

DIPLOMARBEIT

BENUTZERFREUNDLICHE GRAFISCHE MODELLIERUNG VON LAYOUT UND DIALOGSTRUKTUR INTERAKTIVER SEITEN-BASIERTER ANWENDUNGEN

Franziska Wiese

Matrikelnummer: 3023671

Bearbeitungszeitraum: 01. Dezember 2009 – 03. August 2010



Hochschullehrer: Prof. Dr.-Ing. habil. Rainer Groh

Betreuer (TU Dresden): Dipl.-Medieninf. Christian Lambeck

Betreuer (SAP Research): Dipl.-Medieninf. Tobias Nestler

SELBSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die von mir am heutigen Tag dem Prüfungsamt der Fakultät Informatik eingereichte Diplomarbeit zum Thema „Benutzerfreundliche grafische Modellierung von Layout und Dialogstruktur interaktiver Seiten-basierter Anwendungen“ vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 03. August 2010

Franziska Wiese

ZUSAMMENFASSUNG

Endanwenderorientierte „Mashup-Tools“ [PBw] erlauben eine Komposition von Webinhalten durch den Endnutzer. In diesem Zusammenhang beschäftigt sich das EU-Projekt „ServFace“ mit einem präsentionsorientierten Ansatz für die Erstellung von servicebasierten Anwendungen. Die Referenzimplementierung „ServFace Builder“ ermöglicht die Komposition von Webservices zu einer seitenbasierten, interaktiven Anwendung auf UI-Basis.

Da die Ausrichtung endanwenderorientiert ist, muss der Umgang mit dem Werkzeug benutzerfreundlich gestaltet werden. Innerhalb dieser Arbeit werden Konzepte vorgestellt, wie die Funktionalität des Werkzeuges intuitiv präsentiert werden kann. Dafür werden alle möglichen Aspekte einer seitenbasierten Anwendung untersucht.

Konkret werden folgende Themen in dieser Arbeit behandelt:

- Darstellung der Seiten und visualisierten Services auf Basis von Plattformspezifikationen und Gestaltungsempfehlungen
- Repräsentation der Dialogstruktur einer Anwendung
- Angemessene Repräsentation der Funktionalität



AUFGABENSTELLUNG FÜR DIE DIPLOMARBEIT

Name, Vorname: Wiese, Franziska
Studiengang: Medieninformatik
Matr.-Nr.: 3023671

Thema: „Benutzerfreundliche grafische Modellierung von Layout und Dialogstruktur interaktiver Seiten-basierter Anwendungen“

ZIELSTELLUNG

Die Diplomarbeit findet im Rahmen des EU-Forschungsprojektes „ServFace“ statt, welches sich u.a. mit der Präsentations-orientierten Entwicklung von Service-basierten Anwendungen durch den Endanwender beschäftigt. Die Service-Komposition soll mittels vordefinierter Frontend-Komponenten auf Ebene der UI erfolgen und somit auch Nicht-IT-Experten die Erstellung interaktiver Anwendungen ermöglichen. Die einzelnen Frontend-Komponenten lassen sich aus mit UI-Annotationen angereicherten Web-Services generieren und bilden die Grundlage für die weitere Komposition. Zielstellung dieser Arbeit ist es, durch intuitive Visualisierungsformen den Endanwender in der Erstellung einer Seiten-basierter Anwendung zu unterstützen.

In einem ersten Schritt sind existierende Verfahren zur Visualisierung von Anwendungen im Bereich des End-User Development zu untersuchen. Dies beinhaltet die Fragestellungen: Wie kann die Funktionalität einer Anwendung visualisiert werden? Wie kann die Erstellung von Navigationspfaden visualisiert werden? Welche Formen der einfachen Layout-Anpassungen gibt es? Die Konzeption soll auf einem zugrundeliegenden Anwendungsmodell aufbauen und sich in ein existierendes präsentations-orientiertes Kompositionswerkzeug integrieren. Hierbei soll dem Nutzer der Aufbau einer Anwendung in einer für ihn verständlichen Art präsentiert werden. Zum Einen soll es ihm ermöglicht werden die Struktur und Hierarchie der Zielanwendung zu modellieren und die dabei involvierten Dialoge sowie verbindenden Navigationspfade zu spezifizieren. Zum Anderen sollen prinzipielle Layout-Informationen einzelner Seiten (Style Sheets, Logos, Farben und Schriftarten (CD), ...) und Spezifika definierter Plattformen (Auflösung, Bildschirmgröße,...) durch den Nutzer ausgewählt werden können und das Erscheinungsbild der Zielanwendung beeinflussen. Besonderer Schwerpunkt liegt auf der Einhaltung der Kriterien der Gebrauchstauglichkeit. Nach Auswahl eines geeigneten Anwendungsfalles soll eine prototypische Implementierung die Konzepte in das bestehende Kompositionswerkzeug integrieren und damit die Arbeit abschließen.

SCHWERPUNKTE

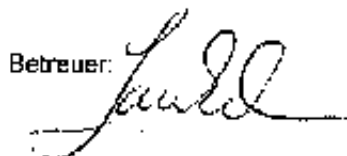
- Überblick über verwandte Arbeiten zur Visualisierung von Anwendungsstrukturen und der Definition von Layout und Dialogfluss durch den Endanwender
- Entwicklung eines Konzepts zur Modellierung von Dialogflüssen und Navigationsstrukturen innerhalb eines präsentationsorientierten Kompositionswerkzeuges
- Entwicklung eines Layout-Managers zur einfachen Spezifikation des prinzipiellen Aussehens der modellierten Anwendungen
- Prototypische Implementierung und Integration in das bestehende Kompositionswerkzeug

Betreuer:	Dipl.-Medieninf. Christian Lambeck Dipl.-Medieninf. Tobias Nestler (SAP AG)
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. habil. Rainer Groh
Beginn am:	01.12.2009
Einzureichen am:	31.05.2010
Institut:	Institut für Software und Multimediatechnik Lehrstuhl für Mediengestaltung
Zwischenpräsentation:	Voraussichtlich Anfang März
Literatur:	Thimbleby, H.: Press on, Massachusetts Institute of Technology, 2007 Groh, R.: Das Interaktionsbild, Otto-von-Guericke- Universität Magdeburg, Fakultät für Informatik, Nov. 2004

Unterschrift:

Student: 

Betreuer:



Hochschullehrer:



DANKSAGUNG

An dieser Stelle bedanke ich mich bei allen Personen, die mich während der Erarbeitung meiner Diplomarbeit unterstützt haben.

Insbesondere möchte ich mich bei meinen Betreuern Tobias Nestler und Christian Lambeck bedanken, die mich fachlich bei allen Schritten der Arbeit begleitet und viele gute Anregungen gegeben haben.

Desweiteren bedanke ich mich bei Prof. Dr.-Ing. habil. R. Groh für die Betreuung und Möglichkeit, diese Arbeit praxisnah bei SAP Research durchführen zu können.

INHALTSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS	XV
GLOSSAR	XVII
1. EINLEITUNG	1
1.1. Motivation	1
1.2. Ziele	2
1.3. Gliederung	2
2. RAHMENBEDINGUNGEN	5
2.1. ServFace	5
2.1.1. Konzept	5
2.1.2. Der ServFace Builder	6
2.1.3. Grundlegende Software-Architektur	9
2.2. Beschreibung der Zielgruppe	11
2.3. Eigenschaften der Zielplattformen	12
2.4. Evaluation des ServFace Builders	12
2.4.1. Ziele der Evaluation	12
2.4.2. Das Anwendungsszenario	13
2.4.3. Durchführung der Evaluation	14
2.4.4. Ergebnisse der Befragung	14
2.5. Zusammenfassung	16
3. ANALYSE UND VERWANDTE ARBEITEN	17
3.1. Navigation innerhalb seitenbasierter Anwendungen	17
3.1.1. Navigationsstrukturen	17
3.1.1.1. Hierarchische Struktur	18
3.1.1.2. Sequentielle Struktur	19
3.1.1.3. Netz Struktur	19
3.1.2. Navigationshilfen	19
3.1.3. Fazit	23

3.2.	Layoutempfehlungen für Webanwendungen	23
3.2.1.	Aufteilung der Seite	23
3.2.2.	Layouts für Formulare	24
3.2.3.	Fazit	30
3.3.	Layoutempfehlungen für mobile Anwendungen	30
3.4.	WYSIWYG-Werkzeuge	32
3.5.	Darstellung der Dialogstruktur	33
3.5.1.	Graphen	33
3.5.1.1.	Eigenschaften eines Graphen	34
3.5.1.2.	Darstellungsmöglichkeiten für einen Graphen	34
3.5.1.3.	Interaktion mit Graphen	36
3.5.1.4.	Fazit	37
3.5.2.	Sitemaps	37
3.5.3.	Visualisierung von Beziehungen	38
3.5.3.1.	Visuelle Programmierwerkzeuge	39
3.5.3.2.	Autorenwerkzeuge	41
3.6.	Zusammenfassung	43
4.	ANFORDERUNGSANALYSE	45
4.1.	Aufbau des Tools	46
4.1.1.	Hierarchische Aufgabenanalyse der endanwenderorientierten Service Komposition	46
4.1.2.	Präsentation der Funktionalität	48
4.2.	Frontend- und Seitendarstellung	48
4.2.1.	Plattformabhängige Seitendarstellung	48
4.2.2.	Präsentationsorientierte Darstellung	50
4.2.3.	Layoutmanager	51
4.3.	Dialogstruktur	51
4.3.1.	Erstellung und Darstellung	52
4.3.2.	Unterstützung verschiedener Navigationsstrukturen	53
4.3.3.	Unterstützung bei der Modellierung von Beziehungen	54
4.3.3.1.	Aspekte der Human-Computer-Interaction	55
4.3.3.2.	Einflussfaktor Datenfluss	56
4.4.	Zusammenfassung	58
5.	KONZEPTION	61
5.1.	Aufbau des Tools	61
5.2.	Frontend- und Seitendarstellung	63
5.2.1.	Plattformspezifische Darstellung	64
5.2.1.1.	Darstellung einer Webanwendung	64

5.2.1.2.	Darstellung einer mobilen Anwendung	67
5.2.1.3.	Evaluation der verschiedenen Frontend-Darstellungen . . .	68
5.2.2.	Präsentationsorientierte Darstellung	68
5.2.3.	Layoutmanager	69
5.3.	Dialogstruktur	69
5.3.1.	Darstellungsmöglichkeiten	70
5.3.1.1.	Bewertung anhand der Kriterien und einer Benutzerstudie .	74
5.3.1.2.	Spezifizierung der Graphen-Darstellung	76
5.3.2.	Unterstützung bei der Modellierung von Beziehungen	77
5.4.	Zusammenfassung	79
6.	IMPLEMENTIERUNG	81
6.1.	Anwendung	81
6.2.	Umsetzung	84
6.2.1.	Plattform- und navigationsspezifische Seitendarstellung	86
6.2.2.	Prozessansicht	87
6.3.	Zusammenfassung	88
7.	ZUSAMMENFASSUNG UND AUSBLICK	89
7.1.	Zusammenfassung	89
7.2.	Ausblick	90
A.	ANHANG	93
A.1.	Evaluation 02/2010	93
A.1.1.	Durchführung der Evaluation	93
A.1.2.	Resultate	94
A.1.3.	Fazit	97
A.2.	Evaluation 05/2010	97
A.2.1.	Durchführung der Evaluation	98
A.2.2.	Resultate	98
A.2.3.	Fazit	100
A.3.	Interaktionselemente	100
A.4.	Kompositionspattern	101
A.5.	Simple Mode	103
ABBILDUNGSVERZEICHNIS		XVII
TABELLENVERZEICHNIS		XIX
LITERATURVERZEICHNIS		XXI

ABKÜRZUNGSVERZEICHNIS

ER	Entity Relationship
EUD	End User Development
HCI	Human Computer Interaction
HTML	Hypertext Markup Language
ISA	Industry Standard Architecture
HTML	Hypertext Markup Language
IT	Information Technology
MS	Microsoft
MWI	Mobile Web Initiative
PERT	Program Evaluation and Review Technique
UI	User Interface
WSDL	Web Services Description Language
WYSIWYG	What you see is what you get
W3C	World Wide Web Consortium

GLOSSAR

Dialogstruktur	Beschreibung der Beziehungen zwischen den Objekten einer seitenbasierten Anwendung
Frontend	Visualisierung einer Service Operation mit allen Eingabe- und Ausgabeelementen aufgrund von grafischen Informationen und der WSDL-Beschreibung
idempotent	Umkehrbarkeit einer Aktion, eine nicht idempotente Aktion kann nicht rückgängig gemacht werden. Ursprünglich kommt der Begriff aus der Mathematik und beschreibt ein Element, das mit sich selbst verknüpft wieder sich selbst ergibt.
präsentationsorientiert	Modellierung ohne Abstraktion des Inhalts
Widget	Von der Umgebung unabhängiges Frontend, frei positionierbar per Drag & Drop
Transition	Übergang zwischen zwei Seiten

1 EINLEITUNG

Serviceorientierte Architekturen sind ein fester Bestandteil der heutigen IT-Welt. Mit Hilfe des Mashup-Konzepts können benutzerdefinierte Anwendungen aus Webinhalten modelliert werden. Das EU-Projekt ServFace befasst sich mit einem präsentationsorientierten Ansatz, wobei die Anwendung nach dem „What you see is what you get“-Prinzip (WYSIWYG) dargestellt wird [NFPS09]. SAP Research bietet für das Konzept eine Referenzimplementierung, den ServFace Builder, an [NFH⁺10]. Diese Arbeit befasst sich mit Konzepten für eine benutzerfreundliche Bedienung und für eine angemessene Präsentation für den Endnutzer im Hinblick auf die Erstellung einer seitenbasierten, interaktiven Kompositionsanwendung.

1.1 MOTIVATION

Das Konzept der Mashups bietet durch die endanwenderorientierte Entwicklung eine Vielzahl neuer Möglichkeiten für Nutzer ohne großes technisches Vorwissen, Anwendungen selbst zu modellieren und damit wiederverwenden zu können. Das EU-Projekt ServFace stellt ein präsentationsorientiertes Konzept bereit, das die Erstellung von Webanwendungen durch die Komposition von Webservices ermöglicht.

Mittels der Referenzimplementierung ServFace Builder kann der Nutzer die Webservices in Seiten integrieren, so dass eine mehrseitige Anwendung entsteht. Dem präsentationsorientierten Ansatz gemäß besitzen die Webservices ein visuelles Frontend, welches aufgrund der WSDL-Beschreibung und zusätzlichen grafischen Informationen generiert wird [NDP09].

Durch eine Evaluation, die zum Ziel hatte, die Benutzerfreundlichkeit des ServFace Builders zu testen, wurde deutlich, dass dem Nutzer noch das allgemeine Verständnis für die Funktionalität des Tools fehlt. Trotz einer nach dem „What you see is what you get“-Prinzip ausgerichteten Darstellung konnten die Probanden sich weder vorstellen, wie die Endanwendung aussieht, noch wie mit dieser zu interagieren ist. Dies lässt den Schluss zu, dass das Erscheinungsbild der Modellierungsansicht im ServFace Builder an bekannte Endanwendungen angepasst werden muss.

Neben der Integration von Webservices soll das Tool ebenfalls die Modellierung von Beziehungen zwischen Frontends und Seiten unterstützen. Dazu ist eine nähere Untersuchung bzgl. der Visualisierung von Beziehungen erforderlich, damit die Erstellung und Übersicht möglichst intuitiv gestaltet werden kann.

Doch es gilt nicht nur, bestehende Probleme zu beheben, sondern auch Erweiterungen bzw. Optimierungen vorzunehmen, um die Funktionalität des Tools angemessen zu präsentieren und die Modellierung so benutzerfreundlich wie möglich zu gestalten. Dazu soll wenn möglich eine Unterstützung durch das Tool bei der Modellierung geboten werden. Alle notwendigen Anforderungen, die sich durch das seitenbasierte Prinzip ergeben, wie beispielsweise verschiedene Navigationsstrukturen, sollen abgedeckt werden.

1.2 ZIELE

Die Ziele dieser Arbeit betreffen verschiedene Aspekte einer seitenbasierten Anwendung, die aus einer Komposition von Webservices besteht. Es soll eine angemessene Visualisierung des Seitenkonzepts sowie eine benutzerfreundliche Modellierung erreicht werden. Es folgt eine Aufzählung der Ziele:

- Visualisierung der Seiten
- Visualisierung der Beziehungen zwischen Seiten
- Unterstützung bei der Modellierung von Beziehungen zwischen Seiten

Die Darstellung der Seiten soll sich nach allgemeinen Layoutempfehlungen der jeweiligen Zielplattform richten. Zudem ist eine angemessene Integration der Frontends erforderlich sowie die Option einer Anpassung durch den Benutzer. Für eine geeignete Repräsentation der Beziehungen sollen mögliche Darstellungsalternativen definiert und diskutiert werden. Nach Untersuchung der zu bewältigenden Aufgaben kann ein angemessener Aufbau des Tools entwickelt werden. Die Konzepte sollen anschließend prototypisch in der Referenzimplementierung „ServFace Builder“ umgesetzt werden.

1.3 GLIEDERUNG

Zunächst werden im Kapitel 2 die Rahmenbedingungen für diese Arbeit erläutert. Daraufhin erfolgt die Präsentation der Grundlagen, auf denen die Konzepte basieren (Kapitel 3). Die Anforderungsanalyse (Kapitel 4) spezifiziert die Anforderungen bzgl. der genannten Ziele (siehe Abschnitt 1.2). Dabei wird zum Kompositionswerkzeug „ServFace Builder“ Bezug genommen. Die Anforderungen betreffen sowohl die Optimierung als auch eine Erweiterung

der Funktionalität. Im Kapitel 5 erfolgt die Entwicklung, Diskussion und Evaluierung der Konzepte. Die Konzepte wurden umgesetzt und werden innerhalb des Kapitels 6 anhand der Softwarearchitektur sowie von Screenshots veranschaulicht. Im letzten Kapitel 7 erfolgt eine kurze Zusammenfassung und ein Ausblick für zukünftige Arbeiten.

2 RAHMENBEDINGUNGEN

Innerhalb dieses Kapitels werden die Rahmenbedingungen der Arbeit erläutert. Da das Thema innerhalb des EU-Forschungsprojekts „ServFace“ stattfindet, werden im Abschnitt 2.1.1 die zugrundeliegenden Konzepte und die Methodologie vorgestellt. Da die Anforderungsanalyse direkt an der Referenzimplementierung des Projektes, dem „ServFace Builder“, ausgerichtet ist, werden dessen Funktionen und Ansichten sowie die grundlegende Software-Architektur detailliert beschrieben (Abschnitt 2.1.2). Das Projekt ServFace strebt eine endanwenderorientierte Entwicklung an. Die konkrete Zielgruppe des Projektes wird in dem Abschnitt 2.2 umrissen. Durch den modellgetriebenen Ansatz des ServFace Builders können verschiedene Zielplattformen unterstützt werden. Die Eigenschaften der für diese Arbeit relevanten Zielplattformen werden im Abschnitt 2.3 beschrieben. Durch eine Evaluation (Abschnitt 2.4), die im August 2009 bzgl. des ServFace Builders erfolgte, stellten sich einige Probleme heraus, die unter anderem diese Arbeit motivieren. Im letzten Abschnitt werden die Konsequenzen der Rahmenbedingungen im Hinblick auf den Inhalt dieser Arbeit erläutert.

2.1 SERVFACE

Diese Arbeit findet im Rahmen des Projektes ServFace statt. Der ServFace Builder ist eine Implementierungsvariante des Projektes. In den folgenden Abschnitten werden das Projekt und der ServFace Builder detailliert erläutert.

