiven Integration der Warnungen.

Die Möglichkeit, seine GUI automatisch modifizieren zu können, wurde von den Testprobanden als sehr praktisch und zeitsparend empfunden. Die automatische Umwandlung einer Gruppe von Checkboxes in eine Liste wurde sehr geschätzt. Auch bei Hinweisen über empfohlene Breiten wäre eine automatisierte Eingabe wünschenswert gewesen. Basierend auf diesen Aussagen kann festgestellt werden, dass automatische Modifizierungen erwünscht sind und noch verstärkt eingebaut werden könnten.

Fast alle Probanden äußerten Unklarheiten über die Gruppierung von Elementen zum Erhalt von gruppenspezifischen Warnungen. Es herrschte Unsicherheit darüber, ob man Checkboxes oder Buttons gruppieren müsse um eine Warnung zu erhalten. Hier wäre eine Integration einer Hilfestellung für den Anwender empfehlenswert.

6.5.2 Kategorie: Checkliste

Während der Erstellung der GUI wurden die Guidelines, dargestellt in Form von Checklisten, von vier der fünf Probanden nur kurz oder gar nicht beachtet. Erst nach Fertigstellung der GUI befassten sich die Teilnehmer mit den Checklisten und evaluierten ihr erstelltes Design gegen die angeführten Richtlinien. Nur eine der Testpersonen beachtete die Guidelines bereits vermehrt während der Erstellungsphase. Dies hatte den Vorteil, dass dieser Teilnehmer während der Erstellungsphase über mehr Informationen bezüglich Richtlinien verfügte und dieses Wissen auch umsetzen konnte. Das vollständige Durchgehen der Checkliste und Abhaken der Richtlinien erfolgte jedoch auch hier erst nach Fertigstellung der Benutzeroberfläche.

Ein Großteil der Probanden hatte Schwierigkeiten mit dem Verständnis bezüglich des Ziels und des Zwecks der Checklisten. Es war bei erstmaliger Verwendung nicht klar, warum die Richtlinien abgehakt werden sollten oder ob dies eine Aktion im Hintergrund auslösen würde. Die Teilnehmer äußerten Wünsche nach einer allgemeinen Information über das Ziel der Checklisten. Da die bestehenden Tooltips, welche die gedachte Funktionsweise teilweise beschreiben, nicht sehr beachtet wurden, ist hier eine andere Art von Hilfestellung anzudenken.

Alle Probanden äußerten den Wunsch nicht relevante Guidelines der Checklisten bearbeiten zu können. Als nicht relevant wurden Richtlinien er-

achtet, wenn diese im aktuellen Design keine Anwendung fanden oder der Benutzer sich nicht mit der Richtlinie identifizieren konnte. Dies bestätigt auch wieder den Wunsch der Benutzer, individuell über das Anwenden der Guidelines und der Empfehlungen entscheiden zu können. Um eine solche Bearbeitung zu ermöglichen, sollte der Benutzer die Möglichkeit haben, für ihn nicht relevante Guidelines deaktivieren oder wegklicken zu können. Bei einigen Elementen, wie Checkboxes oder Radiobuttons, besteht des Weiteren noch die Möglichkeit, je nach aktueller GUI, gruppenspezifische Richtlinien anzuzeigen oder wegzulassen. Dies könnte die Relevanz der Guidelines entsprechend der individuellen Benutzeroberfläche erhöhen.

Bei zwei der fünf Teilnehmer konnte beobachtet werden, dass es die Benutzer bevorzugen würden, pro Element oder Gruppe eine separate Checkliste mit Richtlinien durchgehen zu können. So wurde die eine Checkliste, welche für alle Elemente eines Typs auf der Bühne gedacht wäre, mehrmals an- und abgehakt. Da diese Arbeitsweise sehr viel Zeit in Anspruch nimmt, aber dennoch teilweise wünschenswert sein kann, sollte der Benutzer selbst über die Art der Evaluierung entscheiden können. Eine Erweiterung im Rahmen dieses Prototypen wäre dahingehend anzudenken, dem Benutzer eine Möglichkeit zu bieten, das gesamte Design nur mit einer Checkliste zu evaluieren oder die Evaluierung separat pro Gruppe oder Element durchzuführen.

Die farbliche Rückmeldung von grau auf grün nach Abhaken einer Richtlinie wurde von allen Teilnehmern als hilfreich und gut empfunden. Für vier der fünf Probanden wäre darüber hinaus noch ein weiteres Feedback wünschenswert. Eine Idee hierfür, welche noch umgesetzt werden könnte, wäre eine Darstellung des Fortschrittes in der Einhaltung der Richtlinien. Umgesetzt werden könnte das in Form eines Fortschrittsbalkens oder einer alternativen Visualisierung.

Nach Angaben der Probanden erfordert das Durchgehen der Checklisten erstmalig viel Zeit und Mühe. Jedoch konnte auch beobachtet werden, dass bei mehrmaligem Vorkommen einer Guideline, die Probanden die Richtlinie bereits anhand des fettgedruckten Schlagwortes erkannten und diese nicht mehr vollständig durchlesen mussten. Das Abhaken solcher bereits bekannten Guidelines erfordert demnach auch weniger Zeit und Mühe als das Durchgehen von noch unbekannten Guidelines. Auch die Probanden teilten diese Ansicht. Gemäß ihren Angaben wird der Aufwand für das initiale Durchgehen der Checkliste als sehr hoch eingeschätzt. Die Probanden vermuteten, dass nach mehrmaliger Anwendung jedoch nicht mehr der gesamte Text gelesen werden müsse und somit die Checklisten viel schneller abgearbeitet werden könnten.

6.5.3 Kategorie: Detailansicht

Die Detailansicht einer Guideline, welche durch einen Klick auf das Fragezeichen sichtbar wird, wurde von den Probanden aus unterschiedlichen Gründen

verwendet. Häufig wurde das Fragezeichen bei Unklarheiten über die Bedeutung der Guideline oder bei wichtigen Entscheidungen verwendet. War sich der Proband über die Bedeutung einer Guideline jedoch im Klaren, wurde diese Richtlinie auch ohne detaillierte Information umgesetzt. Ein weiterer Grund für das Anzeigen der Detailansicht war das Interesse des Teilnehmers an der Richtlinie.

Aufgrund der Tatsache, dass die Teilnehmer das Fragezeichen nur sporadisch benutzten, wurden teilweise wichtige Informationen zu den Guidelines hinter den Fragezeichen nicht gelesen. Das führte dazu, dass die GUI trotz Hilfestellung am Ende kleine Fehler beinhaltete. Abbildung 6.2 zeigt, dass der Benutzer bei der Auswahl Mann, Frau einen Default-Wert bei Mann gesetzt hat. Der Detailtext dieser Guideline enthält jedoch den Hinweis, dass ein Default-Wert in solchen Fällen nicht gesetzt werden muss.

Beim Lesen der detaillierten Beschreibung hatten die Probanden teilweise das Gefühl, keine genauere Information über die Guideline zu erhalten. Gemäß den Aussagen der Teilnehmer, sollte der Detailtext die Guideline hinter den Rufzeichen begründen oder noch eine genauere Beschreibung dieser Guideline liefern.

Des Weiteren konnte herausgefunden werden, dass die Texte oft nur grob durchgelesen oder gescannt wurden. Die Bilder mit den Beispielen zur Visualisierung der Guideline wurden von allen Probanden noch vor dem Text beachtet. Diese Beispiele wurden als sehr hilfreich und effizient empfunden. Die Länge der Texte wurde teilweise als zu lange, teilweise als passend empfunden.