2.1.1 KONZEPT

Das Ziel des Projektes besteht in der Erstellung einer servicebasierten Anwendung durch einen Benutzer mit eingeschränkten Kenntnissen im Bereich der IT. Durch die Komposition von verschiedenen Webservices soll eine neue Anwendung entstehen [NFPS09]. Da die Modellierung auf UI-Basis erfolgt, benötigt der Benutzer kein technisches Hintergrundwissen über Webservices oder Service Kompositionen [NNA09], [NNA10]. Die Erstellung einer Anwendung folgt einer dreistufigen Methodologie (Abbildung 2.1).

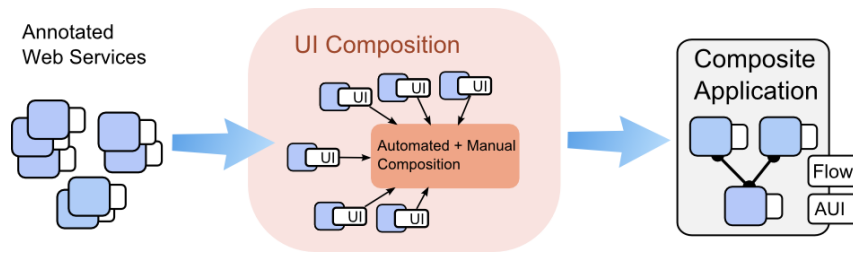


Abbildung 2.1.: Abbildung der dreistufigen ServFace Methodologie ([Jug09])

1. Im ersten Schritt werden den Service Operationen für die Beschreibung der Elemente und die Darstellung der Service-Struktur (mit Hilfe eines speziellen Werkzeugs) grafische Informationen hinzugefügt. Die sogenannten Annotationen enthalten neben Informationen über das UI ebenfalls funktionale Informationen, wie z.B. eine automatische Vervollständigung für Formularfelder [JPS⁺09].
2. Mit Hilfe der Annotationen kann ein Frontend – die Visualisierung einer Service Operation – generiert werden [NDP09]. Auf UI-Basis kann der Benutzer nun beliebig Frontends kombinieren, um eine interaktive Anwendung zu modellieren. Die Frontends können konfiguriert und in Verbindung gesetzt werden. Gemäß des präsentationsorientierten Ansatzes interagiert der Benutzer direkt mit einzelnen Elementen des Frontends. Der Modellierungszustand inklusive aller konfigurierten Frontends und Verbindungen wird auf eine interne Instanz des gegebenen Anwendungsmodells abgebildet und synchron gehalten [FNJ⁺09b].
3. Der letzte Schritt beinhaltet die Generierung einer lauffähigen Endanwendung aufgrund der im ServFace Builder erstellten Modellierungsinstanz. Durch den modellgetriebenen Ansatz kann eine Endanwendung für verschiedene Plattformen generiert werden [FNJ⁺09a], [DFN⁺10].

2.1.2 DER SERVFACE BUILDER

Der ServFace Builder dient zur Unterstützung bei der Modellierung von servicebasierten Anwendungen nach dem Konzept des Projekts ServFace. Entsprechend der spezifizierten Zielgruppe (Abschnitt 2.2) wird ein präsentationsorientierter Ansatz verfolgt [NNA09]; nach dem WYSIWYG-Prinzip soll die Darstellung der Entwurfsansicht der lauffähigen Endanwendung möglichst nahe kommen. Die Darstellung und Interaktionsprinzipien sind an der Benutzeroberfläche von Microsoft PowerPoint ¹ (Präsentationsfolien, Übersicht über alle Folien) orientiert; durch die Verwendung der MS PowerPoint-Metapher sollen die Gestaltungsprinzipien und Werkzeuge dem Benutzer schnell zugänglich werden. Eine Anwendung

¹ <http://office.microsoft.com/en-us/powerpoint/>

besteht aus einer oder mehreren Seiten, innerhalb deren beliebig Frontends integriert werden können. Die Seiten können miteinander verbunden werden; dies beinhaltet die automatische Generierung eines Navigationselementes auf der Startseite, zur Laufzeit kann ein Navigationselement einen Seitenübergang auslösen. Die Frontends werden als Widgets visualisiert und können innerhalb der Seite frei positioniert werden. Auch diese kann der Benutzer durch die Modellierung eines Datenflusses in Beziehung setzen. Die automatische Übertragung der Daten ist jedoch erst in der lauffähigen Version sichtbar.

Im Folgenden werden der aktuelle Zustand sowie die Ansichten des ServFace Builders vorgestellt:

Prozessansicht

Die Prozessansicht ist der initiale Zustand und bietet eine Übersicht über die aktuelle Dialogstruktur der Anwendung (Abbildung 2.2):

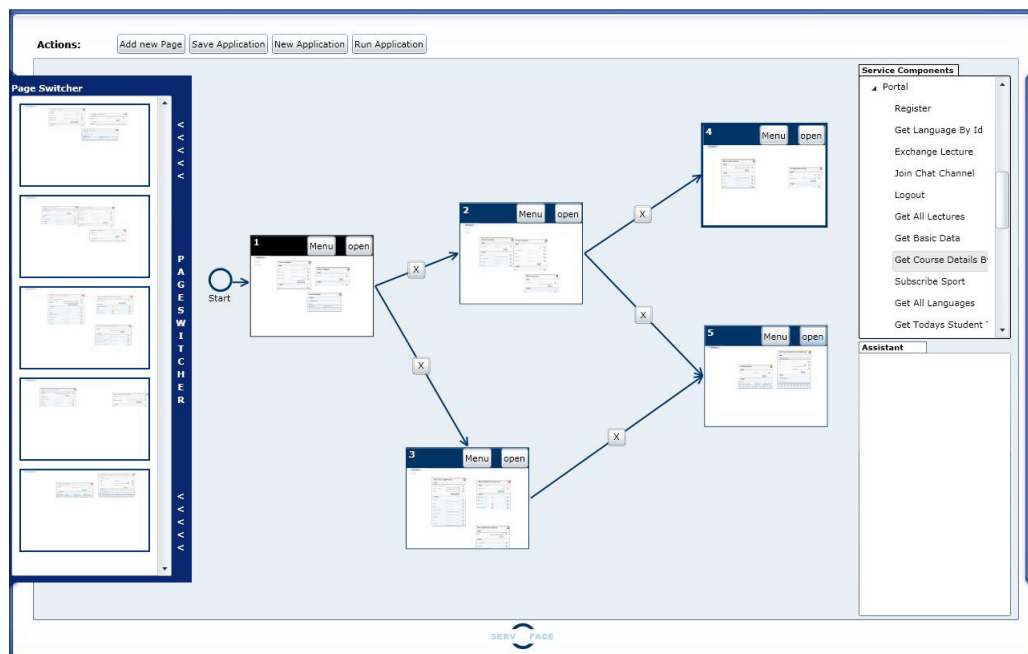


Abbildung 2.2.: Prozessansicht: Dialogstruktur der Anwendung (Screenshot)

Zu sehen sind alle vorhandenen Seiten in Thumbnailgröße, die Dialogflüsse werden als gerichtete Pfeile visualisiert. Um einen neuen Seitenübergang zu erstellen, muss der Benutzer das Seitenmenu aktivieren und die entsprechende Aktion ausführen. Am linken Seitenrand ist der „Pageswitcher“ zu sehen, der eine lineare Abfolge aller vorhandenen Seiten darbietet. Durch „Mouseover“ kann der Pageswitcher temporär angezeigt werden. Aus dieser Ansicht heraus kann der Benutzer in die Seitenansicht wechseln, indem er eine beliebige Seite durch

Klick auf „Open“ oder durch Selektierung innerhalb des Pageswitchers die entsprechende Seite vergrößert.

Seitenansicht

Innerhalb der Seitenansicht kann eine Seite konfiguriert werden (Abbildung 2.3):

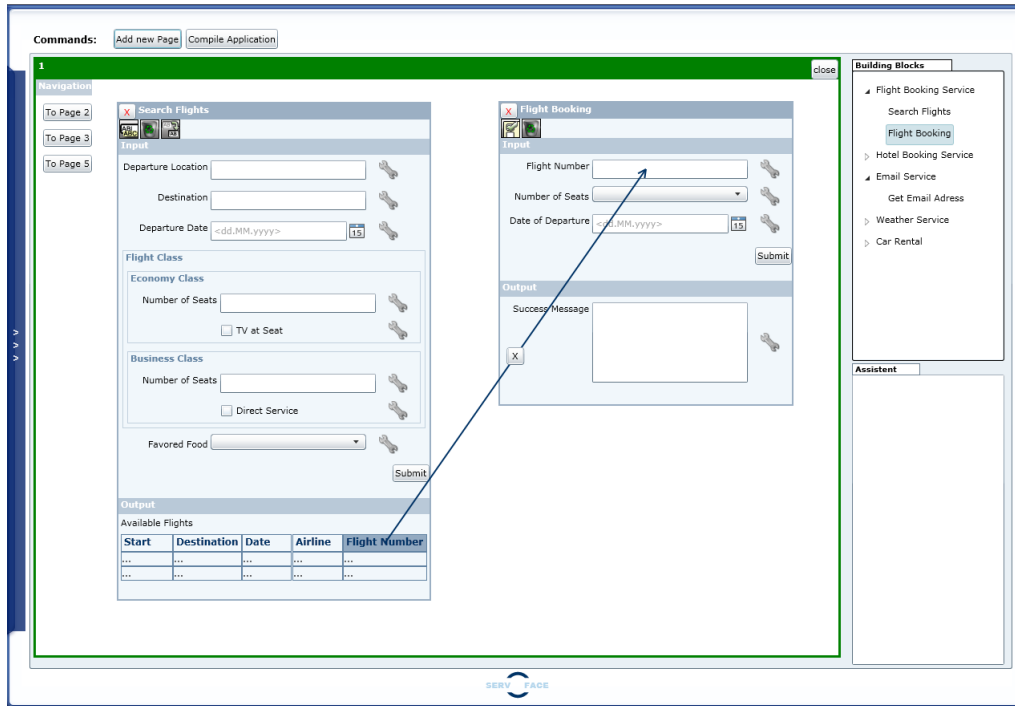


Abbildung 2.3.: Seitenansicht: geöffnete Seite, Möglichkeit zur Konfiguration (Screenshot)

Im rechten Bereich befindet sich der „Service Browser“, in dem alle verfügbaren Services mit ihren Service Operationen aufgelistet sind. Per Drag & Drop kann der Benutzer eine Service Operation auf die geöffnete Seite ziehen. Anschließend wird diese als Frontend visualisiert und kann nun vom Benutzer innerhalb der Seite frei positioniert werden. Jedes Frontend hat einen Ausgabe-Bereich und kann einen Eingabe-Bereich besitzen. Innerhalb eines Eingabe-Bereiches gibt es einen „Submit“-Button, mit Hilfe dessen die Eingaben bestätigt und der Service aufgerufen wird. Jedes Element des Frontends kann durch das nebenstehende Werkzeug-Icon konfiguriert werden. Beispielsweise besteht die Möglichkeit, das Label umzubenennen oder ein gesamtes Element zu verstecken. Zwischen den Frontends können Datenflüsse erzeugt werden, indem der Benutzer die entsprechenden UI-Elemente innerhalb der Frontends selektiert; anschließend wird der Datenfluss durch einen Pfeil repräsentiert, der die Richtung der Übertragung anzeigt. Zur Laufzeit werden die Daten nach Interaktion des Benutzers gesendet und in das Zielfeld automatisch eingetragen. Datenflüsse können zwischen Frontends innerhalb einer Seite („Intrapage“) oder zwischen zwei Seiten („Inter-

page“) erstellt werden. Um einen Interpage-Datenfluss zu modellieren, muss der Nutzer nacheinander die betreffenden Seiten öffnen, um die gewünschten UI-Elemente zu markieren. Im linken Bereich der Seite werden die Navigationselemente angezeigt, die zur Laufzeit einen Seitenübergang auslösen können. Die Navigationselemente werden automatisch generiert, sobald der Benutzer innerhalb der Prozessansicht einen Dialogfluss modelliert.

2.1.3 GRUNDLEGENDE SOFTWARE-ARCHITEKTUR

Das Kompositionswerkzeug wurde in Silverlight 3.0 umgesetzt. Die Implementierung folgt dem Pattern Model-View-Controller, in dem Präsentation und Datenmodell voneinander getrennt werden. Die Controller-Schicht ist für die Integration der Service Frontends und die Umsetzung der Änderungen an der erstellten Anwendung verantwortlich. Zudem stellt sie der View-Schicht wichtige Informationen für die Darstellung bereit.

Bei der Erstellung der Anwendung durch den Benutzer wird im Hintergrund ein Modell generiert, welches die Anwendung beschreibt. Das Meta-Modell dieses Modells heißt „Composite Application Model“ (CAM) und enthält alle Aspekte der interaktiven Applikation wie z.B. Aussehen oder Strukturierung und Verschachtelung der Elemente. Alle CAM-Objekte sind im ServFace Builder abgebildet und werden aktuell gehalten.

Die Abbildung 2.4 illustriert ein Klassendiagramm des ServFace Builders.

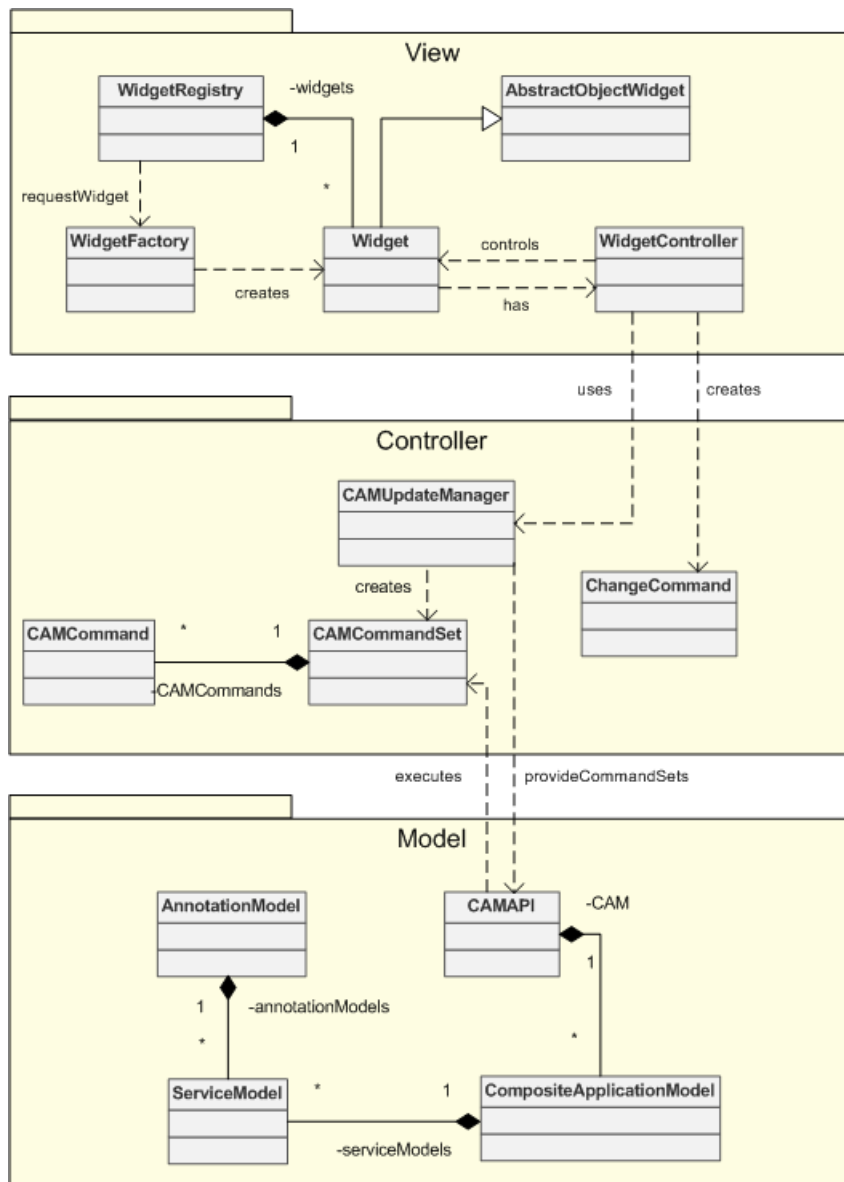


Abbildung 2.4.: Vereinfachtes Klassendiagramm

Alle Visualisierungen („Widget“) von Modellelementen im UI sind von der Klasse „AbstractObjectWidget“ abgeleitet und können über die zentrale „WidgetRegistry“ abgefragt werden. Erzeugt werden die „Widgets“ nach dem Factory-Pattern über die „WidgetFactory“. Eine von „AbstractObjectWidget“ abgeleitete Klasse muss einen „Container“ besitzen, der ein beliebiges Silverlight-spezifisches UI-Element wie z.B. ein „StackPanel“ oder eine „Canvas“ repräsentiert. In dem „Container“ können verschiedene UI-Elemente integriert werden wie z.B. ein Label für den Namen oder die Repräsentation eines interaktiven Elementes in Form von Eingabefeldern oder Tabellen.

Ein „Widget“ enthält Informationen über das Aussehen, der dazugehörige „WidgetController“ legt das Verhalten fest. Dieser entscheidet wie auf Änderungen im Modell zu reagieren ist, als auch welche Auswirkungen die Interaktionen des Nutzers mit dem „Widget“ haben. Falls die Interaktion Änderungen für das darunterliegende Meta-Modell zur Folge hat, sendet er „ChangeCommands“ an den „CAMUpdateManager“. Die „ChangeCommands“ sind die Schnittstellen zwischen den Schichten Controller und View. Somit kann die View ausgetauscht werden während die anderen Teile weiterhin bestehen.

Der „CAMUpdateManager“ entscheidet, welche Änderungen am Modell durchzuführen sind und kommuniziert mit der „CAMAPI“, die das Modell verwaltet und dementsprechend für die Ausführung der Änderungsanweisungen auf dem Modell zuständig ist.

Jedes einzelne „CAMObject“, welches eine UI-Repräsentation im ServFace Builder besitzt, implementiert eine „Change-Notification“. Diese lösen bei einer Änderung am Modell Events aus, für die die Controller der „Widgets“ Event-Listener registrieren können. Dadurch werden sie über Änderungen im Modell informiert und können somit das „Widget“, für das sie zuständig sind, synchron halten.

2.2 BESCHREIBUNG DER ZIELGRUPPE

Das Projekt ist endanwenderorientiert, d.h. eine Applikation soll anstatt von einem Softwareentwickler durch den späteren Anwender modelliert werden. In diesem Fall spricht man von „End User Development“ (EUD). Ein typischer Endnutzer aus der Zielgruppe kann als „Knowledge Worker“ oder „Information Worker“ bezeichnet werden, sein Arbeitsplatz ist meist im Büro.

Die sogenannten Information Worker lassen sich bezüglich ihrer Fähigkeiten im Umgang mit technischen Hilfsmitteln und Bearbeitung von Daten in unterschiedliche Kategorien einteilen. Die Mehrheit der Information Worker besteht jedoch aus Nutzern ohne großes technisches Vorwissen [MS]. Meist ist im Organisationsmodell einer Firma keine Arbeitsteilung vorgesehen, der Information Worker muss sich seine Informationen zur Bearbeitung selbst beschaffen. Dadurch bewegt sich der Information Worker meist zwischen einem Spezialisten und einem Generalisten: neben seinem Fachwissen ist er über die wichtigsten Entwicklungen in seiner Branche informiert und kennt unternehmerische Zusammenhänge, menschliche Eigenheiten sowie technologische Auswirkungen [Fis05]. Er verwendet Software überwiegend im beruflichen Umfeld. Durch den täglichen Gebrauch innerhalb des Arbeitsumfeldes ist er geübt im Umgang mit Werkzeugen aus dem Microsoft Office Paket ² wie z.B. MS Excel oder MS Access. Zusätzliches Wissen über die technischen Details oder die Funktionsweise

² <http://office.microsoft.com/de-ch/>

weiterführender IT-basierter Konzepte besitzt er jedoch nicht, auch verfügt er über keine Programmiererfahrung. Nichtsdestotrotz zeigt er Interesse an neuen IT-Trends.