In Anbetracht dieser Beobachtungen empfiehlt es sich, die detailierte Beschreibung einer Guideline eher kurz zu halten und diese mit Beispielen zu deren Veranschaulichung aufzuwerten. Die ausgewählten Informationen sollen eine Begründung für die Richtlinie liefern. Jene Informationen, welche zum richtigen Anwenden der Guideline notwendig sind, sollten aber direkt nach dem Klick auf das Rufzeichen sichtbar sein und sich nicht in der Detailansicht verstecken.

6.5.4 Kategorie: Allgemein

Als ein sehr wesentliches Ergebnis der Untersuchung kann die Erkenntnis betrachtet werden, dass es für Benutzer der Hilfestellung sehr wichtig zu sein scheint, selbst über die Anwendung einer Guideline entscheiden zu können. Konnte sich der Proband mit einer Guideline nicht identifizieren, stand diese nicht im Einklang mit dessen Erfahrungen oder deckte sich diese nicht mit seinen Zielen, wurde diese Guideline auch nicht umgesetzt. Die Formulierung der Guidelines sollte deswegen in Form von Empfehlungen oder Tipps erfolgen. Darüber hinaus, sollte dem Benutzer immer eine Möglichkeit der Individualisierung geboten werden. Der momentane Prototyp unterstützt diese Individualisierung im Bereich der automatisierten Hilfe sehr gut. Der Bereich

der Checklisten sollte, wie auch oben (unter dem Punkt *Kategorie: Checkliste*) beschrieben, um die Möglichkeit der Bearbeitung von nicht relevanten Guidelines erweitert werden.

Ein weiterer wichtiger Punkt ist die Formulierung der Texte des gesamten Produktes. Teilweise wurden die Texte der Guidelines, Tooltips oder Detailansicht inhaltlich nicht richtig verstanden. War dies der Fall, wurden die Texte von den Probanden entweder ignoriert oder bei falscher Interpretation falsch angewandt. Für den entwickelten Prototypen bedeutet diese Beobachtung die inhaltliche und sprachliche Überarbeitung der Texte.

Die Untersuchung zeigte auch teilweise Unsicherheiten der Probanden bei der richtigen Verwendung der einzelnen Funktionalitäten des Produktes. Um die Benutzer dabei zu unterstützen, sollte eine Benutzerdokumentation/-hilfe integriert werden.

Der folgende Abschnitt beschäftigt sich mit der Frage, ob und warum die Probanden eine derartige Hilfestellung für den privaten oder beruflichen Gebrauch einsetzen würden. Einen großen Vorteil sahen die Probanden darin, dass die Hilfestellung nur bei Bedarf dazu geschaltet werden kann. Für einige der Probanden wäre die Unsicherheit über das eigene Design oder über die Einhaltung von Richtlinien ein Grund für die Aktivierung der Hilfestellungen. Andere Probanden würden die Hilfe bei wichtigen beruflichen Projekten aktivieren.

Für vier Probanden hätte die Verwendung eines solchen Produktes den Vorteil davon zu lernen bzw. ihr Wissen über Guidelines zu erweitern. Manche verglichen es mit einer Lernsoftware umgesetzt als Hilfestellung. Besonders betont wurde diese Aussage von den zwei Personen, die angaben, geringe Vorkenntnisse im Bezug auf Guidelines zu haben (siehe Abbildung 6.1). Für die Person mit der Angabe, hohe Vorkenntnisse zu besitzen, wäre es wünschenswert, wenn sich die Software automatisch updaten könnte und die Guidelines dadurch immer „up-to-date“ bleiben würden. Dadurch wäre man stets über neue oder geänderte Richtlinien im Bilde. Bei einer solchen Funktionalität würde das Produkt auch für Personen mit hohen Vorkenntnissen einen Lernnutzen bieten.

Für manche Teilnehmer, vor allem für jene mit guten oder hohen Vorkenntnissen, wäre der Einsatz eines solchen Produktes bei hohem Installationsaufwand bedenklich. Ein Nutzen durch das Tool wäre auf jeden Fall gegeben, jedoch würde eine bestehende Integration in einen GUI-Builder bevorzugt werden.

Kapitel 7

Diskussion

Im Folgenden soll die im Zuge dieser Arbeit entwickelte Lösung zur automatisierten Inspektion eines User-Interface diskutiert werden. Zu diesem Zweck werden die erreichten Ergebnisse, Vor- und Nachteile, als auch Erweiterungs- und Verbesserungsmöglichkeiten genannt. Zur Diskussion wird auf bereits existierende Lösungen, wie die in Abschnitt 4.1.2 vorgestellten Tools, Bezug genommen.

7.1 Ergebnisse

Durch die direkte Integration der Hilfestellung in eine Entwicklungsumgebung wird eine kontextsensitive Evaluierung ermöglicht. Der Vorteil dabei ist, dass der Benutzer direkt zum Zeitpunkt der Entstehung auf einen Verstoß gegen eine Guideline hingewiesen wird. Durch diese Methodik erhält der Benutzer die Gelegenheit, sich bei erstmaligem Auftreten des Verstoßes über die Guideline zu informieren. Dadurch können im weiteren Verlauf Wiederholungsfehler vermieden werden. Bei allen erwähnten Tools in Abschnitt 4.1.2 hingegen, muss die Überprüfung explizit von den Benutzern angestoßen werden. Die Darstellung der Verstöße erfolgt nicht kontextsensitiv, sondern in zusammengefasster Form nach durchgeführter Evaluierung. Im Bezug auf die Unterteilung zwischen automatisierten Ansätzen von Ivory und Hearst [12] (siehe Beginn von Kapitel 4), kann die entwickelte Lösung als Methode zur Analyse und Kritik von User-Interfaces gesehen werden. Verstöße gegen Guidelines werden nicht nur identifiziert, sondern auch begründet. Bei einem festgestellten Verstoß erhält der Benutzer neben Begründungen auch Vorschläge zur Verbesserung oder zu alternativen Lösungswegen. Bei der Umsetzung von alternativen oder besseren Lösungsmöglichkeiten, wird dem Anwender Unterstützung durch die automatische Modifizierung des Interface geboten. Gemäß [13] bietet eine Methode zur Kritik mit automatisierter Modifizierung das höchste Maß an Unterstützung. Großer Wert wurde auch auf individuelle Überlegungen eines Entwicklers zu seinem Design gelegt. So

kann dieser selbst entscheiden, ob er einer Richtlinie nachkommen möchte und ob diese im weiteren Verlauf noch zur Evaluierung herangezogen werden soll. Wie festgestellt werden konnte, ist die automatisierte Evaluierung eines User-Interface nur gegen messbare Guidelines möglich. Nicht-messbare, qualitative Guidelines werden im entwickelten Prototyp in Form von Checklisten dargestellt und ermöglichen einen Guideline-Review durch den Benutzer selbst. Dadurch werden eine vollständige Evaluierung und ein fehlerfreies Design, im Bezug auf die Einhaltung von vorhandenen Guidelines in der Wissensbasis, ermöglicht.

Für die Veranschaulichung der grundlegenden Idee, wurde eine *Adobe AIR*-Anwendung erstellt. Die Integration eines automatisierten Guideline-Reviews und einer Checkliste für die Evaluierung durch den Benutzer selbst, konnten dadurch realisiert werden. Um die Ideen der Lösungsansätze umsetzen und visualisieren zu können, wurden die Grundfunktionalitäten eines grafischen Editors zur Erstellung von GUIs nachgestellt.

Zusammengefasst kann festgestellt werden, dass die Möglichkeit, ein erstelltes Interface automatisch evaluieren lassen zu können, von den Testpersonen als sehr praktisch und zeitersparend empfunden wurde. Die selbstständige Evaluierung durch die Checklisten erfordert, nach Angaben der Probanden, erstmalig viel Zeit und Mühe, auch wenn der Aufwand nach mehrmaliger Anwendung geringer geschätzt wird. Daraus lässt sich schließen: Je mehr Guidelines von der Software automatisiert überprüft werden können, desto praktikabler wird die Anwendung für den Endbenutzer. Eingesetzt werden würde die Hilfestellung sowohl im privaten als auch im beruflichen Bereich, vorwiegend bei Unsicherheiten bezüglich des eigenen Designs oder bei wichtigen beruflichen Projekten.