2.3 EIGENSCHAFTEN DER ZIELPLATTFORMEN

Der ServFace Builder ist dafür ausgelegt, Anwendungen für verschiedene Zielplattformen zu modellieren. Innerhalb dieser Arbeit sollen Konzepte für zwei Zielplattformen entwickelt werden: Webanwendung und mobile Anwendung. In folgender Tabelle 2.1 werden die Charakteristika der beiden Zielplattformen definiert:

	Webanwendung	Mobile Anwendung
Gerät	Desktop/Laptop	Mobiles Gerät
Bildschirmauflösung	> 1024x768 Pixel	480x320 Pixel
Kommunikation	unterschiedliche	UMTS/HSDPA
Toolkit	Silverlight	iPhone SDK, Cocoa, Android SDK
Interaktion	Maus, Tastatur	Multitouch
Mehrseitige Anwendung	ja	ja
Dateneingabe	Tastatur	Software-Tastatur

Tabelle 2.1.: Plattformspezifika

2.4 EVALUATION DES SERVFACE BUILDERS

Im August 2009 wurde eine Benutzerstudie durchgeführt, um die Benutzerfreundlichkeit des ServFace Builders zu evaluieren [NNA09]. Einige Ergebnisse offenbarten, dass manche Konzepte der Verbesserung bedürfen. Somit dient diese Evaluation mit als Grundlage der Motivation dieser Arbeit. In den folgenden Abschnitten wird die Benutzerstudie beschrieben.

2.4.1 ZIELE DER EVALUATION

Ziel der Evaluation war die Überprüfung des aktuellen ServFace Builder-Prototypen hinsichtlich seiner Bedienbarkeit, Erlernbarkeit und Nachvollziehbarkeit. Insbesondere sollte analysiert werden, inwieweit die angewandten Interaktions- und Gestaltungskonzepte ver-

ständig sind. Im Zusammenhang mit dieser Arbeit waren vor allem folgende Aspekte interessant:

- Darstellung der Frontends in Hinblick auf die resultierende Applikation
- Konzept der Seitenübersicht und Verknüpfung verschiedener Seiten
- Navigation zwischen verschiedenen Ansichten und Seiten
- Notwendige Einarbeitungszeit in das Tool und die Lernkurve während der Verwendung
- Verständlichkeit des Startprozesses
- Unterstützungsgrad des Tools

2.4.2 DAS ANWENDUNGSSZENARIO

Im Lauf der Evaluation sollten die Teilnehmer eine kleine Applikation erstellen. Dazu wurde ihnen folgendes Szenario vorgegeben:

„Stellen Sie sich vor, Sie arbeiten als Berater in einem Software-Unternehmen. Meist arbeiten Sie vor Ort bei den Kunden, wodurch Sie viel und oft unterwegs sind. Da es sich um ein kleines Unternehmen handelt, sind Sie selbst für die Planung und Buchung Ihrer Reisen verantwortlich. So verbringen Sie jede Woche viel Zeit damit, die optimalen Flüge und passende Hotels zu suchen, indem sie mehrere Anbieter im Internet aufrufen, Ihre Daten in den unterschiedlichen Web-Formularen eingeben und die Angebote vergleichen. Sie ärgern sich jedes Mal darüber, dass Sie Ihre Reisedaten doppelt und dreifach eingeben müssen. Es wäre doch viel einfacher, die Daten einmal zentral anzugeben und alle Ergebnisse zentral dargestellt zu bekommen. Außerdem dauert das manchmal so lange, dass Sie vergessen, auch noch abzufragen, wie das Wetter an dem Reiseziel ist. So ist es Ihnen schon einige Male passiert, dass Sie die falsche Kleidung eingepackt haben. Es wäre gut, auch noch diese Information direkt parat zu haben. Die IT-Abteilung hat Sie nun endlich mit einem Tool versorgt, das Ihnen eine Vielzahl von Funktionalitäten zur Verfügung stellt, die Sie nutzen können, um eigene kleine Anwendungen nach Ihren Bedürfnissen zu bauen. Mit Freude setzen Sie sich daran und erstellen Ihre persönliche Reiseanwendung.“

Alle Probanden sollten dieses Szenario selbstständig umsetzen, Rückfragen an die Testbeobachter waren jedoch gestattet. Die anwesenden Testbeobachter konnten verfolgen, welche Schwierigkeiten auftraten und wie die Nutzer mit dem Tool umgingen. Anschließend folgte eine Befragung.

2.4.3 DURCHFÜHRUNG DER EVALUATION

Die Teilnehmer für die Evaluation verfügten über unterschiedliches Vorwissen im Bereich der IT, jedoch besaß keiner der Teilnehmer eine umfassende Informatik-Ausbildung. Alle Teilnehmer waren zwischen 19 und 25 Jahre alt und hatten entweder ihr Studium abgeschlossen oder waren noch Studenten.

Die Evaluation wurde in mehreren Schritten durchgeführt und lässt sich wie folgt gliedern:

1. Im Vorfeld wurden die Teilnehmer zu ihrem Vorwissen im Bereich Webanwendungen, Webservices und Service Komposition befragt.
2. Die Explorationsphase diente zur Orientierung und zum Kennenlernen des Tools. Den Probanden wurde die Möglichkeit geboten, das Tool auszuprobieren und Fragen zu der Funktionalität zu stellen. Anschließend folgte eine erste Befragung zu dem Eindruck der Probanden.
3. Im praktischen Teil modellierten die Teilnehmer das vorgegebene Anwendungsszenario mit dem ServFace Builder.
4. In einer abschließenden Befragung schilderten die Teilnehmer ihre Eindrücke über den Umgang mit den ServFace Builder und dessen Konzepte.

Die Evaluation umfasste zwei Befragungen: Zunächst probierten die Benutzer den ServFace Builder ohne große Einführung aus, anschließend wurden sie befragt. Nach der Klärung von offenen Fragen erstellten die Probanden den vorgestellten Anwendungsfall und wurden nochmals befragt. Durch diese Vorgehensweise wurde ersichtlich, welche Funktionen intuitiv gestaltet bzw. ohne Hilfestellung erfassbar sind.

2.4.4 ERGEBNISSE DER BEFRAGUNG

Die Ergebnisse der Evaluation resultieren aus der explorativen Phase, in der die Probanden zunächst ohne große Einführung mit dem ServFace Builder gearbeitet haben, und aus der Beobachtung und Befragung über die Erstellung des Anwendungsszenarios, wo die Benutzer bereits vertraut im Umgang mit dem ServFace Builder waren. Laut der Evaluation benötigen im Rahmen dieser Arbeit folgende Aspekte eine Optimierung:

- Der Anfangsprozess ist nicht klar: Nach Auswahl des Anwendungstypen wussten 67% der Testpersonen nicht, wie sie weiter vorgehen sollten bzw. welche Aktionen überhaupt möglich sind. Erst nach längerem Probieren und Hinweisen durch die Testbeobachter konnten die Probanden mit dem nächsten Schritt fortfahren.

- Es wurde als störend empfunden, dass der Pageswitcher nur temporär sichtbar ist. Statt dessen wurde vorgeschlagen, diesen permanent anzuzeigen mit der Möglichkeit einer Ausblendefunktion.
- Eine Vereinfachung des Kompositionsprozesses und der Navigation zwischen den Seiten wurde gefordert. Bei der Erstellung eines Interpage-Datenflusses störte die Nutzer das umständliche, manuelle Öffnen und Schließen der Seiten. Der Pageswitcher, der einen deutlich einfacheren Seitenwechsel ermöglicht, wurde von vielen Probanden nicht wahrgenommen.
- Das Drag & Drop-Prinzip zur Integration einer Service Operation in eine Seite wurde als nicht intuitiv befunden, das Hinzufügen per Doppelklick wurde vorgeschlagen.
- Zwei Drittel der Probanden bemängelten die Unterstützung des Tools als unzureichend. Daraus lässt sich die Notwendigkeit ableiten, die Funktionen intuitiver zu gestalten.
- Obwohl der ServFace Builder präsentationsorientiert arbeitet, hatte zwei Drittel der Testnutzer keine Vorstellung, wie die Endanwendung aussehen wird (Diagramm 2.5). Die Testnutzer konnten sich nicht vorstellen, wie die Bausteine zur Laufzeit umgesetzt werden sollen und wie mit den Bausteinen interagiert werden kann.
- Auf der anderen Seite habe manche Probanden versucht, die Webservices bereits in der Entwurfsansicht zu benutzen. So wurden in die Frontends Daten eingetragen und der Submit-Button aktiviert.
- In der Regel wurden lediglich sequentielle Anwendungen erstellt. Daraus lässt sich ableiten, dass für Einsteiger eine lineare Seitenansicht zunächst ausreicht, die komplexere Prozessansicht ist vor allem für erfahrenere Benutzer interessant.

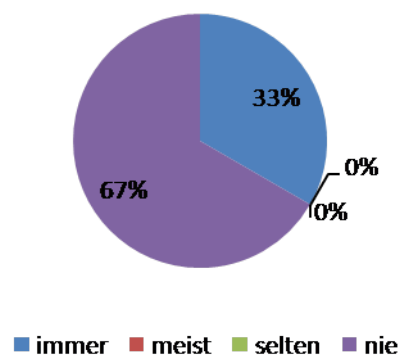


Abbildung 2.5.: Diagramm zu der Frage „Hatten Sie eine Vorstellung davon, wie die finale Anwendung aussieht?“

2.5 ZUSAMMENFASSUNG

In diesem Kapitel wurden die Rahmenbedingungen vorgestellt: zunächst wurde eine Einführung in die Grundkonzepte des Projektes ServFace gegeben und der aktuelle Stand des ServFace Builders geschildert. Die Evaluation, die im August 2009 mit Teilnehmern der definierten Zielgruppe stattfand, dient unter anderem als Grundlage der Motivation. Die Benutzerstudie hat gezeigt, dass an einigen Stellen Optimierungen erforderlich sind. Insgesamt wurde deutlich, dass dem Benutzer das allgemeine Verständnis für die Funktionalität bzw. für die zu erstellende Endanwendung fehlt. Einige Bedienkonzepte wie z.B. die Darstellung sowie Erstellung der Dialogstruktur oder der Pageswitcher bereitete den Benutzern Schwierigkeiten und genügt somit nicht den Ansprüchen einer intuitiven benutzerfreundlichen Bedienung. Das Ziel dieser Arbeit ist, das Verständnis der Zielgruppe zu fördern und eine benutzerfreundliche Interaktion zu ermöglichen.

3 ANALYSE UND VERWANDTE ARBEITEN

In diesem Kapitel werden die Grundlagen beschrieben, auf denen die Konzepte aufbauen und sie manifestieren. Je nach Themengebiet werden elementare Ansätze und der aktuelle Stand der Technik in Hinsicht auf die Ziele dieser Arbeit vorgestellt. Die im ServFace Builder erstellte Anwendung soll dem Benutzer möglichst in gewohnter Weise präsentiert werden. Dazu werden Konventionen für die plattformspezifische Darstellung (Abschnitt 3.2, 3.3) vorgestellt und bekannte Navigationsmuster (Abschnitt 3.1) erläutert. Bewährte Prinzipien sollen wiederaufgenommen werden, um den präsentationsorientierten Ansatz des ServFace Builders zu optimieren (Abschnitt 3.4). Desweiteren soll eine Ansicht wie die Übersicht der Dialogstruktur, die für den typischen Benutzer der Zielgruppe höchstwahrscheinlich unbekannt ist, möglichst intuitiv gestaltet werden. Dazu wird untersucht, welche Möglichkeiten es gibt sowie welche Aspekte bei möglichen Darstellungen zu beachten sind (Abschnitt 3.5). Am Ende jedes Abschnittes wird in einem Fazit zusammengefasst, welche Erkenntnisse gewonnen wurden unter Berücksichtigung der Aufgabengebiete dieser Arbeit.

3.1 NAVIGATION INNERHALB SEITENBASIERTER ANWENDUNGEN

Die Navigation ermöglicht die Fortbewegung innerhalb einer Anwendung. Seitenwechsel oder Sprünge innerhalb einer Seite können mittels Navigations- bzw. Interaktionselementen wie Buttons, Links oder Grafiken durch den Nutzer ausgelöst werden. Desweiteren gibt die Navigation als Orientierungshilfe einen Überblick über den Inhalt der aktuellen Seite und den derzeitigen Standpunkt innerhalb der Anwendung [Jac05]. Eine gute Navigation zeichnet sich dadurch aus, dass sie es jedem Besucher übersichtlich, leicht und interessant macht, sich auf den Internetseiten zu bewegen [sym10].

3.1.1 NAVIGATIONSSTRUKTUREN

Durch die Navigation wird die Struktur einer Anwendung erst nutzbar gemacht. Es existieren unterschiedliche Möglichkeiten, wie die Navigation innerhalb einer Anwendung aufgebaut

werden kann bzw. wie Seiten miteinander verknüpft werden können. Je nach Anwendungsszenario bietet sich ein anderes Navigationsmuster an, dabei soll sich der Nutzer möglichst intuitiv durch die Anwendung bewegen können. Im Folgenden werden gängige Navigationsstrukturen vorgestellt [Abo].

3.1.1.1 HIERARCHISCHE STRUKTUR

Die Seiten sind thematisch gruppiert und gehören zu einer übergeordneten Kategorie. Somit kann eine hierarchische Baumstruktur mit den Seiten als Knoten entwickelt werden. Von der Startseite aus kann der Benutzer zwischen den verschiedenen Hauptkategorien wählen. Um detaillierte Informationen zu erhalten, muss der Benutzer tiefer in die Hierarchie navigieren. Häufig wird eine Pfadnavigation als Orientierungshilfe eingesetzt, wo die Möglichkeit besteht, sowohl rückwärts als auch in Unterkategorien zu springen. Zu unterscheiden sind dabei folgende Navigationselemente:

Globale Navigation

Globale Navigationselemente sind die Hauptkategorien einer Anwendung [Adk05], zu vergleichen mit den Wegweisern innerhalb eines Supermarkts [Ltd], die einen Überbegriff für jedes Regal enthalten. Die globalen Elemente sind in den meisten Fällen permanent sichtbar und auf jeder Seite zu finden.

Lokale Navigation

Wenn eine Gruppe von Seiten inhaltsabhängig zusammengefasst und zu einer Hauptkategorie zugeordnet werden kann, bietet sich ein lokales Navigationssystem innerhalb dieser Untermenge an. Die lokalen Navigationselemente werden im Gegensatz zu den globalen lediglich temporär dargestellt. Abhängig von der ausgewählten Kategorie werden nur Navigationselemente der dazugehörigen Gruppe angezeigt. Es sind verschiedene Strukturen (Abbildung 3.1) für die lokale Navigation innerhalb einer hierarchischen Navigationsstruktur möglich [Adk05]:

- Down-to-Child: Verbindungen, die Zugang zu spezifischeren Informationen bieten, referenzieren in der Navigationshierarchie auf die Kindelemente eines Knotens.
- Across-to-Sibling: Verbindungen, die auf Seiten derselben Kategorie referenzieren, so dass der Nutzer innerhalb einer Ebene navigieren kann.
- Up-to-Parent: Verbindungen, die auf weniger spezifischere Informationen bzw. auf die Überkategorie referenzieren, so dass der Nutzer in die höhere Ebene wechseln kann.
- Down-to-Grandchild: Verbindungen, die auf Seiten referenzieren, die sich zwei Ebenen unter der aktuellen Seite befinden. Auf diese Weise kann der Benutzer schnell und unkompliziert zu den Unterthemen einer Kategorie navigieren.

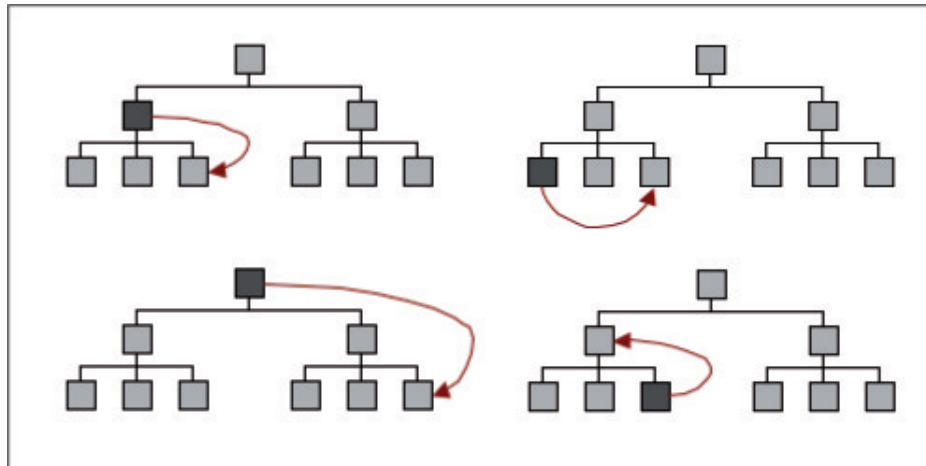


Abbildung 3.1.: Lokale Navigationsstrukturen: Down-to-Child, Across-to-Sibling, Up-to-Parent, Down-to-Grandchild [Adk05]

3.1.1.2 SEQUENTIELLE STRUKTUR

Um den Nutzer in einem Prozess wie z.B. Einkaufen in einem Shopping-Portal oder ein Tutorial zu leiten, kann eine sequentielle Struktur vorgegeben werden, innerhalb der kein Sprungmanöver möglich ist. Der Benutzer kann sich vorwärts („next“) und eventuell zurück („previous“) innerhalb der definierten Abfolge bewegen. Diese Einschränkung sollte wohlüberlegt sein, denn sie kann schnell zu Frustration führen.

3.1.1.3 NETZ STRUKTUR

Alternativ können die Seiten innerhalb einer Netzstruktur organisiert werden wie zum Beispiel das World Wide Web (WWW). Diese Struktur hat keine bestimmte Ordnung und kann dadurch schnell chaotisch erscheinen. Dementsprechend wird sie nur selten eingesetzt.

Direkter Link

Diese Verbindung referenziert direkt auf eine Seite unabhängig von der aktuellen Position wie z.B. der Home-Button.

3.1.2 NAVIGATIONSHILFEN

Navigationshilfen beinhalten die Interaktionselemente, die einen Seitenwechsel bzw. die Bewegung innerhalb einer Seite ermöglichen [Jac05], zudem dienen sie als Orientierungshilfe.

Im Folgenden werden mögliche Darstellungen und Formen von Navigationshilfen vorgestellt [Jac05]:

Navigationsleisten

Am häufigsten werden die Navigationselemente in Navigationsleisten eingebunden.

Navigationsleisten: Frames

Alternativ kann die Navigation innerhalb von Frames abgebildet werden. Frames sind definierbare Bereiche innerhalb einer HTML-Seite. Der Inhalt der einzelnen Frames kann unabhängig voneinander ausgetauscht werden. Zunächst unterscheidet sich eine Seite, die aus Frames besteht, nicht von einer anderen. Wenn der Benutzer jedoch innerhalb eines Frames scrollt, bleiben die Inhalte der übrigen Frames an derselben Position stehen. Somit ist die Navigationsleiste immer sichtbar und der Benutzer kann jederzeit ohne zu scrollen zu einer anderen Seite springen.