7.2 Ausblick

Da es sich bei der umgesetzten Lösung nur um eine erste Version eines Prototypen handelt, gibt es diesbezüglich noch viele Verbesserungs- und Erweiterungsmöglichkeiten. Die Weiterentwicklung des Prototypen kann dazu verwendet werden, eine optimale Lösung für die Integration einer automatisierten Usability-Evaluierung in eine Entwicklungsumgebung zu finden.

Der Prototyp zielt jedoch nicht auf die Entwicklung eines marktreifen Produktes ab, da der grafische Editor einer Entwicklungsumgebung zum Zweck der Demonstration der Lösungsidee nachgebaut wurde und nur über die wichtigsten Funktionalitäten verfügt.

7.2.1 Erweiterte Evaluierung

Evaluiert wird das User-Interface gegen einen Satz ausgewählter Guidelines. Die Guidelines beziehen sich dabei ausschließlich auf die zur Verfügung stehenden GUI-Elemente. Im Vergleich mit anderen Tools ist diese

Lösung sehr fokussiert und ermöglicht dadurch eine präzise Evaluierung für einen bestimmten Bereich von Usability-Guidelines. Als Gegensatz dazu, evaluiert Sherlock [18] (siehe Abschnitt 4.1.2) unter anderem Layout-Eigenschaften (Größe und Ausrichtung der Elemente), Dichte und Balance der gesamten Oberfläche, die Konsistenz von Schriftarten, Schriftgrößen und Hintergrundfarben oder inkonsistente Ausdrucksweisen (unterschiedliche Groß- und Kleinschreibung oder Rechtschreibfehler). Dementsprechend könnte auch die im Zuge dieser Arbeit entwickelte Lösung, um die Evaluierung weiterer Bereiche von Guidelines erweitert werden. Möglich wäre die Integration von zusätzlichen Design-Guidelines, Design-Principles oder Konventionen. Auch die Messung der Ästhetik, wie durch das Tool AMA [45] oder der Einbau von modellbasierten Analysen (siehe Abschnitt 3.2.2) wäre denkbar. Dabei sollte die Präzision bei der Auswahl der Guidelines jedoch nicht verloren gehen, da am Ende nicht der Eindruck entstehen soll, dass alles und gleichzeitig nichts evaluiert wird.

Im Rahmen der prototypischen Umsetzung erfolgte die Evaluierung von vier GUI-Elementen. Eine vollständige Evaluierung würde die Auswahl, Klassifizierung und Umsetzung aller Guidelines für alle in der Entwicklungsumgebung vorhandenen GUI-Elemente voraussetzen.

7.2.2 Auswahl der Guidelines

Die Auswahl der Guidelines für den definierten Bereich, erfolgte durch die Verwendung von vier Quellen. Die Voraussetzung für die Aufnahme in die Wissensbasis war entweder das Vorhandensein einer Guideline in mindestens zwei Quellen oder die Belegung dieser durch die Angabe einer Quelle. Dabei wurde nicht jede mögliche Guideline analysiert, sondern es wurde eine repräsentative Auswahl getroffen. Diese Methodik lässt sich durch die Verwendung von weiteren Quellen und durch die genaue Analyse aller Guidelines mit Sicherheit noch verfeinern. Durch die dadurch steigende Anzahl der Guidelines je GUI-Element, wäre eine noch präzisere Evaluierung möglich.

7.2.3 Klassifizierung der Guidelines

Bei der vorgenommenen Klassifizierung wurde jede Guideline einem GUI-Element zugeordnet. Danach erfolgte eine Unterteilung in messbare und nicht-messbare Guidelines. Diese Klassifizierung hat zum Nachteil, dass eine Guideline öfter als einmal in der Wissensbasis vorkommt. Zum Beispiel betrifft die Guideline *Prefer to align groups of elements vertically instead of horizontally* sowohl Radiobuttons als auch Checkboxes. Ergoval [8], siehe Abschnitt 4.1.2, erstellt durch das Gruppieren von Objekten, welche von denselben Regeln betroffen sind, eine Typologie. Diese Methode verhindert das mehrfache Vorkommen von gleichen Guidelines in der Wissensbasis.

In Anlehnung daran, könnte eine Überarbeitung der bestehenden Klassi-

fizierung erfolgen. Je besser die Klassifizierung der Guidelines, desto geringer ist auch der Aufwand für die Wartung der Wissensbasis.

7.2.4 Verwaltung der Guidelines

Großes Potential steckt in der verbesserten Verwaltung der Guidelines. Im Moment werden die Daten der Guidelines - Name, Kurz- und Detailbeschreibung - direkt in den Programmcode eingebettet. Metrische Grenzwerte einer Guideline werden, als Konstanten definiert, ebenfalls im Programmcode gehalten. Ein Nachteil dieser Lösung ist, dass bei Änderungen der Daten oder steigendem Umfang der Guidelines, die Wartung dieser erheblich erschwert wird. Dazu wäre jedes Mal ein Anpassen des Programmcodes nötig.

Eine verbesserte Datenverwaltung, wie z.B. durch eine relationale Datenbank, könnte diesem Problem entgegenwirken. Eine Alternative wäre ein Ansatz im Sinne des semantischen Webs, bei dem die Daten der Guidelines strukturiert abgelegt werden können. Durch eine Veröffentlichung der Ontologie im Internet, könnten die Guidelines dezentral von mehreren Personen verwendet und erweitert werden.

Gemäß eigenen Angaben in [15], ist der große Vorteil des Tools Magenta, die Flexibilität bei der Verwaltung der Guidelines. Magenta bietet einen Editor zur Erfassung und Spezifizierung neuer oder existierender Guidelines. Eine dahingehende Erweiterung der bestehenden Lösung könnte angedacht werden.

7.2.5 Empirische Evaluierung

Um die durch die erste Untersuchung erkannten Probleme (Abschnitt 6.5) zu beheben, sollte in einem weiteren Schritt ein Redesign des Prototypen durchgeführt werden. Dieser Schritt beinhaltet auch die Umsetzung der gewünschten zusätzlichen Funktionalitäten der Testpersonen. Um auch diese redesignten oder neu implementierten Funktionen evaluieren zu können, müsste anschließend ein zweiter Test durchgeführt werden. Auch [38] und [26] beschreiben den Vorteil des iterativen Testens. Um ein optimales Ergebnis zu erzielen, empfiehlt Jakob Nielsen in [26] drei iterative Tests mit jeweils fünf Probanden und anschließendem Redesign.

Aufgrund des begrenzten zeitlichen Rahmens wurden die Ergebnisse der ersten Untersuchung in dieser Arbeit nur textuell festgehalten und nicht in einem Redesign umgesetzt.

7.3 Resumee

Zur automatisierten Durchführung eines Guideline-Reviews, existieren zum Zeitpunkt dieser Arbeit eine Reihe von Tools. Diese Tools assistieren dem Benutzer bei der Erstellung eines benutzerfreundlichen Interface, durch das

Finden und Erläutern von Usability-Verstößen. Ein Nachteil dieser Lösungen ist, dass die Evaluierung vom Entwickler selbst zu einem gewählten Zeitpunkt angestoßen werden muss, um ein Feedback über die Usability des Interface zu erhalten. Ziel dieser Arbeit war es, eine Lösung zu entwickeln, bei welcher der Entwickler unmittelbar beim Auftreten des Usability-Verstoßes einen Hinweis darauf erhält. Dadurch wird eine kontextsensitive Evaluierung, durchgehend während der gesamten Erstellungsphase des User-Interface, ermöglicht.