Navigationsleisten: Eigene Fenster

Die Navigation kann in ein eigenes Fenster ausgelagert werden, das beim Öffnen der Startseite erscheint. Diese Navigation kann jedoch bei einer Benutzergruppe ohne große Erfahrung im Umgang mit dem Internet eher für Verwirrung sorgen. Falls Pop-ups deaktiviert sind, kann das Fenster erst gar nicht geöffnet werden. In diesem Fall ist keine Navigation mehr möglich.

Drop-Down Menu (Aufklappende Navigationsleisten)

Bei größeren Anwendungen mit einer hierarchischen Navigationsstruktur bieten sich aufklappende Navigationsleisten an. Zunächst werden nur die verschiedenen Hauptthemen angezeigt. Durch Selektieren eines Hauptthemas erscheint in der Leiste eine temporäre Liste der Unterpunkte dieser Kategorie. Sobald der Benutzer ein anderes Thema auswählt, verschwindet die erste Liste wieder. Somit wird die Navigationsleiste nicht zu lang, gleichzeitig wird dem Benutzer eine genaue Übersicht der Unterpunkte dargeboten. Diese Darstellung bietet sich bei einer Webanwendung in einer vertikalen Navigationsleiste an.

Drill-Down Menu

Das Drill-Down Menu wird bei hierarchischen Navigationsstrukturen angewendet. Es wird immer nur eine Ebene des Navigationsbaumes angezeigt: bei Start der Anwendung werden zunächst die Hauptkategorien dargestellt, durch Auswahl eines Elementes wird der komplette Inhalt durch die entsprechenden lokalen Elemente ausgetauscht [SM]. Dieses Menu wird normalerweise mit einer Pfadnavigation kombiniert, damit sich der Benutzer orientieren kann und bietet sich insbesondere für eine hierarchische Navigation innerhalb einer mobilen Anwendung an.

Pop-up Navigationsleisten

Alternativ zu den aufklappenden Navigationsleisten können die Untermenupunkte nur kurzzeitig angezeigt werden. Durch Klick oder Rollover auf ein Hauptthema erscheinen die Unterpunkte innerhalb eines Pop-up Menus. Dies hat den Vorteil, dass nur wenig Platz für das Menu benötigt wird. Jedoch sind die Auswahlmöglichkeiten nicht dauerhaft sichtbar, zudem verdeckt das Pop-up Menu möglicherweise Teile der Seite, was in manchen Fällen nicht nur unschön aussieht, sondern auch der Orientierung schaden kann. Die Pop-up Navigation wird normalerweise nur in der oberen Horizontalen eingesetzt.

Karteikarten und Reiter

Häufig wird die Navigation in Karteikarten und Reitern (auch Registerkarten genannt) integriert. Diese Metapher leitet sich aus der Technik ab, die im Büro seit Generationen verwendet wird und setzt sich immer mehr in Computerprogrammen wie beispielsweise Microsoft Office Produkten ³ durch. Auch in Webseiten wird dieses Prinzip gerne angewendet. Unter Reiter (englisch „tab“) versteht man das kleine Etikett, auf dem die Bezeichnung steht. Die Karteikarte ist die dazugehörige Inhaltsseite. Die Anzahl der Reiter sollte aus Gründen der Übersichtlichkeit auf maximal neun Reiter begrenzt werden.

Pfadnavigation

Die Pfadangabe erleichtert dem Benutzer die Orientierung. Innerhalb der Pfadnavigation stehen die Namen der Seiten, die der Benutzer auf seinem Weg durchlaufen hat (Abbildung 3.2).

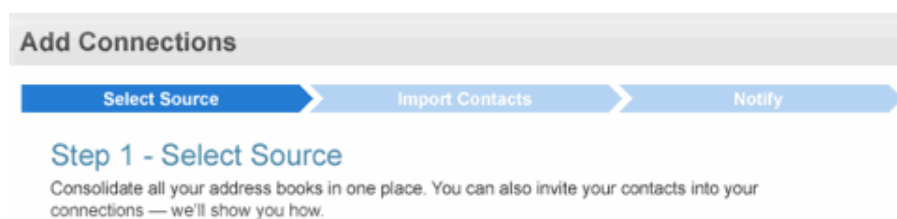


Abbildung 3.2.: Pfadnavigation⁴

Zudem ermöglicht sie dem Benutzer, zurück zu einer der vorherig aufgerufenen Seiten zu gehen. Diese Navigation wird meist bei tiefen Hierarchien und sequentiellen Strukturen angewandt.

³ <http://office.microsoft.com/de-ch/>

⁴ <http://developer.yahoo.com/ypatterns/navigation/bar/progress.html>

Eingebettete Links

Die einfachste Darstellung ist der im Text eingebettete Link. Meist wird diese Navigation für einen Link auf verwandte Themen oder um zusätzliche Informationen zu erhalten eingesetzt.

Home-Button

Der Benutzer sollte jederzeit die Möglichkeit haben, mittels eines deutlich platzierten Links wieder zur Startseite zurückkehren zu können. Meist wird das Firmenlogo als Link zur Startseite eingesetzt.

Back-Button

Innerhalb einer hierarchischen Struktur besteht eine weitere Orientierungsmöglichkeit in einem Link zu der Seite, die in der Hierarchie der Anwendung eine Ebene höher liegt als die aktuelle. Durch die Benennung des Links bzw. Buttons kann der Benutzer die entsprechende Seite einordnen. Der Button sollte eindeutig platziert sein (z.B. unten links auf der Seite), so dass dem Benutzer klar ist, dass durch diesen Button einen Schritt zurück gegangen wird. Bei einer aufklappenden Navigationsleiste oder Pfadangabe kann auf einen Back-Button verzichtet werden.

Navigation in der Fußzeile

Für die Verlinkung auf administrative Inhalte wie z.B. Angaben zum Copyright, Benutzungshinweise etc. wird oftmals eine Navigation in der Fußzeile eingesetzt [Ltd]. Wie globale Navigationselemente sind diese auf jeder Webseite zu finden. Navigationselemente innerhalb der Fußzeile werden normalerweise in Textform dargestellt. In manchen Fällen werden globale Navigationselemente nochmals in der Fußzeile angezeigt. Dadurch muss der Benutzer, unten am Ende der Seite angekommen, nicht erneut hochscrollen, um eine neue Seite aufzurufen. Die Duplizierung der Links besitzt sowohl Vor- als auch Nachteile und deren Einsatz sollte gut überlegt werden.

In-page Navigation

Bei längeren Seiten mit verschiedenen Themengebieten bietet es sich an, am Anfang der Seite Navigationsmöglichkeiten zu den verschiedenen Bereichen bereitzustellen. Üblicherweise wird eine In-page Navigation bei einer Seite angeboten, bei der mehr als zwei Scrolls notwendig sind, um das Ende zu erreichen. Nach jedem Abschnitt soll zusätzlich eine Verlinkung auf den Anfang der Seite bestehen.

3.1.3 FAZIT

Innerhalb der Diplomarbeit sollen Konzepte für die hierarchische, sequentielle und eine frei konfigurierbare Navigation entwickelt werden. Abhängig von der Zielplattform sind geeignete Navigationshilfen einzusetzen. Die Pfadnavigation ist hilfreich zur Orientierung und kann für mehrere Navigationsarten eingesetzt werden.

3.2 LAYOUTEMPFEHLUNGEN FÜR WEBANWENDUNGEN

Eine Webseite setzt sich meist aus verschiedenen Elementen zusammen: Navigationsmenu, Interaktionselemente, Hilfsinformationen wie z.B. eine Kontaktinformation oder ein Suchfeld. Im Rahmen dieser Arbeit sollen jedoch nur Layouteigenschaften für das Navigationsmenu sowie aktuelle Inhalte einer Seite – Text, Interaktionselemente und Grafiken – betrachtet werden. In folgenden Abschnitten wird untersucht, welche gängigen Aufteilungen es innerhalb von Webseiten und welche Darstellungsmöglichkeiten es für Formulare gibt.

3.2.1 AUFTEILUNG DER SEITE

Im Folgenden werden gängige Aufteilungen einer Seite vorgestellt [Web]. Dabei wird im Zusammenhang mit dieser Arbeit nur die Positionierung der Navigationsleisten und des eigentlichen Inhalts betrachtet, Elemente wie Werbung und Hilfsinformationen werden nicht miteinbezogen. In Abhängigkeit von der Navigationsleiste, in der sich die Interaktionselemente befinden, die einen Seitenübergang auslösen können, werden typische Aufteilungen präsentiert. Der Navigationsbereich wird durch eine blaue Einfärbung repräsentiert, der Platz für die Seitenüberschrift ist beige eingefärbt, der aktuelle Inhalt der Seite (Informationen in Form von Text und Grafiken, Interaktionselemente) in grauer Farbe (Abbildung 3.3).

Vertikale Navigation

Die Navigation ist vertikal ausgerichtet und befindet sich im linken Randbereich der Seite. Im oberen Bereich steht die Überschrift der Seite (beiger Bereich). Der Navigationsbereich muss nicht die gesamte Höhe einnehmen.

Horizontale Navigation

Alternativ kann der Navigationsbereich direkt unter der Seitenüberschrift platziert werden. Dies hat zur Folge, dass der jeweilige Textinhalt die ganze Breite der Seite einnimmt. Bei einer Auflösung größer als 600 Pixel hat das eine schlechte Lesbarkeit zur Folge. Optional kann die Fläche am linken und rechten Rand mit Grafiken gefüllt werden.

Vertikale und horizontale Navigation

Die gängigste Aufteilung einer Seite ist das „gekippte L“. Dabei wird die vertikale mit der horizontalen Navigation kombiniert. Diese Aufteilung bietet sich bei einer hierarchischen Navigationsstruktur an.

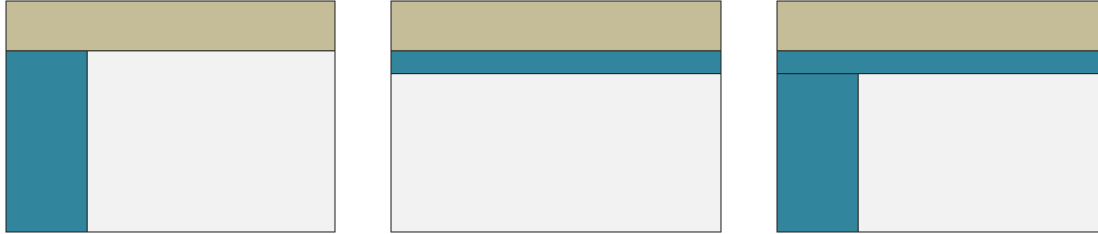


Abbildung 3.3.: Aufteilungsmöglichkeiten einer Seite: Vertikal, Horizontal, Kombination

Das adäquate Aufteilungsprinzip

Die passende Positionierung des Navigationsbereichs ist von mehreren Kriterien abhängig [Web]:

- Bei einer Auflösung von 800 x 600 Pixeln und nicht mehr als 10 Navigationselementen empfiehlt sich ein horizontales Menu.
- Bei mehr als 15 Navigationselementen sollte eine vertikale Navigation an der linken Seite eingesetzt werden.
- Die Darstellung des Navigationsbereichs erfolgt ebenfalls abhängig vom Seiteninhalt. Wenn viele Seiten einer Webanwendung über einen umfangreichen Inhalt verfügen, eignet sich ein vertikales Navigationsmenu.
- Für ein dynamisches Drop-Down Menu wird meist eine horizontale Navigation bevorzugt, da das Menu eher nach unten (horizontale Navigation) anstatt zur Seite (vertikale Navigation) ausgeklappt wird. Wird ein Menu nach unten ausgeklappt, sind im Fall einer horizontalen Navigation alle übrigen Navigationselemente stets sichtbar.
- Die vertikale Navigation ist leichter zu aktualisieren. Auf Wunsch können dem Menu nach unten hin weitere Navigationselemente hinzugefügt werden. Bei einer horizontalen Navigation sollte die Anzahl der Navigationselemente nicht größer als 10 sein, demzufolge ist eine Erweiterung nur begrenzt realisierbar. Eine Möglichkeit besteht darin, der horizontalen Navigation mehrere Zeilen hinzuzufügen.

3.2.2 LAYOUTS FÜR FORMulare

Die meisten Webanwendungen beinhalten eine formularbasierte Dateneingabe und Konfiguration. Ein Formular kann aus verschiedenen Interaktionselementen wie z.B. Texteingabe,

Kontrollkästchen für Multiple Choice, Auswahlmenüs, Landkarten, Buttons etc. bestehen, wobei den Eingabefeldern normalerweise ein Label für die Beschreibung beigelegt ist. Nachdem der Benutzer das Formular ausgefüllt hat, bestätigt er mit Hilfe irgendeiner Form des „Submit“-Buttons, um die Daten an den Server zu schicken, woraufhin ein Webservice aufgerufen und eine Antwort generiert wird.

Dabei existieren unterschiedliche Varianten der Anordnung von Interaktionselementen bzw. Eingabeelementen. Eingabeelemente sollten in logischen Gruppen organisiert werden, damit das Gehirn des Betrachters die Felder als zusammenhängend erkennt [MK06]. Im Folgenden werden verschiedene Möglichkeiten betrachtet, Eingabeelemente im Zusammenhang mit ihren Labels zu positionieren [Wro].

Vertikale Labels

Wenn die Ausfüllung des Formulars möglichst schnell gehen soll und die einzugebenden Daten dem Benutzer bereits bekannt sind (wie z.B. Name oder Adresse), eignet sich eine vertikale Anordnung der Labels und der dazugehörigen Eingabefelder. Durch räumliche Nähe ist die Beziehung des Labels mit dem entsprechenden Eingabefeld zu erkennen, die konsistente Ausrichtung der Labels und Eingabefelder reduziert die Augenbewegung und damit die Bearbeitungszeit. Die Leserichtung ist eindimensional, nämlich nach unten (Abbildung 3.4).

Vertical Labels

Label

Longer Label

Even Longer Label

One More Label
☒ Value 1
☐ Value 2

Primary Action

Advantage:
 Adjacent Label and corresponding Input field

Advantage:
 Rapid Processing

Disadvantage:
 Increased vertical space

Abbildung 3.4.: Formular mit vertikal angeordneten Labels und Eingabefeldern [Wro]

Horizontale Labels

Wenn die einzugebenden Daten dem Benutzer nicht geläufig und dementsprechend nicht eindeutig zu gruppieren sind, bietet sich eine horizontale Anordnung der Labels und Eingabefelder an. Dabei empfiehlt sich, die Labels durch Fettdruck hervorzuheben, damit sie neben den Eingabefeldern nicht ignoriert werden.

Links ausgerichtete Labels

Eine gängige Anordnung ist es, die Labels links auszurichten (Abbildung 3.5). Auf diese Weise kann der Benutzer sich zunächst mit den verlangten Daten durch die Labels vertraut machen, ohne beim Lesen von den Eingabefeldern unterbrochen zu werden. Im Fall einer Linksausrichtung der Labels besteht oft ein großer Abstand zwischen Label und Eingabefeld, da sich der Abstand an dem längsten Label orientiert. Dies führt unter Umständen zu einer längeren Bearbeitungszeit, da die Leserichtung nicht mehr eindimensional ist und der Benutzer zunächst das entsprechende Label zuordnen muss.

Right-Justified Horizontal Labels

Label

Longer Label

Even Longer Label

One More Label ☒ Value 1
☐ Value 2

Primary Action

Disadvantage:
Reduced readability

Advantage:
Adjacent Label and corresponding Input field

Advantage:
Reduced vertical space

Label

Longer Label

Even Longer Label

One More Label ☒ Value 1
☐ Value 2

Primary Action

Abbildung 3.6.: Formular mit rechts ausgerichteten Labels in horizontaler Anordnung [Wro]

Backgrounds & Rules

Label:	<input type="text"/>
Longer Label:	<input type="text" value="Select Value"/>
Even Longer Label:	<input type="text"/>
One More Label:	<input checked="" type="radio"/> Value 1 <input type="radio"/> Value 2

Additional Visual Elements

1 —	Label:	<input type="text"/>	— 9
2 —	Longer Label:	<input type="text" value="Select Value"/>	— 10
3 —	Even Longer Label:	<input type="text"/>	— 11
4 —	One More Label:	<input checked="" type="radio"/> Value 1	— 12
5 —		<input type="radio"/> Value 2	— 13
6 —			— 14
7 —			— 15

8

Impaired Scanning

↓	Label:	<input type="text"/>
↓	Longer Label:	<input type="text" value="Select Value"/>
↓	Even Longer Label:	<input type="text"/>
↓	One More Label:	<input checked="" type="radio"/> Value 1 <input type="radio"/> Value 2

Abbildung 3.7.: Trennung der Inhalte mit visuellen Hilfsmitteln

Obwohl dieser Ansatz vorteilhaft erscheint, verursacht er auch Probleme. Durch dieses Layout entstehen neben dem relevanten Inhalt – Labels und Eingabfelder – neue Elemente: die vertikale Mittellinie und horizontale Linien, die durch die verschiedenen Farben entstehen, sowie jede farbige Hintergrundbox. Wie Edward Tufte [Tuf] formulierte: „Information consists of differences that make a difference.“ Das heißt, jedes Element, welches das Layout nicht unterstützt, schadet ihm. In diesem Fall untersucht das Auge nicht nur die UI-Elemente, sondern auch jede Linie und Hintergrundbox.

Diese Feststellung soll jedoch nicht zur Folge haben, überhaupt keinen farbigen Hintergrund und Linien innerhalb von Formularen einzusetzen. Wenn eine zusammengehörende Gruppe von Informationen betont werden soll, kann beispielsweise eine dünne horizontale Linie oder eine helle Hintergrundfarbe den Zusammenhang der Daten hervorheben (Abbildung 3.8).

Separating Related Content

Label:

Longer Label:

Even Longer Label:

One More Label: ☒ Value 1 ☐ Value 2

Label:

Longer Label:

Even Longer Label:

Primary Action

Abbildung 3.8.: Trennung der Inhalte mit visuellen Hilfsmitteln

Insbesondere kann man diese Gestaltungselemente (Linien und farbiger Hintergrund) einsetzen, um die Priorität einer Aktion innerhalb eines Formulars zu verdeutlichen. Die Hauptaktion wie z.B. „Bestätigen“ oder „Speichern“ eines Formulars muss visuell von den übrigen Formularelementen abgegrenzt werden und sollte vertikal unter den Eingabefeldern positioniert sein. Auf diese Weise wird dem Benutzer eine klare Richtlinie für die Bearbeitung des Formulars vorgegeben. Wenn ein Formular über verschiedene Aktionen verfügt („Fortfahren“, „Zurückgehen“), sollten diese visuell reduziert erscheinen, um sie als Aktionen mit sekundärer Priorität zu markieren. Dadurch können potentielle Fehler vermieden werden (Abbildung 3.9).



Abbildung 3.9.: Priorität der Aktionen durch Farbe signalisiert

3.2.3 FAZIT

Anhand der Layoutempfehlungen kann die Visualisierung der verschiedenen Navigationsstrukturen und der Einsatz von Navigationshilfen bestimmt werden: Für eine sequentielle Navigationsstruktur wird zur Orientierung meist eine Pfadnavigation in der oberen Seitenleiste verwendet, die Verbindung auf die nächste Seite wird am Ende der Seite visualisiert. Bei einer hierarchischen Navigation können die globalen und lokalen Elemente einerseits durch verschiedene Aufteilungsprinzipien der Seite getrennt positioniert werden, wie z.B. eine Kombination der horizontalen Seitenleiste, die die globalen Elemente enthält, und der vertikalen Seitenleiste, in der die temporären lokalen Elemente platziert sind. Eine andere Möglichkeit bietet sich durch den Einsatz verschiedener Navigationshilfen in der oberen oder linken Seitenleiste.

Es empfiehlt sich eine horizontale Anordnung der Eingabefelder und Labels mit Linksausrichtung, da dies ästhetischer erscheint.

3.3 LAYOUTEMPFEHLUNGEN FÜR MOBILE ANWENDUNGEN

In diesem Abschnitt werden Guidelines wie die „Mobile Web Initiative“ (MWI) vom W3C (World Wide Web Consortium) zur Spezifikation von Layouteigenschaften für die Inhaltsdarstellung einer Anwendung für mobile Geräte wiedergegeben [Ini09].

Bildschirmauflösung

Innerhalb dieser Arbeit wird bei mobilen Geräten von einer Bildschirmauflösung von 480 x 320 Pixel ausgegangen. Bei Geräten mit niedriger Auflösung kann die Darstellung von umfangreichen Inhalten schnell zu einem großem Scrollaufwand führen. Um Scrolling zu vermeiden, sollten umfangreiche Inhalte auf mehrere Seiten aufgeteilt und mit einem „Next“-Button durchlaufen werden.

Seitengröße

Die Seitengröße soll wegen der begrenzten Speicherkapazität und zur Vermeidung von längeren Ladezeiten nicht zu groß sein.

Scrolling

Scrolling soll möglichst vermieden werden und ansonsten wenigstens auf eine Richtung begrenzt sein.

Farbe

Farben müssen vorsichtig eingesetzt werden. Mobile Geräte verfügen meist über keine guten Farbkontraste, zudem werden sie auch unter nicht-idealen Lichtbedingungen benutzt. Im schlechtesten Fall ist die durch Farbe hervorgehobene Information für den Benutzer überhaupt nicht sichtbar. Auf keinen Fall sollte blauer oder violetter Text eingesetzt werden, da dieser in erster Linie mit Hyperlinks assoziiert wird – insbesondere bei Geräten, die keine unterstrichenen Links unterstützen. Es sollte eine farblose Version der Webanwendung verfügbar sein, ansonsten sollten die Farben wenigstens in einem ausreichenden Kontrast zueinander stehen.

Labels für Interaktionselemente

Das Label eines Interaktionselements soll dem Platzverhältnis gemäß angemessen positioniert sein. Die Zugehörigkeit zum entsprechenden Interaktionselement soll aus der Darstellung deutlich hervorgehen.

Navigationsbereich

Die Navigationsleiste, in der sich die Interaktionselemente zur Auslösung eines Seitenübergangs befinden, sollte am oberen oder unteren Ende der Seite platziert sein [Ltd10]. Bei einem Gerät mit niedriger Auflösung kann es sein, dass die Navigationselemente den größten Teil des Fensters einnehmen. Wenn die Navigationselemente am Anfang der Seite situiert sind, muss der Benutzer runterscrollen, damit er zum eigentlichen Inhalt der Seite gelangt.

Frames

Es sollen generell keine Frames benutzt werden, da viele Geräte diese nicht unterstützen.

Fazit

Durch die Layoutempfehlungen werden konkrete Vorgaben für das Layout gesetzt: Alle Navigationsmuster sollen möglichst in eine horizontale Seitenleiste integriert sein. Eine hierarchische Navigation mit der typischen Verteilung der globalen Elemente in der oberen horizontalen und der lokalen Elemente in der linken vertikalen Seitenleiste soll vermieden werden, da in diesem Fall die Navigation bereits den gesamten Platz einnehmen könnte. Bei einer hierarchischen Navigation kann entweder ein Drop-Down oder ein Drill-Down Menu eingesetzt werden. Beide Varianten haben ihre Vor- und Nachteile, jedoch ist ein Drill-Down Menu für Geräte mit numerischer Tastatur einfacher zu bedienen [Pat10]. Eine zusätzliche Pfadnavigation ist insbesondere für mobile Anwendungen hilfreich, um die Orientierung auch bei einer komplexen Navigationsstruktur bzw. einer tiefen Hierarchie nicht zu verlieren. Für die sequentielle Navigation empfiehlt sich wie bei der Webanwendung eine Pfadnavigation in der oberen sowie ein Interaktionselement in der unteren Seitenleiste für die Verbindung auf die nachfolgende Seite.

Für die Anordnung der Labels und Eingabeelemente empfiehlt sich aufgrund des beschränkten Platzes eine vertikale Anordnung.

3.4 WYSIWYG-WERKZEUGE

Die Abkürzung WYSIWYG steht für „What you see is what you get“. Dies ist eine Bezeichnung für Programme, die den Inhalt bzw. dessen Layout schon so am Bildschirm anzeigen, wie er später ausgegeben (z.B. gedruckt) wird [Mro07]. Insbesondere HTML-Editoren arbeiten oft nach dem WYSIWYG-Prinzip.

Bei der Modellierung mit HTML-Editoren wie Adobe Dreamweaver ⁵ oder Microsoft Frontpage ⁶ kann man auswählen, ob UI-Elemente wie Textbereiche oder Buttons in der Entwurfsansicht mit oder ohne „Formular-Tags“ angezeigt werden sollen (Abbildung 3.10). Die Formular-Tags werden zur Laufzeit nicht mehr dargestellt, wodurch die Unterscheidung zwischen Entwurfsansicht und Laufzeit leichter fällt.

Fazit

Es muss entschieden werden, ob die Verwendung von Formulartags nicht eine zu große Abweichung von der präsentationsorientierten Darstellung ist.

⁵ <http://www.adobe.com/products/dreamweaver/>

⁶ <http://office.microsoft.com/de-de/frontpage-help/>

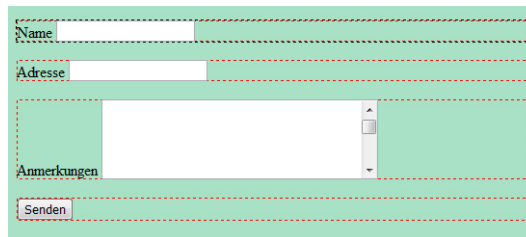


Abbildung 3.10.: Adobe Dreamweaver: Darstellung der UI-Elemente innerhalb von Formular-Tags

3.5 DARSTELLUNG DER DIALOGSTRUKTUR

In diesem Unterkapitel soll untersucht werden, welche Darstellungsmöglichkeiten sich für die Dialogstruktur anbieten. Die Dialogstruktur als Menge von Seiten und Seitenbeziehungen kann auf einen Graphen zurückgeführt werden. Dementsprechend sollen verschiedene Alternativen für die Darstellung von Graphen und den Umgang mit Graphen vorgestellt werden. Die Dialogstruktur dient der Visualisierung der Navigationsstruktur der zu erstellenden Anwendung. Eine weitere Orientierungsmöglichkeit bieten bereits bewährte Darstellungen der Navigationsstruktur in Webanwendungen oder Werkzeugen zur Erstellung einer Webanwendung. Die Betrachtung der Dialogstruktur soll jedoch auch von einem abstrakteren Standpunkt aus analysiert werden. Somit erfolgt eine Präsentation verschiedener Möglichkeiten zur Visualisierung von Beziehungen aus dem Bereich der endanwenderorientierten Entwicklung.

3.5.1 GRAPHEN

Wenn inhärente Beziehungen zwischen einer unstrukturierten Menge von Daten bestehen, kann man diese durch einen Graphen visualisieren. Dabei werden die Daten durch Knoten und die Beziehungen als Kanten repräsentiert. Die Abbildung in Form eines Graphen wird in vielen Bereichen angewendet: beispielsweise erleichtert die Darstellung eines großen Dateisystems durch einen Graphen die Navigation, auch in Sitemaps findet diese Darstellung eine häufige Anwendung.

Graphenvisualisierung findet in vielen Bereichen statt: für die Repräsentation einer Hierarchie in einem Organisationsdiagramm, in Taxonomien, in wissenschaftlichen Anwendungsgebieten wie der Biologie für evolutionäre Bäume und der Chemie für molekulare Karten. Andere Einsatzgebiete sind objektorientierte Systeme, Datenstrukturen, Echtzeitsysteme, Zustandsdiagramme, Datenflussdiagramme etc. [HM00].

Ein Graph besitzt eine Menge von Knoten und Kanten, kann ungerichtet oder gerichtet sein und eine Hierarchie beinhalten. Nachstehend werden zunächst die Eigenschaften eines Graphen beschrieben, die dazu beitragen, eine gut lesbare Darstellung zu erhalten und die Interaktion mit dem Graphen erleichtern. Anschließend erfolgt eine Vorstellung von verschiedenen Darstellungsmöglichkeiten.

3.5.1.1 EIGENSCHAFTEN EINES GRAPHEN

Es existieren mehrere ästhetische Eigenschaften, um eine anschauliche Abbildung von ungerichteten Graphen zu erhalten [BETT94], [HM00]. Die Hauptkriterien sind folgende:

- Überkreuzungen der Kanten vermeiden (Planarität)
- Symmetrie einhalten
- Knicke und Biegungen in den Kanten vermeiden
- Kantenlängen einheitlich halten
- Summe der Kantenlängen minimieren
- Knoten gleichmäßig verteilen
- Größe des Graphen möglichst klein halten

Um die Effektivität eines Diagramms bzw. Graphen zu steigern, wird bei Textelementen oft eine visuelle Gewichtung durch Schriftgröße, kursive oder farbige Markierung vorgenommen. Dadurch wird die Aufmerksamkeit des Lesers erregt und ihm damit die Priorität des entsprechenden Elementes vermittelt. Die wichtigsten Konzepte und Zusammenhänge können ebenfalls visuell hervorgehoben werden [Moo07].

3.5.1.2 DARSTELLUNGSMÖGLICHKEITEN FÜR EINEN GRAPHEN

Für einen Graphen als Menge von Knoten und Kanten existieren unterschiedliche Darstellungsmöglichkeiten. Im Folgenden werden verschiedene Visualisierungen für Graphen und Bäume (hierarchischer Graph) vorgestellt [BETT94].

Graphen:

- **Polygonal:** Eine Menge von Knoten und Kanten kann standardmäßig auf eine polygonale Darstellung abgebildet werden. Es entsteht ein Vieleck mit mehrkantigen Verbindungen. Die polygonale Abbildung ist die Grundform der Graphen-Visualisierung.

- **Geradlinig:** Jede Verbindung wird auf eine gerade Linie abgebildet, d.h. es sind keine Biegungen vorhanden. Diagramme (wie z.B. ER-Diagramme) werden meist nach diesem Standard präsentiert.
- **Orthogonale Linien:** Jede Kante wird als Kette von horizontalen und vertikalen Segmenten dargestellt. Um eine orthogonale Darstellung von einem einfachen polygonalen Graphen zu erhalten, müssen drei Phasen durchlaufen werden:
 1. Zunächst muss eine planare Darstellung des Graphen generiert werden.
 2. Anschließend muss jede Kante als Kette von orthogonalen Liniensegmenten abgebildet werden.
 3. Zuletzt soll der Umfang reduziert werden, so dass eine kompakte Darstellung entsteht.
- **Liniensegmente:** Die Knoten werden durch horizontale, die Kanten durch vertikale Segmente repräsentiert.
- **Azyklischer Graph:** Diese Graphenform wird häufig für hierarchische Strukturen wie z.B. PERT Diagramme oder ISA Hierarchien eingesetzt. Dabei ist es üblich, dass die Kanten alle in die gleiche Richtung zeigen, z.B. von oben nach unten oder von links nach rechts.
- **Gerichteter Graph mit bipolarer Orientierung:** Ausrichtung der Kanten eines ungerichteten Graphen, so dass ein gerichteter azyklischer Graph mit exakt einer Quelle (Knoten ohne eingehende Kanten) und einer Senke (Knoten ohne ausgehende Kanten) generiert wird.

Geradlinige und orthogonale Graphen sind Spezifikationen des polygonalen Graphen. Innerhalb eines polygonalen Graphen kann es auch kurvige Kanten geben. Geradlinige Graphen werden gewöhnlich in Büchern und wissenschaftlichen Veröffentlichungen über Graphentheorie benutzt, wohingegen orthogonale Graphen eher für kreisförmigen Schemata und Software Engineering Diagramme eingesetzt werden. Planarität ist in erster Linie ein ästhetisches Anliegen [BETT98].

Hierarchische Graphen:

- **Wurzelbaum:** Wurzelbäume werden für die Visualisierung von Hierarchien wie z.B. von Stamm- oder Suchbäumen eingesetzt. Gewöhnlich besteht die Darstellung des Wurzelbaumes in einer planaren geradlinigen oder orthogonalen Abbildung. Dabei schließen die meisten Darstellungen folgende ästhetische Eigenschaften mit ein:
 - Die Knoten werden entlang von horizontalen Linien abhängig von ihrer Ebenentiefe platziert.

- Der Abstand zwischen zwei aufeinanderfolgenden Knoten innerhalb einer Ebene ist minimal.
- Die Größe des Baumes ist so klein wie möglich gehalten.
- Die linken und rechten Kinder eines Knotens v sind links und rechts von v positioniert.

Alternativ können die Knoten durch Boxen, Vater-Kind Beziehungen durch Schachtelungen repräsentiert werden; ein Kindknoten bzw. -blatt wird innerhalb des Vaterknoten platziert („inclusion“). Eine andere Variante besteht darin, die Anordnung „umzukippen“ („tip over“). Anstatt die Kindknoten horizontal anzuordnen, können sie ebenfalls in vertikaler Reihenfolge dargestellt werden, d.h. alle Geschwisterknoten befinden sich auf einer vertikalen Orthogonalen.

- **Freie Bäume:** Freie Bäume besitzen keine Wurzel. Die für die Wurzelbäume angewandten Algorithmen können modifiziert werden, so dass beispielsweise eine radiale Darstellung mit einer konzentrisch kreisförmigen Anordnung der Knoten um das graphentheoretische Zentrum des Baumes entsteht.

3.5.1.3 INTERAKTION MIT GRAPHEN

Zoomen und Verschieben sind gängige Hilfsmittel, um von einem großen Graphen eine Übersicht gewinnen zu können. Man unterscheidet zwischen **geometrischem** und **semantischem Zooming** [HM00]. Geometrisches Zooming bezieht sich auf das bloße Vergrößern bzw. Verkleinern von Daten, wohingegen semantisches Zooming den Detaillierungsgrad der Informationen verändert. Bei einem großen Graphen sollen die einzelnen Objekte nicht komplett, sondern durch einen repräsentativen Titel visualisiert werden. Je mehr der Benutzer an den Graphen heranzoomt, desto mehr Detailinformationen werden sichtbar. Dabei muss entschieden werden, welche Daten zu welchem Zeitpunkt angezeigt werden sollen. Die Interaktion kann durch das Zooming und Verschieben umständlich sein, da jedes Heranzoomen vor dem Verschieben rückgängig gemacht werden muss.

Das System „Instant Bookplex“ setzt nach diesem Konzept ein zoombares UI für die Navigation durch eine Dokumentensammlung um [BPGN04]. Die Dokumente werden durch farbige Rechtecke mit einer Kurzbeschreibung (Autoren, Titel) und einem Bild in Thumbnailgröße abgebildet. Je näher der Benutzer an das Dokument heranzoomt, desto mehr wird von dem Dokument preisgegeben. Die Farbe des Rechtecks gibt die Gewichtung des Dokumentes an, die Sättigung, ob der Benutzer das Dokument schon einmal geöffnet hat. Aus dieser einfachen Präsentation kann der Benutzer für die Navigation und Organisation viele nützliche Informationen erhalten.

Wenn innerhalb eines Graphen gezoomt wird, geht der übrige Kontext verloren. Verschiedene Fokus-Kontext Methoden beschäftigen sich mit diesem Problem. Beispielsweise kann der Verlust des Kontextes durch die „Fish-Eye-View“ verhindert werden. Dabei wird ein Teil des Graphen vergrößert, wobei aber der übrige Graph in Normalgröße angezeigt bleibt.

3.5.1.4 FAZIT

Für die Repräsentation der Dialogstruktur durch einen Graphen bieten sich mehrere Varianten an. Der Graph als Darstellungsmittel ermöglicht somit die Abbildung von verschiedenen Eigenschaften der Beziehungen innerhalb der Dialogstruktur sowie die Abbildung von verschiedenen Navigationsstrukturen. Orthogonale Abbildungen bieten sich bei komplexeren, unübersichtlichen Strukturen an, geradlinige Strukturen erscheinen dahingegen ästhetischer. Die Verwendung von abstrakteren Darstellungen (z.B. Schachtelung) soll vernachlässigt werden, um den Benutzer nicht zu verwirren.

3.5.2 SITEMAPS

Eine übliche Darstellung von Navigationsstrukturen innerhalb einer Webanwendung ist die Sitemap, die eine Übersicht für den Benutzer über die Struktur des Inhalts bietet. Dabei ist zwischen einer Sitemap zu unterscheiden, die für die Konzeption einer Webanwendung oder für die Orientierung innerhalb einer fertigen Webanwendung dient.

Es existieren vier übliche Darstellungsprinzipien für eine Sitemap [Jac]:

- Liste
- Dynamische Darstellung
- Kreis
- Metapher

Die Liste ist die am meisten verbreitetste Darstellung. In einer Baumansicht wird die Struktur der Webanwendung abgebildet. Die Liste kann dynamisch erweitert werden, indem durch Interaktion Unterbereiche aufklappen können. Mit der dynamischen Darstellung kann man eine „dreidimensionale Repräsentation“ erreichen, d.h. Verlinkungen innerhalb einer Seite werden ebenfalls abgebildet. Kreise als Anordnungsmuster bieten sich vor allem für kleinere Webanwendungen an. Mit dieser Darstellung ist eine Gewichtung der Themen abbildbar: je näher am Zentrum, desto wichtiger. Bei einer Darstellung durch Metaphern muss jedem Inhaltsobjekt eine Grafik zugeordnet werden, beispielsweise können in einer Bibliotheks-Anwendung die Inhalte als Bücher in einem Regal abgebildet werden. Metaphern als Darstellungsmittel kommen jedoch so gut wie gar nicht mehr zum Einsatz, da es unter

Umständen sehr schwierig sein kann, für alle Inhaltsobjekte eine geeignete Metapher zu finden. Zudem wirkt diese Darstellung meist kindlich und bemüht [Jac].

Für eine Sitemap in der Konzeption wird üblicherweise eine Darstellungsweise ähnlich zu Flussdiagrammen gewählt [Jac03]. Eine Vereinheitlichung wurde vom UI-Designer Jesse James Garrett [Gar02], [Gar03] vorgenommen. Laut seiner Darstellungsweise werden die Seiten einer Webanwendung als Knoten bzw. Kästchen und die Verbindungen als Linien visualisiert. Bei der Gestaltung einer Sitemap ist darauf zu achten, Farbe nur sparsam einzusetzen und die Sitemap insgesamt übersichtlich zu halten. Eine große Sitemap kann man der Überschaubarkeit wegen in mehrere Graphen unterteilen. Nicht alle Verbindungen müssen eingezeichnet werden. Beispielsweise kann man voraussetzen, dass immer eine Verbindung zur Startseite besteht; diese muss in der Abbildung nicht extra dargestellt werden. Auch technische Prozesse wie z.B. das Anmelden eines Benutzers müssen im Navigationsbaum nicht festgehalten sein. Als oberste Priorität gilt es, die Struktur übersichtlich zu halten. Eine Sitemap kann mit regulären Programmen wie Microsoft PowerPoint⁷ oder dem Zeichenprogramm Adobe Illustrator⁸ erstellt werden. Es existieren jedoch auch spezielle Diagramm-Software für die Generierung von Sitemaps.