Um die gesteckten Ziele zu erreichen, wurde eine Lösungsidee entwickelt, bei der die Hilfestellung direkt in eine Entwicklungsumgebung zur Erstellung von grafischen Benutzeroberflächen integriert ist. Die Umsetzung der Lösungsidee als funktionsfähiger Prototyp erfüllt das Ziel der Visualisierung einer kontextsensitiven Evaluierung während der Erstellung eines User-Interface. Neben der Visualisierung der Lösungsidee, werden durch die Umsetzung technische Überlegungen und Vorgehensweisen, welche die Qualität einer derartigen Hilfestellung beeinflussen, sichtbar. Die durchgeführte empirische Evaluierung durch potentielle Anwender beweist die Alltagstauglichkeit und Nützlichkeit der grundlegenden Lösungsidee.

Abschließend sei gesagt, dass automatisierte Methoden zur Evaluierung der Usability eines Produktes dem Entwickler zwar helfen Arbeit, Zeit und Kosten zu sparen, jedoch Tests mit echten Benutzern nicht ersetzen. So kann die automatisierte Durchführung eines Guideline-Reviews zwar als Ergänzung, aber nicht als Ersatz für empirisch belegbare Ergebnisse gesehen werden.

Anhang A

Personas & Szenarios

Zur Spezifizierung der Zielgruppe und zur Vermittlung der Design-Konzepte, wurden vor der Umsetzung des Prototypen Personas und Szenarios entwickelt. Nachfolgend werden drei Personas mit dazugehörigem Szenario dargestellt. Zum besseren Verständnis enthalten die Szenarios Skizzen, welche in den Texten referenziert werden.

Michaela

Die Webdesignerin

Persönliches Profil:

Michaela arbeitet bereits seit fünf Jahren in einer Werbeagentur für Onlinemedien. Ihr Aufgabenbereich umfasst, die Designs für die Onlineauftritte diverser Kunden zu entwerfen und umzusetzen. Je nach Auftrag verwendet Michaela dafür verschiedene Entwicklungswerkzeuge und Programme. Zum Erstellen ihrer GUIs verwendet sie hauptsächlich die Designansicht von *Adobe Dreamweaver* und *Adobe Flash Builder*.

Michaela informiert sich täglich über News im Bereich Onlinemedien im Internet oder in Fachzeitschriften. Sie ist ständig „up-to-date" und weiß, welche Designs angesagt sind und somit auch was bei den Kunden gut ankommt. Der Chef der Werbeagentur ist sehr zufrieden mit Michaela und mit ihrer Arbeit.

Auch privat ist sie ein sehr gewissenhafter Mensch und liebt es wenn alles seine Ordnung hat. Urlaube plant Michaela immer schon ein Jahr vor Reiseantritt, damit auch sicher nichts schief geht. Sie will nichts dem Zufall überlassen und deckt sich mit so vielen Informationen ein, wie sie finden kann.

Persönliche Informationen:

Alter:
35 Jahre

Bildungsabschluss:
FH Salzburg
Master of Arts in Arts and Design

Beruf:
Webdesignerin in einer Werbeagentur

Hobbys:
Mode, Museum, Internet, Fachzeitschriften für Kunst

Mediennutzung:

Nutzt Internet viel für private Zwecke: Online-Shopping usw.

Interessiert sich für neue Technologien und Anwendungen

Abbildung A.1: Beschreibung der Persona Michaela.

Szenario:

Michaela soll für einen Kunden ein Design für einen Online-Fragebogen entwerfen. Sie verwendet dafür die Designansicht von *Adobe Flash Builder*. Michaela ist mit den GUI-Elementen der Designansicht vertraut und weiß auch wie diese grundsätzlich verwendet werden.

Beim Erstellen ihres Fragebogens verwendet Michaela Elemente wie Checkboxes, Radiobuttons, Textfelder usw. Danach zieht sie die benötigten Elemente auf die Bühne und erstellt ihren Fragebogen. Dabei verwendet sie die Elemente richtig und macht keine Fehler. Sie bekommt lediglich ein **paar Hinweise** über die sie dankbar ist und die sie auch liest [Bild 1].

Während Sie einen Radiobutton auf die Bühne zieht fällt ihr auf, dass ein Icon eingeblendet wird. Das Icon enthält eine Liste mit allgemeinen Ratschlägen zur richtigen Verwendung von Radiobuttons. Sie liest die Hinweise in der Liste. Nachdem Sie mit den Radiobuttons fertig ist, geht sie die Liste durch und markiert alle **Hinweise grün**, die sie bereits beachtet [Bild2, Nr.1]. Sie erkennt, dass sie anstatt numerischer Daten in Radiobuttons einen Slider verwenden sollte und behebt diesen Fehler. Danach markiert sie den Ratschlag in der Liste grün [Bild 2, Nr. 2].

Bild 1:

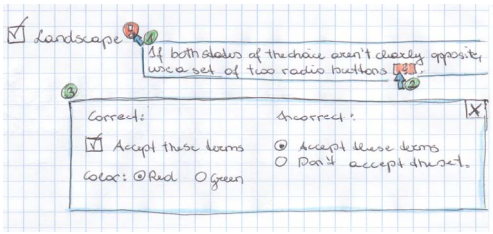


Bild 2:

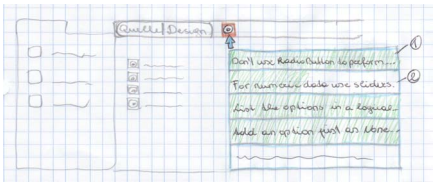


Abbildung A.2: Beschreibung eines Szenarios für die erstellte Persona Michaela.

Andreas

Der Programmierer

Persönliches Profil:

Andreas ist Webentwickler bei einer Agentur für Internet- und Intranet-Projekte und als Programmierer tätig. Vor drei Jahren hat er sein Studium der Informatik mit dem Schwerpunkt Web abgeschlossen. Die meisten Webshops programmiert er entweder mit *MS Visual Studio* oder mit *Adobe Flash Builder*. Andreas ist ein wirklich sehr guter Programmierer und legt nicht viel Wert darauf, dass seine Produkte optisch ansprechend sind. Er sieht seine Arbeit eher als Mittel zum Zweck um Geld zu verdienen.

Privat ist Andreas ein eher verschlossener Typ und freut sich wöchentlich auf das Fußballtraining mit seinen Kumpels. Wenn er mit seiner Arbeit fertig ist, möchte er so wenig wie möglich mit Medien zu tun haben. Auch mag er es nicht, wenn Unvorhergesehenes eintritt. Am liebsten ist es ihm, wenn alles beim Alten ist und bleibt.

Persönliche Informationen:

Alter:
32 Jahre

Bildungsabschluss:
Studium Informatik
Master of Science

Beruf:
Webentwickler

Hobbys:
Kino, Freunde treffen, Fußball

Mediennutzung:

Nutzt Internet für Recherchearbeiten

Abbildung A.3: Beschreibung der Persona Andreas.

Szenario:

Andreas ist gerade dabei ein Formular für die Registrierung in einem Onlineshop zu erstellen. Er hat nur wenig Zeit dafür, da die Seite in den nächsten Tagen online gehen soll. Er verwendet dafür *Visual Studio* und wechselt in die Designansicht.

Andreas liest die Tooltips zu den GUI-Elementen und verwendet dadurch die meisten Elemente richtig. Er bemerkt die Warnungen bei seinen Elementen auf der Bühne. Er möchte **momentan keine Hinweise** erhalten und blendet diese über die Checkbox aus [Bild 1]. Danach arbeitet weiter.