Fazit

In der Konzeption wird die Darstellung durch einen Flussgraphen – eine Graphenvariante – gewählt. Im Gegensatz zu einer Metapher-Darstellung ist dieser wesentlich unkomplizierter zu erstellen. Die Repräsentation durch Listen kann nicht alle Beziehungen innerhalb einer frei konfigurierbaren Navigation visualisieren wie z.B. die Verbindung von zwei Seiten auf dieselbe Zielseite.

3.5.3 VISUALISIERUNG VON BEZIEHUNGEN

In diesem Abschnitt wird untersucht, welche alternativen Darstellungsmöglichkeiten sich für die Dialogstruktur anbieten. Die Darstellung soll aus einem anderen Blickwinkel betrachtet werden und unter Einsatz von Metaphern erfolgen. Beispielsweise kann ein Programmiersystem vereinfacht als Firma gesehen werden, wobei die Fabrik das resultierende Programm ist und der Anwender die Fabrik baut [KMA]. Die Fabrik verfügt über Maschinen, die koordiniert werden müssen, um systematisch Produkte zu erhalten, zu manipulieren und zu erstellen. Die Maschinen repräsentieren folglich Programmsequenzen, das Produkt ist die Ausgabe. Der Anwender kann für die Konstruktion der Fabrik mehrere Werkzeuge benutzen, welche die Entwicklungsumgebungen repräsentieren. Analog dazu werden verschiedene Metaphern, die innerhalb Programmiersystemen eingesetzt werden, erläutert. Die Analyse erstreckt sich dabei auf das Gebiet der endanwenderorientierten Entwicklung.

⁷ <http://office.microsoft.com/en-us/powerpoint/>

⁸ <http://www.adobe.com/products/illustrator/>

3.5.3.1 VISUELLE PROGRAMMIERWERKZEUGE

Visuelle Programmierung ermöglicht Softwareentwicklung auf Basis von grafischen Codesegmenten anstatt von textuellen Notationen [Sch]. Dieses Prinzip ist weitaus intuitiver als textuelle Programmiersprachen und kann meist von Endanwendern genutzt werden, so dass auch von End User Development (EUD) gesprochen werden kann. Es soll untersucht werden, wie verschiedene visuelle Programmierwerkzeuge Beziehungen bzw. Verknüpfungen der grafischen Codesegmente realisieren.

In grafischen Programmierwerkzeugen variieren die Darstellungen für Codesegmente und Verbindungen, die Abbildungen sind jedoch weitestgehend auf ein Flussdiagramm zurückzuführen. Beispielsweise lassen sich in der grafischen Programmierumgebung **LabViews**⁹ Funktionsblöcke per Drag & Drop miteinander verbinden, so dass eine Art Flussdiagramm entsteht (Abbildung 3.11).

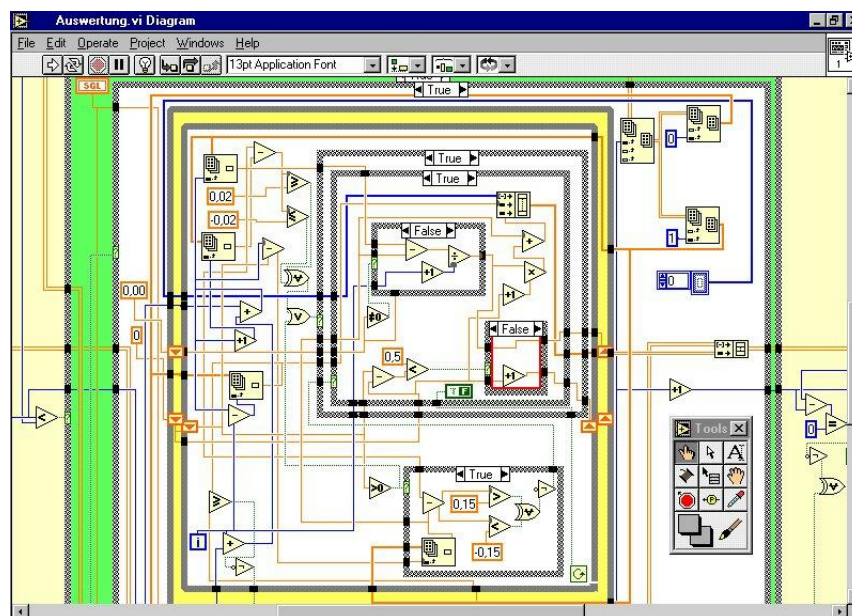


Abbildung 3.11.: Labviews: Programm aus Funktionsblöcken¹⁰

Für die an Kinder gerichtete Anwendungsentwicklung bietet Agentsheets¹¹, ein Entwicklungswerkzeug zur Programmierung von Spielen und interaktiven Demonstrationen, verschiedene Metaphern für die Codesegmente an. Beispielsweise kann ein Netzwerk durch die Kombination von Rohrleitungen, die für die Codebausteine stehen, visualisiert werden. Es gibt verschiedene Arten von Leitungen, die sich in ihrer Form und Ausrichtung

⁹ <http://www.ni.com/labview/d/>

¹⁰ <http://www.ntecs.de/old-hp/uu9r/lang/html/labview.de.html>

¹¹ <http://www.agentsheets.com/>

(uni/bidirektional) sowie Anzahl der Anschlusspunkte unterscheiden. Damit können gerichtete Verbindungen geschaffen werden. Eine zusätzliche Unterstützung bei der Entwicklung bietet die grafische Entwicklungsumgebung **Scratch**¹²: die Codesegmente werden durch verschiedene Bausteine bzw. Puzzleteile repräsentiert und unterscheiden sich je nach Art des Kommandos in ihren Anschlussformen. Durch das Zusammenstecken der Bausteine kann der Benutzer ein Programm erstellen, jedoch lassen sich die grafischen Codesegmente nicht beliebig kombinieren (Abbildung 3.12). Dadurch können automatisch Fehler unterbunden werden.

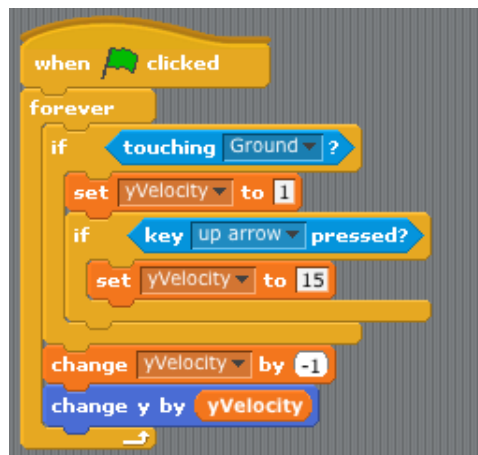


Abbildung 3.12.: Scratch: Programm aus Puzzleteilen¹³

Fazit

In einer Flussdiagramm-ähnlichen Darstellung werden Beziehungen meist durch Linien repräsentiert, in einer abstrakteren Abbildung werden Verbindungen durch die sichtbare Kombination von grafischen Codesegmenten visualisiert. Den Verbindungen können weitere Eigenschaften hinzugefügt werden wie z.B. ein gerichteter Pfeil oder die eingeschränkte Kombination von Bausteinen durch nur bedingt verknüpfbare Anschlüsse.

¹² <http://scratch.mit.edu/>

¹³ http://commons.wikimedia.org/wiki/File:Scratch_Screenshot,_Gravity_Script.png

3.5.3.2 AUTORENWERKZEUGE

Mit Autorenwerkzeugen können interaktive multimediale Präsentationsanwendungen wie z.B. multimediale Lernumgebungen erstellt werden. Innerhalb dieses Abschnittes wird untersucht, wie Verbindungen bzw. Kontrollflüsse dargestellt und wie eine Interaktion ausgelöst werden kann [Bol].

Für die Darstellung von Kontrollflüssen kommen Metaphern wie Buch, Stapel, Film etc. zum Einsatz. Anhand der Darstellung der zeitlichen Beziehungen bzw. der Kontrollflüsse lassen sich die Autorenwerkzeuge in Bildschirm-, Zeitachsen- und Flussdiagramm-basierte kategorisieren:

Bildschirm-basierte Autorenwerkzeuge präsentieren die Medienobjekte innerhalb von Flächen wie z.B. Karten, Seiten oder Dias. Im Prinzip stellen die Flächen einen Bildschirm dar, auf dem der Benutzer die Anwendung betrachtet. Um eine Anwendung zu erstellen, modelliert der Benutzer eine Menge dieser Flächen und definiert deren Präsentationsablauf. Durch Nutzerinteraktion kann während der Präsentation der Wechsel einer Karte bzw. Seite ausgelöst werden. Beispiele sind Hypercard¹⁴ von Apple Inc., welches mit der Metapher Karte und Stapel arbeitet, wobei der Stapel eine Anwendung repräsentiert (Abbildung 3.13).

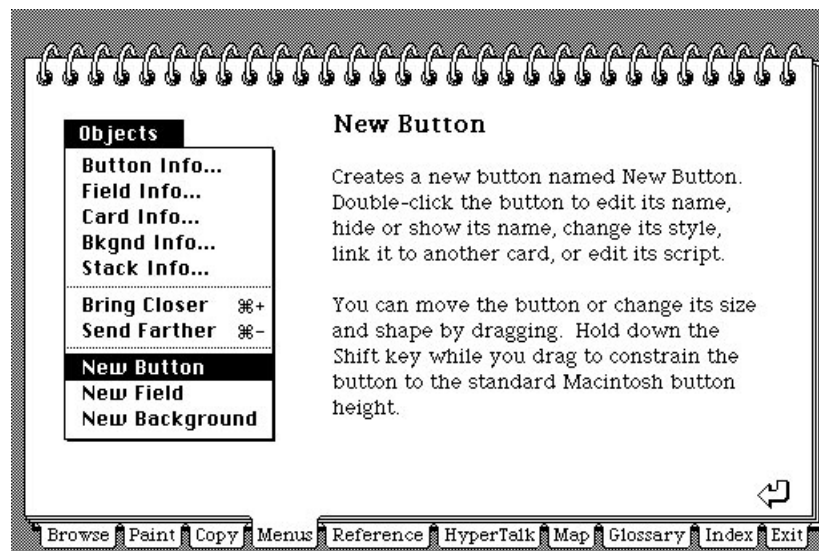


Abbildung 3.13.: Hypercard: Verwendung der Metapher „Stapel“¹⁵

Das Gegenstück von Microsoft ist Toolbook¹⁶, welches mit der Metapher Buch arbeitet. Auf einer Karte bzw. Buchseite können Navigationselemente platziert werden, in einem

¹⁴ <http://www.hypercard.de/>

¹⁵ http://www.appleinsider.com/articles/07/10/18/road_to_mac_os_x_leopard_safari_3_0.html

¹⁶ <http://www.sumtotalsystems.com/products/toolbook-elearning-content.html>

Dialog kann der Autor festlegen, welche Aktion ausgelöst wird. ShareME [Vää92] stellt unterschiedliche Metaphern zur Verfügung, um die Präsentationsstruktur zu gliedern. Neben den bekannten Metaphern Karten, Stapel oder Buch wird die Haus-Raum-Metapher angeboten: ein Haus besteht aus Räumen, in denen die Dokumente untergebracht sind, die Zimmertüren bzw. Wände stellen die Verbindungen dar.

Um den zeitlichen Verlauf einer Präsentation festzulegen, ordnet der Benutzer in **Zeitachsen-basierten** Autorenwerkzeugen wie z.B. Adobe Flash ¹⁷ die Medienobjekte auf einer Zeitachse an. Durch Navigationsinteraktionen werden Zeitsprünge ausgelöst (Abbildung 3.14).

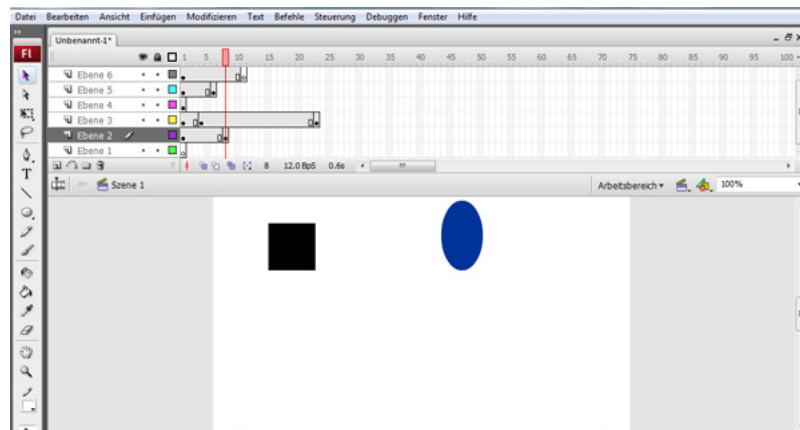


Abbildung 3.14.: Adobe Flash: Verwendung der Metapher „Film“

In MODE [BHL91] werden die Medienobjekte auf verschiedenen Abstraktionsebenen betrachtet, d.h. jeder Aspekt wird in ein eigenständiges Objekt gekapselt. Beispielsweise gibt es Informationsobjekte für die zu präsentierenden Daten, zur Handhabung von Zeitspannen Timer-Objekte, Interaktionsobjekte werden für die Kommunikation mit dem Benutzer eingesetzt.

Flussdiagramm-basierte Autorenwerkzeuge präsentieren die Medienobjekte als Ikonen oder Miniaturen innerhalb eines Diagramms. Die Kanten, welche die Medienobjekte verbinden, geben einen möglichen Verlauf der Präsentation vor. Der tatsächlich durchlaufende Pfad wird von der Navigationsinteraktion durch den Benutzer bestimmt. Flussdiagramm-basierte Autorenwerkzeuge kann man in strukturierte und unstrukturierte unterteilen. Bei strukturierten Flussdiagramm-basierten Autorenwerkzeugen lassen sich logisch bzw. funktional zusammenhängende Teile zusammenfassen und in ein externes Diagramm auslagern, so dass eine gewisse (hierarchische) Strukturierung entsteht. Jedes externe Diagramm wird durch ein spezielles Icon in seinem „Vater-Diagramm“ repräsentiert. Das bekannteste Flussdiagramm-basierte Autorenwerkzeug ist Adobe Authorware ¹⁸.

¹⁷ <http://www.adobe.com/de/products/flash/>

¹⁸ <http://www.adobe.com/de/products/authorware/>

Fazit Insgesamt wird die flexible Kombinierbarkeit voneinander unabhängiger Informationseinheiten, d.h. die Definition von Beziehungen, vernachlässigt. Insbesondere bei Bildschirm-basierten Autorenwerkzeugen können die Interaktionsbeziehungen nur eingeschränkt modelliert werden: ein Seiten- bzw. Stapelwechsel ist lediglich über das Interaktionselement „Button“ auslösbar. Am übersichtlichsten lassen sich die zeitlichen Beziehungen innerhalb einer Zeitachse darstellen. Jedoch gestaltet es sich schwierig, weitere Beziehungstypen innerhalb der Zeitachse abzubilden. Außerdem fehlt die Flexibilität für die Integration von Medienobjekten, deren Lebenszeit unbestimmt ist wie z.B. Live Videos.

Diese Nachteile werden bei einer Flussdiagramm-Darstellung ausgeglichen. Zwar wird nur der grobe Ablauf der Präsentation abgebildet, jedoch lassen sich die Medienobjekte unabhängig von ihrer Lebenszeit problemlos integrieren. Leider sind die verschiedenen Beziehungstypen nur eingeschränkt darstellbar. Außer den Kontrollflüssen, durch Kanten repräsentiert, lassen sich keine weiteren Beziehungen definieren bzw. visualisieren. Hier würde sich eine Erweiterung des Flussdiagramms zu einem allgemeinen Beziehungsgraphen anbieten, in dem alle möglichen Beziehungstypen abgebildet werden.

3.6 ZUSAMMENFASSUNG

Es wurde eine Grundlage geschaffen, ein ansprechendes Design für mobile und Webanwendungen basierend auf Layoutempfehlungen zu entwerfen sowie durch die Präsentation der gängigsten Navigationsstrukturen eine möglichst natürliche Interaktion der zu modellierenden Anwendung zu entwickeln. Der Betrachtung von Programmen, die nach dem WYSIWYG-Prinzip arbeiten, können Darstellungsprinzipien entnommen werden. Bezüglich einer geeigneten Repräsentation der Dialogstruktur wurde die Untersuchung auf verschiedene Bereiche ausgeweitet, um die Problemstellung aus verschiedenen Sichtweisen zu beleuchten. Für eine mögliche Visualisierung durch einen Graphen wurden potenzielle Darstellungen unterschieden und Kriterien bzw. Eigenschaften aufgeführt. Anstatt die Dialogstruktur als Graphen abzubilden, kann sie gemäß der Darstellungsweisen für die Visualisierung von Beziehungen ebenfalls abstrahiert werden. Nun muss spezifiziert werden, welchen Anforderungen die Darstellung und Erstellung der Dialogstruktur genügen muss, um ein geeignetes Konzept zu finden.

4 ANFORDERUNGSANALYSE

Innerhalb dieses Kapitels werden die identifizierten Anforderungen bezüglich der benutzerfreundlichen Modellierung von Layout und Dialogstruktur seitenbasierter Anwendungen formuliert. Es konnten drei Hauptanforderungen spezifiziert werden, die verschiedene Aspekte einer seitenbasierten Anwendung behandeln:

- Aufbau des Tools (Unterkapitel 4.1)
- Darstellung der Frontends und Seiten (Unterkapitel 4.2)
- Darstellung und Umgang mit der Dialogstruktur (Unterkapitel 4.3)

Die Analyse geht vom ServFace Builder aus, der laut Evaluation und allgemeinen Betrachtungen aus Sicht der Benutzerfreundlichkeit einige Optimierungen bzw. Erweiterungen benötigt. Im Wesentlichen geht es darum, das Verständnis der definierten Zielgruppe für die selbstständige Erstellung einer seitenbasierten, aus Webservices zusammengesetzten Anwendung zu fördern. Die Modellierung soll benutzerfreundlich gestaltet und intuitiv zu bedienen sein. Der ServFace Builder soll sowohl eine mobile als auch eine Webanwendung unterstützen und drei verschiedene Navigationsstrukturen (freie, sequentielle und hierarchische) anbieten.

Zunächst ist zu untersuchen, wie die Funktionalität angemessen präsentiert werden kann (Abschnitt 4.1). Dafür sollen alle notwendigen Funktionen definiert und anhand des aktuellen Standes des ServFace Builders die Schwachstellen bezüglich des Aufbaus aufgezeigt und daraus folgend Anforderungen für Optimierungen spezifiziert werden.

Anschließend soll eine möglichst traditionelle Darstellung der Anwendung entwickelt werden, die dem präsentationsorientierten Ansatz gemäß der Endanwendung entspricht (Unterkapitel 4.2). Zudem soll der Benutzer das Layout der Anwendung in geringem Maße anpassen können (Abschnitt 4.2.3).

Insbesondere die Darstellung der Dialogstruktur wird in dieser Arbeit umfassend ergründet (Abschnitt 4.3), die Modellierung der Navigationsstruktur soll dem Nutzer möglichst leicht gemacht werden. Dementsprechend soll die Darstellung intuitiv gestaltet sein und wenn möglich eine Unterstützung innerhalb des Modellierungsprozesses angeboten werden.

Die Ermittlung der Anforderungen erfolgt in folgender Weise: Zunächst wird der aktuelle Stand des ServFace Builders bzgl. des aktuellen Aspektes kurz vorgestellt, anschließend werden die Probleme bzw. Erweiterungsmöglichkeiten geschildert. Der Pfeil „→“ leitet die Schlussfolgerung ein bzw. die konkrete Formulierung der Anforderung.

4.1 AUFBAU DES TOOLS

In diesem Abschnitt soll untersucht werden, welche Anforderungen an den Aufbau eines Werkzeugs zur Erstellung einer servicebasierten Anwendung wie den ServFace Builder bestehen. Dafür sind zunächst alle Funktionen zu definieren, die das Werkzeug leisten muss, anschließend wird auf die Anforderungen an die Präsentation der Funktionen eingegangen.