Am Ende schaltet er die Hinweise wieder ein und liest diese auch teilweise. Ein paar davon erscheinen ihm nützlich und werden von ihm abgearbeitet. Aufgrund des Platzmangels, entschließt er sich dafür eine **Drop-Down-Liste** anstatt seiner langen Radiobutton-Gruppe zu verwenden [Bild 2]. Die anderen Hinweise für die Elemente interessieren ihn nicht mehr.

Obwohl Andreas nicht viele der Hinweise beachtet, hat er aufgrund der Tooltips und der Beachtung von einigen Hinweisen ein gutes Formular für die Registrierung erstellt.

Bild 1:

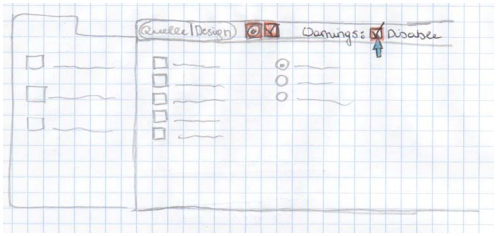


Bild 2:

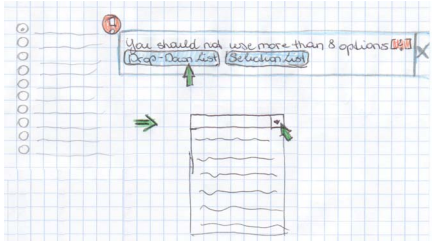


Abbildung A.4: Beschreibung eines Szenarios für die erstellte Persona Andreas.

Martin

Der Allrounder

Persönliches Profil:

Martin studiert bereits im sechsten Semester Medientechnik und Design. Er ist sehr vielseitig interessiert und weiß noch nicht so richtig in welchem Bereich er nach dem Studium arbeiten möchte.

In seiner Freizeit beschäftigt er sich mit neuen Technologien und fotografiert gerne. Durch sein Studium ist Martin jedoch sehr stark gefordert und plagt sich ständig mit Übungen und Abgabeterminen herum. Auch mit Entwicklungsumgebungen kennt Martin sich durch das Studium sehr gut aus. Bei ein paar Studienprojekten konnte er bereits Erfahrungen im Bereich GUI-Design und –Entwicklung sammeln.

Um ein bisschen Geld zu verdienen, arbeitet er neben dem Studium als Freelancer. Die Projekte an denen er arbeitet reichen von einfachen Softwarelösungen bis hin zu größeren Webauftritten. Martin arbeitet hier nach dem Motto „Zeit ist Geld".

Martin ist ein interessierter junger Mann, der seine wirkliche Leidenschaft noch finden muss.

Persönliche Informationen:

Alter:
24 Jahre

Beruf:

Student Medientechnik und Design

Hobbys:
Webdesign, Handys, Joggen

Mediennutzung:

Interessiert sich für neue Technologien und Anwendungen

Studium

Abbildung A.5: Beschreibung der Persona Martin.

Szenario:

Martin, der Allrounder, möchte in seiner Tätigkeit als Freelancer für seinen Freund, den Pizzameister im Ort, ein kleines Bestellformular erstellen, damit seine Kunden in Zukunft ihr Essen auch online bestellen können. Martin verwendet dazu *Adobe Flash Builder*. Zum Erstellen des Formulars verwendet er die Designansicht.

Er entdeckt die GUI-Elemente, die er für sein Bestellformular brauchen kann und kennt diese auch teilweise recht gut. Martin macht ein paar Fehler hinsichtlich der Einhaltung von Design-Guidelines. Er verwendet eine Checkbox zum Akzeptieren der Bedingungen. Dieses Element wird von Martin richtig eingesetzt. Die Hinweise, die er bekommt, sieht er sich an und stellt dadurch fest, dass er keinen Fehler gemacht hat. Er **entfernt die Hinweise** [Bild 1] und das Rufzeichen verschwindet.

Am Ende ist Martin sehr froh, die Hilfestellungen erhalten zu haben, da er andernfalls das Bestellformular nur mit erheblich mehr Zeitaufwand erstellen hätte können. Auch die Kunden des Pizzameisters sind zufrieden, da sie in Zukunft ihre Pizzas auch online bestellen können.

Bild 1:

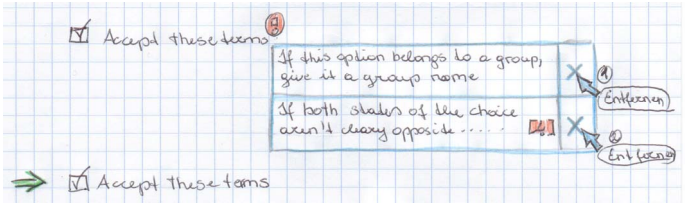


Abbildung A.6: Beschreibung eines Szenarios für die erstellte Persona Martin.

Anhang B

Klassifizierte Guidelines

Im Rahmen der Entwicklung des Prototypen, wurden die für die Evaluierung verwendeten Guidelines kategorisiert. Die Ergebnisse werden nachfolgend in tabellarischer Form dargestellt. Der Zusammenhang zwischen den Überlegungen bei der vorgenommenen Klassifizierung und den abgebildeten Tabellen ist dem Abschnitt 5.1.4 zu entnehmen.

Tabelle B.1: Kategorisierung der Guidelines für Radiobutton und Drop-Down-Liste.

	Guidelines	Measureable								Immeasureable		
		Available	Default	Alignment	Distance	Number	Size	Action	Sorted	Right Element	Sorted	Text/Meaning
Radiobutton	Every group of radiobuttons needs to be labeled.	x										
	You should have one radiobutton selected by default.		x									
	Prefer to align radiobuttons vertically instead of horizontally.			x								
	Provide adequate separation between choices.				x							
	Never use a single radiobutton.					x						
	Don´t use the selection of a radiobutton to perform a command.							x				
	It is recommendable not to use more than 8 options in a group.					x						
	Radiobuttons should be used to choose only one choice from a list.									x		
	The data should be small and fixed in number.									x		
	The options should be important enough to be a good use of the screen space.									x		
	The options of a group are listed in a logical order.										x	
	All labels should be written as a word or phrase, not as a sentence.											x
Drop-Down-Liste	Make the drop-down-list wide enough to display the longest entry.						x					
	Select the safest and most secure option by default.		x									
	Lists with a large number (more than 10) of items should be sorted alphabetically.					x			x			
	There should be no blank list items.	x										
	Don´t use the change of a drop-down-list to perform commands.							x				
	The user should be able to choose among more than two values.					x						
	Use a drop-down-list for a large or unlimited number of items.									x		
	Use a drop-down-list to choose only one choice from a list.									x		
	The list items should be sorted in a logical order.										x	
	Use a drop-down-list if it is not very important to draw attention to the options.									x		

Tabelle B.2: Kategorisierung der Guidelines für Button und Checkbox.

	Guidelines	Measureable								Immeasureable		
		Available	Default	Alignment	Distance	Number	Size	Action	Sorted	Right Element	Sorted	Text/Meaning
Button	For each window make the command buttons the same width.						x					
	Avoid narrow, short or tall command buttons.						x					
	Restrict the number of button within one window to 5 or 6.					x						
	Provide adequate and consistent spacing between adjacent buttons.				x							
	Provide a default action by making the most likely command the default behavior.		x									
	Use a command button to initiate an immediate action.									x		
	Provide consistency in button labeling across the screen.											x
	Use a meaningful description for the button label. Prefer specific labels over generic.											x
Checkbox	Every group of checkboxes needs to be labeled.	x										
	Prefer to align checkboxes vertically instead of horizontally.			x								
	Provide adequate separation between choices.				x							
	It is recommendable not use more than 10 options.					x						
	Use a group of checkboxes to select zero or more choices.									x		
	Use a stand-alone checkbox to select an individual choice.									x		
	All labels should be written as a word or phrase, not as a sentence.											x
	The options of a group are listed in a logical order.										x	

Anhang C

Unterlagen der empirischen Evaluierung

C.1 Aufgabenstellung

Im Rahmen der empirischen Evaluierung erstellten die Testpersonen anhand von Aufgabenstellungen ein Formular zur Buchbewertung.

Buchbewertung

1. Aufgabe
   Der Benutzer soll seine Anrede definieren können. Erstellen Sie eine Auswahl bestehend aus folgenden Daten:
   - Herr
   - Frau
2. Aufgabe
   Der Benutzer soll die Möglichkeit haben, seinen Namen bekannt zu geben.
3. Aufgabe
   Der Benutzer soll ein Buch der Autorin auswählen können. Je nach Schreibgeschwindigkeit der Autorin, können früher oder später noch weitere Bücher hinzugefügt werden. Eventuell wird die Auswahl auch auf mehrere Autoren erweitert. Folgende Bücher stehen im Moment zur Bewertung offen:
   - Als gäbe es kein gestern
   - Schwesterherz
   - Zwei Leben - eine Liebe
4. Aufgabe
   Der Benutzer soll das Buch mit folgenden Möglichkeiten bewerten können. Die Bewertungsoptionen sind gut durchdacht und werden sich

ziemlich sicher nicht mehr verändern.

- Großartig
- OK
- Grauenhaft

5. Aufgabe
   Die Art von Büchern, welche der Benutzer gerne liest, ist für den Betreiber der Seite interessant. Bieten Sie dem Benutzer folgende Auswahlmöglichkeiten:
   - Romane
   - Erzählungen
   - Krimis
   - Science Fiction
   - Fantasy
   - Kinderbücher
   - Fachbücher
   - Zeitschriften
6. Aufgabe
   Der Benutzer soll festlegen können, ob er über neue Bücher der ausgewählten Autorin einen Newsletter erhalten möchte.
7. Aufgabe
   Am Ende hat der Benutzer noch folgende Möglichkeiten seine Bewertung abzuschließen:
   - Senden
   - Zurücksetzen
   - Wiederherstellen

C.2 Ergebnistabellen

Nachfolgend werden die Ergebnisse der durchgeführten empirischen Evaluierung (Kapitel 6) tabellarisch dargestellt. Die Beschriftungen der Tabellen entsprechen den definierten Hauptkategorien. Die Spaltenbeschriftungen definieren sich wie folgt:
Die erste Spalte der Tabellen beinhaltet die Nummierungen, welche als Referenzen zu den wortwörtlichen Aussagen in den Transkripten dienen. Jeder Testperson wurde im Rahmen der Untersuchung ein Buchstabe zugeteilt. Die zweite Spalte beinhaltet diese Buchstaben und beschreibt damit wieviele der Probanden die generalisierte Aussage in Spalte drei tätigten. Diese Generalisierungen stellen die in Abschnitt 6.4 unter dem Punkt *Auswertung* beschriebenen Unterkategorien dar. In Spalte vier finden sich Ankerbeispiele zu den Generalisierungen.

Tabelle C.1: Kategorie 1: Allgemein.

Nr.	*Untersuchung*	*Generalisierung*	*Ankerbeispiel*
1.1	A,B,C,D,E	Texte der Guidelines oder Tooltips werden inhaltlich nicht 100 prozentig verstanden.	*Fixed in Number* verstehe ich jetzt nicht ganz.
1.2	A,B,C,D,E	Kann sich der Benutzer mit der Hilfe nicht identifizieren oder diese deckt sich nicht mit seinen Zielen, werden diese Tipps auch nicht umgesetzt.	Nein eigentlich mache ich das nicht, weil ich mir denke es sind eh nur neun und die sind übersichtlich genug.
1.3	A,D	Tooltips werden beachtet und als hilfreich empfunden.	Tooltip finde ich gut.
1.4	A,C,D,E	Vorteil: Ein Einsatz ist bei Bedarf oder Unsicherheit möglich.	Ich schaue es mir an, wenn ich mir nicht sicher wäre ob es jetzt gut ist, was ich gemacht habe.
1.5	B,C,D,E	Die Benutzer glauben viel durch die Verwendung der Hilfestellung zu lernen. Hilfreich sei diese für Einsteiger oder für Programmierer um up-to-date zu bleiben.	Ich glaube, dass man unheimlich viel lernt durch das Tool. Vor allem für Einsteiger ist das interessant.
1.6	A,D,E	Die Hilfestellung wird eher bei einer direkten Integration in den GUI-Builder verwendet. Ein hoher Installationsaufwand ist bedenklich.	Ja finde ich schon praktisch. Ob ich es jetzt installieren würde, weiß ich jetzt nicht.
1.7	E	Ein Grund für den Einsatz wäre die Bestätigung des eigenen Wissens durch das Tool.	Die Bestätigung durch das Programm finde ich sehr gut.

Tabelle C.2: Kategorie 2: Detailansicht.

Nr.	*Untersuchung*	*Generalisierung*	*Ankerbeispiel*
2.1	A,B,C,D,E	Das Fragezeichen wird bei Unklarheiten oder wichtigen Entscheidungen verwendet. Ansonsten: Klick zu viel.	Dann schauen wir was das genau meint. Also ich lese mir da jetzt das ganze durch was da steht vom Fragezeichen.
2.2	A,B,C,E	Die Detailtexte des Fragezeichens werden nur grob durchgelesen: - Inhalte werden als klar angesehen - Detailtext wird als zu lange empfunden.	Ok, das lese ich jetzt nicht alles durch weil ich mir denke das ist jetzt eh nicht so schwer zum anwenden, also schaue ich es nur grob durch.
2.3	A,B,C,D,E	Die Bilder werden zuerst beachtet. Benutzer vermuten, dass der Text dasselbe beschreibt wie die Bilder.	Die Bilder habe ich kurz gesehen. Zuerst habe ich auf die Bilder geschaut.
2.4	A,B,C,D	Die Inhalte wurden teilweise als hilfreich empfunden: - Details sollten Aussagen in Rufzeichen begründen - Details sollten mehr Informationen bieten als der kurze Satz davor.	Ja ok, das darf man eh anscheinend. Ok, da gibt es jetzt nicht viel mehr wichtige Informationen.
2.5	D	Der Detailtext wird als angemessen, logisch und nicht zu lange empfunden.	Das meiste ist sehr logisch und recht kurz gehalten.

Tabelle C.3: Kategorie 3: Checkliste.

Nr.	*Untersuchung*	*Generalisierung*	*Ankerbeispiel*
3.1	A,B,C,E	Der Sinn der Checkliste wird nicht verstanden. Eine Hilfestellung über das Ziel der Checkliste wäre erwünscht .	Da würde mir jetzt eine allgemeine Information helfen, für was das überhaupt ist. Und warum ich das jetzt anhaken soll oder nicht.
3.2	A,B,D,E	Die Guidelines der Checkliste werden während des Design-Prozesses nur kurz oder gar nicht beachtet, sondern erst am Ende der Erstellung abgehakt.	Gut dann klicke ich unten auf Checkbox bei der Checkliste weil es mich interessiert
3.3	A,D	Der Sinn und die Ziele der Checklisten werden richtig interpretiert.	Meine Überlegung ist jetzt, dass ich schauen soll, ob die Guideline zutrifft und dass ich es abhaken kann.
3.4	A,B,C,D,E	Es besteht der Wunsch nach der Bearbeitung von nicht relevanten oder nicht erwünschten Guidelines: - Guidelines, die nicht zutreffen, wegklicken oder verschieben - Nur relevante Guidelines sollen angezeigt werden.	Wenn ich es so durchlese, bemerke ich aber, es ist nicht immer jede Guideline für alles relevant.
3.5	A,C	Die Überprüfung erfolgt pro Element oder Gruppe separat.	Nein, also ich hätte es immer nur für eine Gruppe evaluiert.
3.6	A,B,C,E	Die Einschätzung des Zeitaufwands: - hoher Zeitaufwand - am Anfang hoch, dann niedriger.	Aber unten, das würde ich eher nicht immer alles durchlesen.
3.7	A,B,C,D	Eine Rückmeldung über erfolgreiches Abhaken ist erwünscht: - Prozent wie viel man schon richtig hat - Fortschrittsbalken.	Ganz automatisch würde ich mir wünschen, dass wenn ich alle angehakt habe, dass irgendeine Rückmeldung kommt, dass ich weiß das passt jetzt.
3.8	C	Die Guidelines werden während des Designprozesses beachtet. Dies erfolgt nach der Fertigstellung einer Gruppe oder eines Elements. Der Benutzer behält die Guidelines im Gedächtnis.	Also wenn das eine variable Anzahl hat, dann habe ich schon gelernt, dann muss man eine Drop-Down-Liste nehmen.