4.1.1 HIERARCHISCHE AUFGABENANALYSE DER ENDANWENDERORIENTIERTEN SERVICE KOMPOSITION

Zunächst sollen die fundamentalen Aufgaben identifiziert werden, die für die Erstellung von Anwendungen aus Service Kompositionen zu bewältigen sind:

- Konfiguration der Anwendung (Auswahl der Zielplattform, Navigationsmodus)
- Verwaltung der Seiten
- Verwaltung der Transitionen
- Verwaltung der Service Operationen
- Verwaltung der Datenflüsse
- Konfiguration des Layouts

Die meisten Aufgaben kann die aktuelle Version des ServFace Builders bereits gewährleisten. Die Funktionalität muss noch um die Aufgabenbereiche „Konfiguration der Anwendung“ und „Konfiguration des Layouts“ erweitert werden. Für eine genauere Beschreibung der Aufgabenbereiche wird die Methode „Hierarchical Task Analysis“ (HTA) eingesetzt, bei der ein abstraktes Aufgabenmodell erzeugt wird (Abbildung 4.1). Bei dieser Methode werden die Aufgaben schrittweise in Unteraufgaben zerlegt, bis nur noch die atomaren Einzelaufgaben vorhanden sind [KBS05]. Jede Ebene beinhaltet einen feineren Detaillierungsgrad.

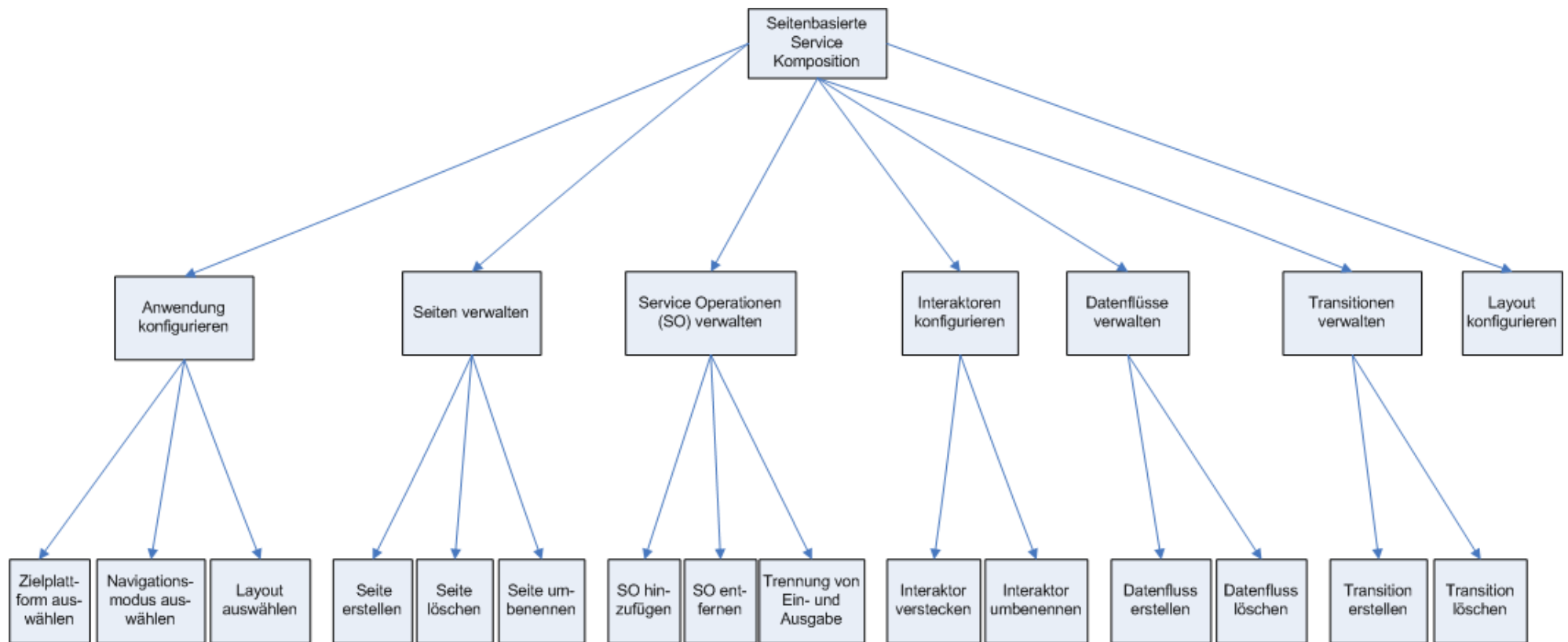


Abbildung 4.1.: Aufgabenanalyse mit HTA

4.1.2 PRÄSENTATION DER FUNKTIONALITÄT

Standardansicht und initialer Zustand ist derzeit die Prozessansicht, in der die Dialogstruktur erstellt und visualisiert wird. Angezeigt werden eine Übersicht über alle vorhandenen Seiten in Thumbnailgröße und die Dialogflüsse. Nach Belieben kann eine Seite vergrößert werden, um in die Seitenansicht zur Konfiguration der Seite zu wechseln. Laut der Evaluation war dem Benutzer der Anfangsprozess unklar, d.h. er wusste nicht, wie nach Auswahl der Zielplattform zu verfahren ist. Zu Beginn wäre der erste Schritt die Erstellung von einer oder mehreren Seiten und die Befüllung mit Service Operationen. Anschließend können Transitionen und Datenflüsse modelliert werden.

→ Die Vorgehensweise bei der Modellierung soll durch den Aufbau der Anwendung vorgegeben werden [A1].

Um einen Interpage-Datenfluss zu erstellen, muss der Benutzer nacheinander die betroffenen Seiten öffnen und die gewünschten Felder kennzeichnen.

→ Die Durchführung von seitenübergreifenden Aktionen wie die Erstellung eines Interpage-Datenflusses oder Verschiebung der Ausgabe soll angemessen unterstützt werden [A2].

4.2 FRONTEND- UND SEITENDARSTELLUNG

In der bisherigen Version des ServFace Builders werden die Frontends der Service Operationen als voneinander unabhängige Widgets visualisiert, die sich innerhalb der Seite per Drag & Drop frei positionieren lassen (Abbildung 4.2).

Die Entwurfsansicht soll dem präsentationsorientierten Ansatz zufolge möglichst der Endanwendung nahe kommen, so dass durch die Modellierungsdarstellung das Verständnis für das Endprodukt gefördert wird. Laut Evaluation konnten die Benutzer sich jedoch nicht vorstellen, wie die Frontends in der Anwendung umgesetzt werden sollen und wie sie zu bedienen sind.

4.2.1 PLATTFORMABHÄNGIGE SEITENDARSTELLUNG

Unterstützt wird bis jetzt die Erstellung einer Webanwendung, die Modellierungsdarstellung ist jedoch keiner plattformtypischen Anwendung zuzuordnen und erscheint demzufolge eher ungewohnt. Der Benutzer erkennt gemäß der Evaluation nicht den Zusammenhang zwischen der Seite, die er zur Entwurfsansicht selbst mit Service Operationen füllen kann und einer Webseite, die er innerhalb eines Browsers ganz selbstverständlich benutzt.

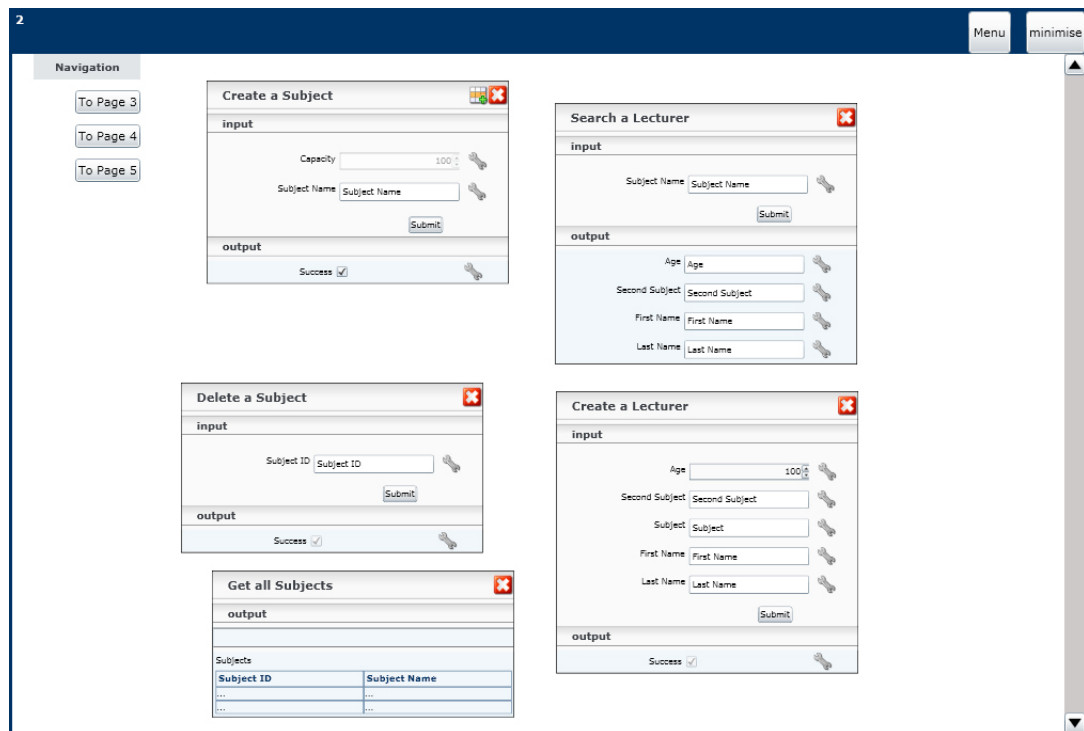


Abbildung 4.2.: Derzeitige Visualisierung der Service Operationen als Widgets (Screenshot)

Zukünftig soll der ServFace Builder unterschiedliche Plattformen unterstützen, was sich auch in der Modellierungsansicht widerspiegeln soll. Innerhalb dieser Arbeit soll ein Konzept für die Unterstützung einer normalen Webanwendung wie einer mobilen Applikation entworfen werden.

In Tabelle 4.1 sind plattformspezifische Eigenschaften aufgelistet, die Einfluss auf die Darstellung im ServFace Builder haben.

→ Anhand dieser Eigenschaften sollen unterschiedliche Layouts für alle Navigationsstrukturen entwickelt werden, die an schon bekannte Anwendungen der jeweiligen Plattform erinnern [A3].

Dem Benutzer wird durch die frei positionierbaren Frontends bei dem Layout völlig freie Hand gelassen, bei einem Anwender ohne Hintergrundwissen in Sachen Layout kann dieses Prinzip zu einem ungünstigem Design führen (Abbildung 4.2).

→ Die Integration der Frontends soll ebenfalls gemäß plattformspezifischer Konventionen überarbeitet werden [A4].

KRITERIUM	Web-Anwendung	Mobile Anwendung
<i>Bildschirm-auflösung</i>	1024 x 768 px	480 x 320 px
<i>Navigation</i>	Interaktionselemente am linken bzw. oberen Seitenrand	Interaktionselemente in der untersten oder obersten Zeile
<i>Platzierung der Labels</i>	Label und dazugehöriger Interaktor nebeneinander	Label und dazugehöriger Interaktor untereinander
<i>Platzierung der UI-Elemente</i>	Untereinander in festem Bereich	Untereinander, gesamte Breite des Displays
<i>Ausführungsumgebung</i>	Browser	Eigenständige Anwendung
<i>Mehrspaltige Seiten/Frontends</i>	ja	nein
<i>Scrolling</i>	ja	In eine Richtung begrenzt

Tabelle 4.1.: Plattformspezifika

4.2.2 PRÄSENTATIONSORIENTIERTE DARSTELLUNG

Unter anderem fiel es den Benutzern innerhalb der Evaluation schwer, zwischen Entwurfs- und Laufzeit zu unterscheiden. Somit wollten manche Benutzer bereits in der Entwurfsansicht die Services aufrufen. Dies ist unter anderem auf das angewandte WYSIWYG-Prinzip zurückzuführen. Die grafische Darstellung der Services unterscheidet sich in der Entwurfsansicht im Wesentlichen nicht von Endanwendung. Alle UI-Elemente eines Services sind in der Entwurfsansicht bereits abgebildet.

→ Dem Benutzer soll deutlich gemacht werden, dass er sich innerhalb des ServFace Builders noch in der Entwurfsansicht befindet und die Services noch nicht nutzbar sind [A5]. Dazu sollen bereits bewährte Prinzipien aus WYSIWYG-Editoren aufgenommen werden.

Andererseits findet eine Vermischung des WYSIWYG-Prinzips – die endanwendungsorientierte Darstellung – und Konfigurationswerkzeugen in der Darstellung der Frontends statt. So wird zu jedem UI-Element ein Werkzeug-Icon angezeigt, mit Hilfe dessen man z.B. den Namen des Labels verändern oder das gesamte UI-Element verstecken kann. Weiterhin sind Elemente zum Löschen des gesamten Frontends oder zum Anzeigen versteckter Elemente vorhanden (Abbildung 4.3).

→ Es soll eine klare Trennung zwischen UI-Elementen, die auch in der Endanwendung sichtbar sind, und Konfigurationswerkzeugen, die nur in der Entwurfsansicht angezeigt werden, erkennbar sein [A6].

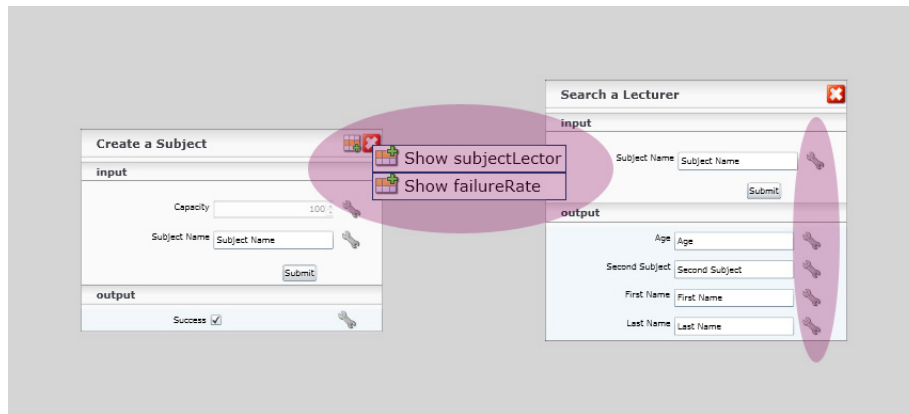


Abbildung 4.3.: Vermischung der präsentationsorientierten Darstellung und Konfigurationswerkzeugen

4.2.3 LAYOUTMANAGER

Der Anwender soll zukünftig das Layout der Anwendung in der Design-Phase in eingeschränktem Umfang konfigurieren können. Durch einen „Layoutmanager“ soll es ermöglicht werden, ein Corporate Design zu integrieren. Dabei muss berücksichtigt werden, dass der typische Anwender aus der Zielgruppe nicht unbedingt über Erfahrung in Design verfügt und zu viele Freiheiten in der Gestaltung zu einem schlechtem Layout führen können.

Das Frontend einer Service Operation wird in der Entwurfsansicht automatisch generiert. Für jede Plattform soll es eine einheitliche Darstellung der Service Operation geben, auf die der Benutzer generell keinen Einfluss hat. Um eine passende Anordnung der Frontends in Abhängigkeit vom individuellen Inhalt der Seite zu finden, soll der Benutzer geringfügig Einfluss auf die Gestaltung der Frontends nehmen können.

→ Einerseits soll mit Hilfe einer **Masterpage** in eingeschränktem Umfang das generelle Layout aller Seiten definiert, andererseits individuelle Anpassungen von Frontends und UI-Elementen vorgenommen werden können [A7].

4.3 DIALOGSTRUKTUR

In diesem Abschnitt werden verschiedenen Aspekte der Seitenübergänge betrachtet. Einerseits soll die Erstellung und Darstellung der Dialogstruktur intuitiver gestaltet, andererseits soll – sofern möglich – eine Unterstützung vom System bei der Modellierung von Seitenübergängen und Datenflüssen geboten werden. Zudem sollen verschiedene Navigationsstrukturen analysiert und als Templates zur Verfügung stehen.

4.3.1 ERSTELLUNG UND DARSTELLUNG

In der aktuellen Version wird die Dialogstruktur in der Prozessansicht erstellt und visualisiert. Die Prozessansicht bietet eine Übersicht über alle bestehenden Seiten in Thumbnailgröße mitsamt den integrierten Service Operationen und Navigationselementen sowie über die Transitionen (Abbildung 4.4).

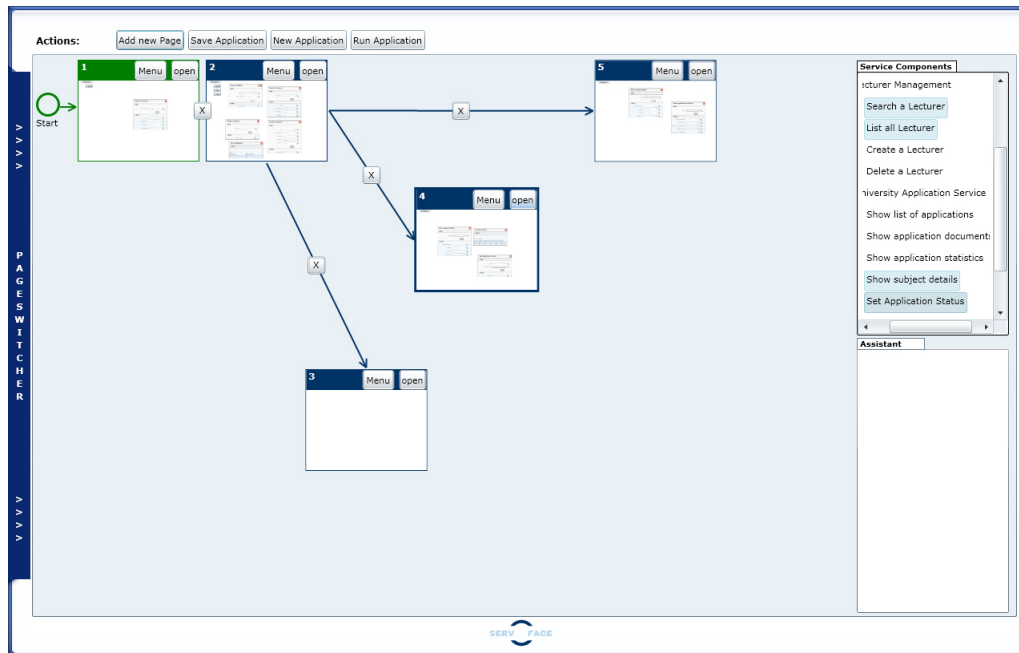


Abbildung 4.4.: Prozessansicht als initiale Ansicht (Screenshot)

In der Evaluation haben die Testpersonen in der Regel sequentielle Anwendungen gebaut. Komplexere Transitionsstrukturen mit mehrdimensionalen Verzweigungen wurden nicht erstellt. Dies ist auf das Vorwissen der Testpersonen zurückzuführen. Einem Benutzer, der beispielsweise schon selber Webseiten erstellt und HTML-Editoren benutzt hat, ist die Darstellungsweise und der Umgang mit komplexen Dialogfluss-Strukturen geläufig. Für Benutzer mit weniger Erfahrungshintergrund hingegen erscheint diese Graphen-Darstellung ungewohnt und die Erstellung nicht unbedingt intuitiv. Anstatt den Benutzer sofort mit der komplexen Prozessansicht zu konfrontieren, soll er zunächst langsam an die Transitionen herangeführt werden.

Es sollen verschiedene Darstellungsmöglichkeiten ausgearbeitet werden, dazu sollen bewährte Methoden, die in der Analyse (Abschnitt 3.5) vorgestellt wurden, als Grundlage dienen. Verschiedene Aspekte der Darstellung von Seiten und Beziehungen sind dabei zu berücksichtigen, um zu entscheiden, welche Informationen in der Übersicht enthalten sein sollen. Durch den begrenzt zur Verfügung stehenden Platz und die beliebige Größe der Dialogstruktur

bedingt soll die Darstellung auf das Wesentliche reduziert werden und gleichzeitig alle für diese Ansicht notwendigen Informationen liefern. Die Erstellung der Dialogstruktur soll möglichst intuitiv gestaltet sein und dem Benutzer alle Freiheiten lassen. Dazu gehört die Visualisierung der Richtung der Transition sowie ob diese uni- oder bidirektional ist. Weitere Eigenschaften einer Transition können in die Visualisierung miteinbezogen werden wie z.B. ob sie in der Endanwendung automatisch ausgelöst wird oder ob sie an eine Bedingung geknüpft ist. Im Folgenden werden alle Kriterien bzw. Aspekte aufgelistet, die in die Darstellung miteinfließen können:

- Übersicht über alle Seiten und Transitionen
- Gesamtkontext deutlich
- Seitendarstellung: Inhalt bzw. Kontext einer Seite sichtbar
- Flexible Erstellung der Dialogstruktur ohne Einschränkung
- Übersichtliche Darstellung komplexer Transitionen
- Eigenschaften einer Transition: (un)gerichtet, uni-/bidirektional, automatische Auslösung, Auslösung bedingt durch ein Ereignis
- Datenflüsse

→ Die Darstellung und Erstellung der Dialogstruktur soll die Bandbreite des unterschiedlichen Hintergrundwissens der potentiellen Nutzer berücksichtigen [A8].

→ Die Übersicht der Dialogstruktur soll verschiedene Aspekte berücksichtigen [A9].

4.3.2 UNTERSTÜTZUNG VERSCHIEDENER NAVIGATIONSSTRUKTUREN

Derzeit kann der Benutzer die Navigation frei modellieren und beliebige Strukturen entwickeln. Ziel ist jedoch, den Benutzer bei der Modellierung zu unterstützen, um eine möglichst benutzerfreundliche Endanwendung zu erhalten. In Hinsicht auf die Navigationsstruktur bedeutet dies zu untersuchen, welche gängigen Konventionen bzw. Navigationsarten für eine sinnvolle Strukturierung der Anwendung existieren und sie in den Modellierungsprozess zu integrieren.

Die gängigsten Strukturen wurden in der Analyse (Abschnitt 3.1) detailliert vorgestellt. Im Folgenden sind nochmals kurz die wichtigsten Eigenschaften aufgelistet:

- **Hierarchische Struktur:** Die Inhalte werden thematisch gruppiert, so dass eine Hierarchie mit Hauptkategorien (Globale Navigation) und Unterthemen (Lokale Navigation) entsteht. Standardmäßig wird die Hierarchie von der Startseite aus in die Tiefe durch-

laufen. Alle Unterthemen, die zu derselben Kategorie gehören, sind genauso wie alle globalen Elemente miteinander verbunden.

- **Sequentielle Struktur:** Die Anwendung wird in vorgegebener linearer Reihenfolge durchlaufen. Meist besteht die Möglichkeit neben der Navigation auf die nachfolgende Seite eine der vorherigen Seiten durch eine Pfadnavigation aufzurufen.
- **Freie Struktur:** Es existiert keine bestimmte Struktur, der Benutzer kann beliebige Transitionen konfigurieren. Dies entspricht dem aktuellen Stand.

Aus diesen Navigationsmustern sollen Templates entwickelt werden, die dem Benutzer zur Verfügung stehen. Anstatt die Dialogstruktur ohne bestimmte Ordnung zu erstellen, ist durch die Auswahl eines Templates die Struktur vorgegeben. Dadurch soll es dem Benutzer erleichtert werden, eine logisch aufgebaute Anwendung zu erstellen. Der Benutzer muss sich Gedanken machen, wie der Inhalt der Anwendung zusammenhängt und wie sie am besten zu durchlaufen ist. Diese Vorgehensweise empfiehlt sich für eine größere Wiederverwendbarkeit und um anderen Benutzern, die die Anwendung nur benutzen und nicht selbst erstellt haben, den Einstieg zu erleichtern.

→ Es sollen Templates für unterschiedliche Navigationsstrukturen zur Auswahl stehen [A10].

Wie im Abschnitt 4.2.1 beschrieben, soll das Layout der Modellierungsansicht möglichst an bereits bekannte, plattformspezifische Anwendungen erinnern. Bei der Anordnung von Navigationselementen müssen demzufolge ebenfalls unter Berücksichtigung der angewandten Navigationsstruktur allgemeine Layoutempfehlungen und Konventionen gemäß der ausgewählten Zielplattform miteinbezogen werden.

→ Die Navigationselemente sollen gemäß der ausgewählten Struktur und Plattform angemessen platziert sein [A11].

Für die vorgegebenen Templates sollen verschiedene Repräsentationen der Dialogstruktur in der Prozessansicht spezifiziert werden. Ziel ist dabei, durch die Positionierung der Elemente eine geordnete und sinnvolle Struktur zu erhalten sowie mögliche Zusammenhänge zwischen Seiten zu verdeutlichen.

→ Je nach ausgewähltem Template soll sich die Darstellung der Dialogstruktur anpassen [A12].

4.3.3 UNTERSTÜTZUNG BEI DER MODELLIERUNG VON BEZIEHUNGEN

Nachdem der Benutzer verschiedene Frontends kombiniert und Seiten erstellt hat, können Relationen modelliert werden. Es besteht die Möglichkeit, einen Datenfluss (Übertragung von Daten zwischen Frontends) oder eine Transition (Verbindung zwischen zwei Seiten)

zu erstellen. Dabei kann eine Beziehung aus unterschiedlichen Beweggründen modelliert werden.

Zunächst kann der Benutzer beliebig Datenflüsse oder Transitionen aufgrund von inhaltlichen Zusammenhängen manuell erstellen. Aus diesen ersten Relationen können nun Abhängigkeiten resultieren. Beispielsweise erfordert ein Datenfluss, der zwischen unterschiedlichen Seiten stattfindet, eine indirekte Transition zwischen den betroffenen Seiten. Weiterhin ist aus Sicht der Benutzerfreundlichkeit die Modellierung gewisser Transitionen notwendig. Insbesondere sollte die „Erreichbarkeit aller Zustände“, ein Kriterium der Graphentheorie, eingehalten werden.

Da ein typischer Anwender aus der definierten Zielgruppe weder über Fachkenntnisse aus dem Bereich der Human-Computer-Interaction (HCI) verfügt noch ein professioneller UI-Designer ist, soll der ServFace Builder den Benutzer in der Entwurfsansicht bei der Modellierung von Datenflüssen und Transitionen unterstützen. Zunächst sollen verschiedene HCI-Aspekte betrachtet werden. Anschließend werden die Abhängigkeiten zwischen Datenflüssen und Transitionen erörtert.

4.3.3.1 ASPEKTE DER HUMAN-COMPUTER-INTERACTION

Die Interaktion mit der Anwendung soll benutzerfreundlich gestaltet sein. Die Navigation einer Webanwendung kann auf einen Graphen mit den Seiten als Knoten und Transitionen als Kanten abgebildet werden. Im Hinblick auf diese Navigationsstruktur können verschiedene Usability-Anforderungen identifiziert werden [Thi07].

Orientierung

Laut [Jac05] soll eine Möglichkeit bestehen, einen Schritt zurück zur nächsthöheren Ebene zu machen sowie jederzeit zum Anfang zurückkehren zu können. Somit kann sich der Benutzer orientieren bzw. seinen Standpunkt ermitteln. Die Graphentheorie besagt zusätzlich, dass alle Knoten innerhalb eines Systems erreichbar sein müssen.

→ Es sollen Transitionszyklen zur Orientierung und zum komfortablem Durchlaufen der Anwendung vorhanden sein. Dabei ist zu berücksichtigen, ob und bei welchen Navigationsstrukturen eine Unterstützung bei der Modellierung von Transitionen erwünscht ist [A13].

Datenflusszyklus

Wenn ein Interpage-Datenfluss von Seite A zu Seite B besteht, sollte kein Interpage-Datenfluss von B zu A möglich sein.

→ Die Erstellung einer Datenfluss-Schleife zwischen zwei oder mehreren Seiten sollte durch das System verhindert oder wenigstens auf die Entstehung einer Schleife hingewiesen werden [A14].

Idempotente Services

Es gibt Services, die nicht idempotent sind. Wenn dieser Service aufgerufen wurde, ist die ausgeführte Aktion nicht umkehrbar (wie z.B. eine Bezahlungstransaktion). Wenn jedoch innerhalb der erstellten Anwendung dieser Service mehrmals aufgerufen werden kann, wird dem Benutzer suggeriert, den Aufruf rückgängig machen zu können. Das System soll eine entsprechende Unterstützung bieten.

→ Idempotente Services sollen in der Modellierung von Transitionen berücksichtigt werden [A15].

4.3.3.2 EINFLUSSFAKTOR DATENFLUSS

Die Verbindung von Frontends bringt Abhängigkeiten mit sich und kann Transitionen beeinflussen bzw. die Erstellung einer Transition erfordern. Zusätzlich hat die Modellierung einer Beziehung Auswirkungen auf die darunterliegenden Webservices sowie auf das Verhalten der Endanwendung zur Laufzeit. Um alle Aspekte bei der Modellierung eines Datenflusses bzw. einer Transition zu berücksichtigen, sollen im Folgenden zunächst die Einflussfaktoren spezifiziert werden.

Frontend-Beziehung

Um einen Datenfluss zu erstellen, muss der Benutzer das Quell- und Ziel-Frontend des Datenflusses definieren. Ein Frontend kann über einen Input-Bereich verfügen und besitzt immer ein Output-Bereich. Bei einem Datenfluss muss der Benutzer bestimmen, welche Teile miteinander verbunden werden sollen:

- *Input to Input*: Die befüllten Input-Elemente des Quell-Frontends werden als Eingabedaten für das Ziel-Frontend benutzt. Dadurch kann eine unnötige manuelle Befüllung äquivalenter Daten verhindert werden.
- *Output to Input*: Die Ausgabedaten des Quell-Frontends werden als Eingabe für das Ziel-Frontend verwendet.
- *Input to Output*: Der Output-Bereich eines Frontends kann abgespalten und an einen anderen Ort versetzt werden.

Ausgabetypp

Die Ausgabe eines Webservices kann unterschiedliche Strukturen haben. Nicht nur die Darstellung des Rückgabewertes variiert, sondern auch, wie beim Verbinden der Ausgabe mit einem anderen Element verfahren wird.

- *Single*: Der Rückgabewert ist ein einfacher Datentyp und kann als UI-Element dargestellt werden. Das UI-Element kann als Quell-Element für einen Datenfluss dienen.
- *Liste*: Die Ausgabe ist eine Menge an einfachen Datentypen und kann als Liste von UI-Elementen angezeigt werden. Jedes einzelne UI-Element kann Ausgangselement für einen Datenfluss sein.
- *Tabelle*: Die Ausgabe ist eine Menge an Objekten, die aus einfachen Datentypen bestehen und in einer Tabelle abgebildet werden. Jede Spalte ist als Quelle für einen Datenfluss verwendbar. Der konkrete Wert wird in der Endanwendung von dem Benutzer ausgewählt.
- *Komplexe Tabelle*: Die Ausgabe ist eine Menge an Objekten, die aus komplexen Datentypen besteht und in einer komplexen Tabelle abgebildet wird. Die komplexe Tabelle zeigt die Instanzen der komplexen Datentypen in einfachen, aufgelösten Datentypen an. Jedes UI-Element kann als Quell-Element ausgewählt werden. Der konkrete Wert wird in der Endanwendung vom Benutzer bestimmt.
- *Boolscher Rückgabewert*: An einem boolschen Rückgabewert können verschiedene Bedingungen geknüpft sein. Abhängig von der Erfüllung einer Bedingung wird zur Laufzeit eine Aktion ausgelöst.

Frontend-Verteilung

Die Positionierung der Frontends soll abhängig vom Seitenkonzept des ServFace Builders betrachtet werden:

- *Intrapage*: Die in den Datenfluss involvierten Frontends befinden sich auf derselben Seite.
- *Interpage*: Die vom Datenfluss betroffenen Frontends sind auf unterschiedlichen Seiten platziert.

Frontend-Abhängigkeiten

Es soll unterschieden werden, an wievielen Beziehungen ein Datenfluss bzw. Frontend teilnehmen kann. Dazu sind folgende Kardinalitäten zu unterscheiden:

- *1:1*: Die Daten werden von einem Quell- zu einem Ziel-Frontend übertragen.
- *1:n*: Die Daten eines Quell-Frontends werden zu mehreren Ziel-Frontends übertragen.
- *n:1*: Die Daten mehrerer Quell-Frontends werden zu einem Ziel-Frontend übertragen.

Runtime-Effekt

Die in der Entwurfsansicht definierten Datenflüsse und Transitionen müssen zur Laufzeit interpretiert werden. Durch Interaktion des Benutzers sind verschiedene Runtime-Effekte auslösbar: Initialisierung einer Transition, eines Datenflusses oder eines Service-Aufrufs.

Um einen Runtime-Effekt auslösen zu können, sind je nach Anwendungsfall spezielle Interaktionselemente, die dem Nutzer eine möglichst natürliche Interaktion mit dem System erlauben, erforderlich.

→ Abhängig von den definierten Dimensionen soll spezifiziert werden, wann eine automatische Unterstützung wie z.B. die automatische Erstellung oder Unterbindung einer Transition sinnvoll ist und in welcher Form sie stattfinden soll [A16].

4.4 ZUSAMMENFASSUNG

Innerhalb dieses Kapitels wurden die notwendigen Anforderungen bezüglich benutzerfreundliche Konzepte für eine seitenbasierte Anwendung spezifiziert. Durch die Analyse der zu bewältigenden Aufgaben und Funktionalität können entsprechende Ansichten des Tools entwickelt werden. Die Darstellung der Seite soll plattformabhängig gestaltet werden und ein angemessenes Konzept für die Integration der Frontends bieten. Zudem wurden Anforderungen gestellt, die für eine präsentationsorientierte Darstellung erforderlich sind. Zuletzt soll es möglich sein, die Seite mittels einem Layoutmanager zu konfigurieren. Für die Repräsentation der Dialogstruktur wurden notwendigen Kriterien bzw. Aspekte definiert, die eine angemessene Darstellung erfüllen muss. Die Einführung neuer Navigationsstrukturen hat Auswirkung auf alle Ansichten. Um dem Benutzer bei der Modellierung von Beziehungen eine Unterstützung anzubieten, wurden verschiedene Anforderungen aus Sicht der Benutzerfreundlichkeit sowie mögliche Einflussfaktoren und Dimensionen spezifiziert, die einen Eingriff seitens des Systems erfordern. In der Tabelle 4.2 werden die identifizierten Anforderungen nochmal zusammengefasst.

ANFORDE- RUNGSNR	Beschreibung
/A1/	Modellierung durch Aufbau vorgegeben
/A2/	Unterstützung für seitenübergreifende Aktionen
/A3/	Plattformabhängiges Layout
/A4/	Anpassung Frontend-Integration
/A5/	Entwurfs- vs. Laufzeitansicht
/A6/	Reine präsentationsorientierte Darstellung
/A7/	Layoutmanager: Masterpage sowie individuelle Anpassungen
/A8/	Angemessene Darstellung der Dialogstruktur
/A9/	Berücksichtigung der Kriterien bei der Dialogstruktur
/A10/	Vorgegebene Templates für Navigationsstrukturen
/A11/	Angemessene Anordnung der Navigationselemente
/A12/	Angemessene Darstellung der Dialogstruktur
/A13/	Transitionszyklen für Orientierung und komfortablen Navigation
/A14/	Vermeidung von Datenflusszyklen
/A15/	Berücksichtigung von idempotenten Services
/A16/	Unterstützung bei der Modellierung von Transitionen abhängig von den spezifizierten Dimensionen

Tabelle 4.2.: Identifizierte Anforderungen

5 KONZEPTION

Innerhalb dieses Kapitels werden die Konzepte für die in der Anforderungsanalyse definierten Anforderungen spezifiziert.

Das Unterkapitel 5.1 beinhaltet ein Konzept für den Aufbau des Tools, das zum Ziel hat, die Funktionalität angemessen zu präsentieren. Im Unterkapitel 5.2 werden verschiedene Layouts abhängig von Zielplattform und Navigationsstruktur sowie Möglichkeiten für eine rein präsentationsorientierte Darstellung und die Definition eines Layoutmanagers vorgestellt. Desweiteren werden im Unterkapitel 5.3 verschiedene mögliche Darstellungen für die Dialogstruktur ausführlich diskutiert und bewertet. Zudem wird die Unterstützung bzgl. der Modellierung von Transitionen definiert. Die Einführung verschiedener Navigationsstrukturen hat dabei Einfluss auf alle Bereiche.

Zunächst werden die bereits identifizierten Konflikte und die daraus resultierenden Anforderungen nochmal kurz zusammengefasst sowie der aktuelle Stand knapp erläutert. Die Konzepte basieren meist auf den in der Analyse vorgestellten Konventionen. Einige Konzepte wurden innerhalb einer Benutzerstudie evaluiert. Wenn eine vorhanden ist, ist das Fazit der Evaluierung im Abschluss des jeweiligen Abschnitts als Bewertung des Konzepts integriert.

5.1 AUFBAU DES TOOLS

Der Benutzer soll durch das Tool geleitet werden und alleine durch seinen Aufbau die Vorgehensweise erfassen [A1]. Aus der Evaluation ging hervor, dass dies durch den derzeitigen Aufbau jedoch nicht gefördert wird. Alle Benutzer hatten Einstiegsprobleme und konnten mit der Prozessansicht nicht viel anfangen. In der Anforderungsanalyse wurden im Abschnitt 4.1.1 alle fundamentalen Aufgaben, die ein seitenbasiertes Werkzeug zur Erstellung von servicebasierten Anwendungen wie der ServFace Builder gewährleisten muss, identifiziert. Nun muss ein logischer Aufbau des ServFace Builder definiert werden, um diese Funktionalität angemessen zu repräsentieren.

Die Prozessansicht ist vor allem für fortgeschrittene Nutzer wichtig, die komplexe Anwendungen bauen möchten. Für den Einsteiger ist diese Darstellung zunächst eher nicht notwendig. Deswegen wird die Funktionalität aufgeteilt und unterschiedliche Modi entwickelt. Im initialen Modus soll der Benutzer vor allem Seiten konfigurieren, der zweite Modus soll in erster Linie eine Übersicht über die Struktur der Anwendung bieten. Diese Aufteilung soll dem Benutzer die Vorgehensweise erleichtern und zudem die unterschiedliche Bandbreite an Vorwissen der Benutzer berücksichtigen. Im Folgenden werden alle Funktionen der beiden Modi genau festgelegt:

Die Ansicht des **Simple Modes** orientiert sich gemäß der Zielgruppe, die überwiegend im Büro beschäftigt ist, am Präsentationsprogramm Microsoft PowerPoint¹⁹. Die Hauptfunktionalität besteht in der Erstellung neuer Seiten und deren Befüllung mit Service Operationen. Zudem kann mit Hilfe eines Layoutmanagers ein einheitliches Design vorgenommen werden. Es besteht ebenfalls die Möglichkeit, Transitionen zu erstellen (Abbildung 5.1).