Tabelle C.4: Kategorie 4: Automatisierte Hilfe.

Nr.	*Untersuchung*	*Generalisierung*	*Ankerbeispiel*
4.1	A,B,C,D,E	Die Warnungen werden während der Erstellungsphase verwendet, sobald diese erscheinen. Sie werden nicht ausgeblendet. Das Rufzeichen stört nicht.	Jetzt ist mein Wissen aus, jetzt schaue ich nach.
4.2	A,B,C,D,E	Der Benutzer behält die Hilfestellungen teilweise im Gedächtnis und vermutet die Texte hinter den Warnungen.	Ich schätze mal er sagt da das gleiche, also mache ich diese gleich mal breiter.
4.3	A,B,D,E	Die Modifizierungen werden als praktisch empfunden. Mehr automatische Modifizierungen sind erwünscht, da sie einiges an Zeit sparen.	Es wäre auch noch praktisch, wenn es eine Möglichkeit gäbe, die optimale Breite einzugeben oder, dass man sieht wie lange der längste Eintrag ist.
4.4	A,B	Selbstständig ausgeblendete Warnungen sollen nicht mehr angezeigt werden. Auch nicht bei anderen Gruppen oder Elementen.	Vielleicht könnte es zwei Optionen geben, *Nie wieder anzeigen* und *Für diese Gruppe nicht mehr anzeigen.*
4.5	A,C,D,E	Es gibt Unklarheiten über die Gruppierung.	Dann mache ich aus den Buttons eine Gruppe.
4.6	B	Das Rufzeichen könnte den Arbeitsfluss unterbrechen. Eine Auflistung aller Warnungen wäre eine Möglichkeit.	Dass man alle Fehler die man macht in einer Liste hat und dass ich dann da drauf klicken kann, wäre praktisch.
4.7	B	Eine Belohnung für das Abarbeiten einer Fehlermeldung wäre erwünscht.	..eine Art Belohnung gekriegt hätte, wenn ich das Problem löse. Zum Beispiel ein grünes Häkchen.
4.8	B,C,D,E	Eine History wäre erwünscht, um vergangene oder zu schnell weggeklickte Warnungen abrufen zu können.	Jetzt hätte ich noch die Details zum zweiten Problem wissen wollen. Naja, jetzt ist es schon weg.
4.9	A,B,C,D,E	Der Button *Don´t show this Guideline again* ermöglicht das individuelle Entscheiden über die Anwendung der Guidelines.	Es ist gut, dass man eine Alternative aufgezeigt kriegt, aber dann trotzdem entscheiden kann, will ich das jetzt oder nicht.

Anhang D

Inhalt der CD-ROM

Format: CD-ROM, Single Layer, ISO9660-Format

D.1 Diplomarbeit

Pfad: diplomarbeit/

Reindl_Susanne_2011.pdf	Diplomarbeit (PDF-Datei)

D.2 Online-Quellen

Pfad: /

quellen/	Kopien der als Literatur verwendeten Internetseiten

D.3 Projektdateien

Pfad: projekt/

GuidelineReview.air . . .	Installationsdatei des Projekts
GuidelineReviewSource/	Projektordner mit Quellcode

Literaturverzeichnis

[1] Atterer, R.: *Model-based automatic usability validation: a tool concept for improving web-based UIs.* In: *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*, NordiCHI '08, S. 13–22, New York, USA, 2008. ACM.

[2] Badashian, A. S., M. Mahdavi, A. Pourshirmohammadi und M. M. Nejad: *Fundamental Usability Guidelines for User Interface Design.* In: *Proceedings of the 2008 International Conference on Computational Sciences and Its Applications*, S. 106–113, Washington, USA, 2008. IEEE Computer Society.

[3] Baker, S., F. Au, G. Dobbie und I. Warren: *Automated Usability Testing Using HUI Analyzer.* In: *Proceedings of the 19th Australian Conference on Software Engineering*, S. 579–588, Washington, USA, 2008. IEEE Computer Society.

[4] Bias, R. G.: *The pluralistic usability walkthrough: Coordinated empathies.* In: Nielsen, J. und R. L. Mack (Hrsg.): *Usability Inspection Methods*, Kap. 3, S. 63–76. Wiley, 1994.

[5] Card, S. K., A. Newell und T. P. Moran: *The Psychology of Human-Computer Interaction.* L. Erlbaum Associates Inc., Hillsdale, USA, 1983.

[6] Chi, E. H., P. Pirolli und J. Pitkow: *The scent of a site: a system for analyzing and predicting information scent, usage, and usability of a web site.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, S. 161–168, New York, USA, 2000. ACM.

[7] Farenc, C., V. Liberati und M. F. Barthet: *Automatic Ergonomic Evaluation: What are the Limits?* In: Vanderdonckt, J. (Hrsg.): *Proceedings of the 2nd International Conference on Computer-Aided Design of User Interfaces*, S. 159–170. Presses Universitaires de Namur, 1996.

[8] Farenc, C. und P. Palanque: *A generic framework based on ergonomics rules for computer aided design of user interface.* In: *Proceedings of the 3rd International Conference on Computer-Aided Design of User Interfaces*, S. 281–292, Norwell, USA, 1999. Kluwer Academic Publishers.

[9] Frommann, U.: *Die Methode „Lautes Denken“.* 2005. Online verfügbar unter http://www.e-teaching.org/didaktik/qualitaet/usability/Lautes Denken_e-teaching_org.pdf, aufgerufen am 15. März 2011. Kopie auf CD-ROM (lautes_denken.pdf).

[10] Galitz, W. O.: *The Essential Guide to User Interface Design - An Introduction to GUI Design Principles and Techniques.* Wily, New York, USA, 2. Aufl., 2002.

[11] Heindl, E.: *Logfiles richtig nutzen.* Galileo Press GmbH, Bonn, Germany, 2003.

[12] Ivory, M. Y. und M. Hearst: *The state of the art in automating usability evaluation of user interfaces.* ACM Computing Surveys, 33:470–516, 2001.

[13] Ivory, M. Y., R. R. Sinha und M. A. Hearst: *Empirically validated web page design metrics.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, S. 53–60, New York, USA, 2001. ACM.

[14] Johnson, J.: *GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos.* Morgan Kaufmann, Burlington, USA, 2. Aufl., 2008.

[15] Leporini, B., F. Paternò und A. Scorcia: *Easing web guidelines specification.* In: *Proceedings of the 7th International Conference on Web Engineering*, ICWE'07, S. 254–268, Berlin, Germany, 2007. Springer-Verlag.

[16] Löwgren, J. und T. Nordqvist: *Knowledge-based evaluation as design support for graphical user interfaces.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, S. 181–188, New York, USA, 1992. ACM.

[17] MacKenzie, S.: *Motor Behaviour Models for Human-Computer Interaction.* In: Carroll, J. M. (Hrsg.): *HCI Models, Theories, and Frameworks. Toward a Multidisciplinary Science*, Kap. 3, S. 27–54. Morgan Kaufmann, San Francisco, USA, 2003.

[18] Mahajan, R. und B. Shneiderman: *Visual and Textual Consistency Checking Tools for Graphical User Interfaces.* IEEE Trans. Softw. Eng., 23:722–735, Nov. 1997.

[19] Mariage, C., J. Vanderdonckt, J. V und C. Pribeanu: *State of the art of web usability guidelines.* In: Proctor, R. W. und K. P. Vu (Hrsg.): *The Handbook of Human Factors in Web Design*, Kap. 38, S. 688–700. CRC Press, 2005.

[20] Mayring, P.: *Qualitative Inhaltsanalyse - Grundlagen und Techniken.* UTB, Weinheim, Germany, 9. Aufl., 2007.

[21] Microsoft Corporation: *Windows User Experience Interaction Guidelines.* 2011. Online verfügbar unter http://msdn.microsoft.com/en-us/library/aa511258.aspx, aufgerufen am 29. Mai 2011. Kopie auf CD-ROM (windows_guidelines.pdf).

[22] Molich, R., A. D. Thomsen, B. Karyukina, L. Schmidt, M. Ede, W. van Oel und M. Arcuri: *Comparative evaluation of usability tests.* In: *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, S. 83–84, New York, USA, 1999. ACM.

[23] Ngo, D. C. L., L. S. Teo und J. G. Byrne: *Modelling interface aesthetics.* Information Sciences, 152:25–46, Juni 2003.

[24] Nielsen, J.: *Usability Engineering.* Morgan Kaufmann, San Diego, USA, 1993.

[25] Nielsen, J.: *Guidelines for multimedia on the web.* 1995. Online verfügbar unter http://www.useit.com/alertbox/9512.html, aufgerufen am 30. Mai 2011. Kopie auf CD-ROM (multimedia_guidelines.pdf).

[26] Nielsen, J.: *Why you only need to test with 5 users.* 2000. Online verfügbar unter http://www.useit.com/alertbox/20000319.html aufgerufen am 10. Mai 2011. Kopie auf CD-ROM (user_testing.pdf).

[27] Nielsen, J.: *Ten usability heuristics.* 2005. Online verfügbar unter http://www.useit.com/papers/heuristic/heuristic_list.html, aufgerufen am 30. Mai 2011. Kopie auf CD-ROM (heuristics.pdf).

[28] Nielsen, J. und H. Loranger: *Web Usability.* Addison-Wesley Verlag, München, Germany, 2006.

[29] Nielsen, J. und R. Molich: *Heuristic evaluation of user interfaces.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People*, CHI '90, S. 249–256, New York, USA, 1990. ACM.

[30] Niesen, J.: *Heuristic evaluation.* In: Nielsen, J. und R. L. Mack (Hrsg.): *Usability Inspection Methods*, Kap. 2, S. 25–62. Wiley, 1994.

[31] Ohnemus, K. R.: *Web style guides: who, what, where.* In: *Proceedings of the 15th Annual International Conference on Computer Documentation*, SIGDOC '97, S. 189–197, New York, USA, 1997. ACM.

[32] Rieman, J., S. Davies, D. C. Hair, M. Esemplare, P. Polson und C. Lewis: *An automated cognitive walkthrough.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology*, CHI '91, S. 427–428, New York, USA, 1991. ACM.

[33] Rosson, M. B. und J. M. Carroll: *Usability Engineering: Scenario-Based Development of Human-Computer Interaction (Interactive Technologies).* Morgan Kaufmann, San Francisco, USA, 2002.

[34] Saffer, D.: *Designing for Interaction: Creating Smart Applications and Clever Devices.* New Riders Press, Berkeley, USA, 2007.

[35] Sandrine, B.: *Automatic evaluation of user interface usability: Dream or reality.* In: *Proceedings of the Queensland Computer-Human Interaction Symposium*, S. 44–46. Bond University, 1995.

[36] Scriven, M.: *The Methodology of Evaluation.* In: Tyler, R., R. Gagné und M. Scriven (Hrsg.): *Perspectives of Curriculum Evaluation, AERA Monograph Series on Curriculum Evaluation*, S. 39–83. Rand McNally, Chicago, USA, 1967.

[37] Sharp, H., Y. Rogers und J. Preece: *Interaction Design: Beyond Human-Computer Interaction.* Wiley, Chichester, England, 2. Aufl., 2007.

[38] Steve, K.: *Don't make me think!: Web Usability: Das intuitive Web.* mitp-Verlag, Heidelberg, Germany, 2. Aufl., 2006.

[39] Stone, D., C. Jarrett, M. Woodroffe und S. Minocha: *User Interface Design and Evaluation (Interactive Technologies).* Morgan Kaufmann, San Francisco, USA, 2005.

[40] Stroessl, S.: *Methoden des Testings im Usability Engineering.* In: Beier, M. und V. Von Gizycki (Hrsg.): *Usability: Nutzerfreundliches Web-Design*, S. 75–96. Springer-Verlag, Berlin, 2002.

[41] Sun Microsystems, Inc.: *Sun Web Application Guidelines – Version 4.1.* 2007. Online verfügbar unter http://developers.sun.com/docs/web-app-guidelines/uispec4_1/index.html, aufgerufen am 11. Juni 2011. Kopie auf CD-ROM (sun_guidelines_v4.pdf).

[42] Vanderdonckt, J.: *Development milestones towards a tool for working with guidelines.* Interacting with Computers, 12(2):81–118, 1999.

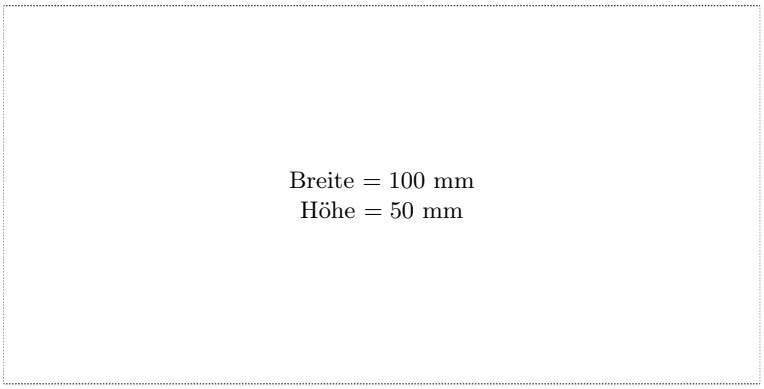
[43] Wharton, C., J. Rieman, C. Lewis und P. Polson: *The cognitive walkthrough method: a practitioner's guide.* In: Nielsen, J. und R. L. Mack (Hrsg.): *Usability Inspection Methods*, Kap. 5, S. 105–140. Wiley, 1994.

[44] World Wide Web Consortium (W3C): *Richtlinien für barrierefreie Webinhalte (WCAG) 2.0*. 2008. Online verfügbar unter http://www.w3.org/Translations/WCAG20-de/WCAG20-de-20091029/, aufgerufen am 11. Juni 2011. Kopie auf CD-ROM (wcag_2.0.pdf).

[45] Zain, J. M., M. Tey und G. Y. Soon: *Using Aesthetic Measurement Application (AMA) to Measure Aesthetics of Web Page Interfaces*. In: *Proceedings of the 2008 4th International Conference on Natural Computation - Volume 06*, S. 96–100, Washington, USA, 2008. IEEE Computer Society.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —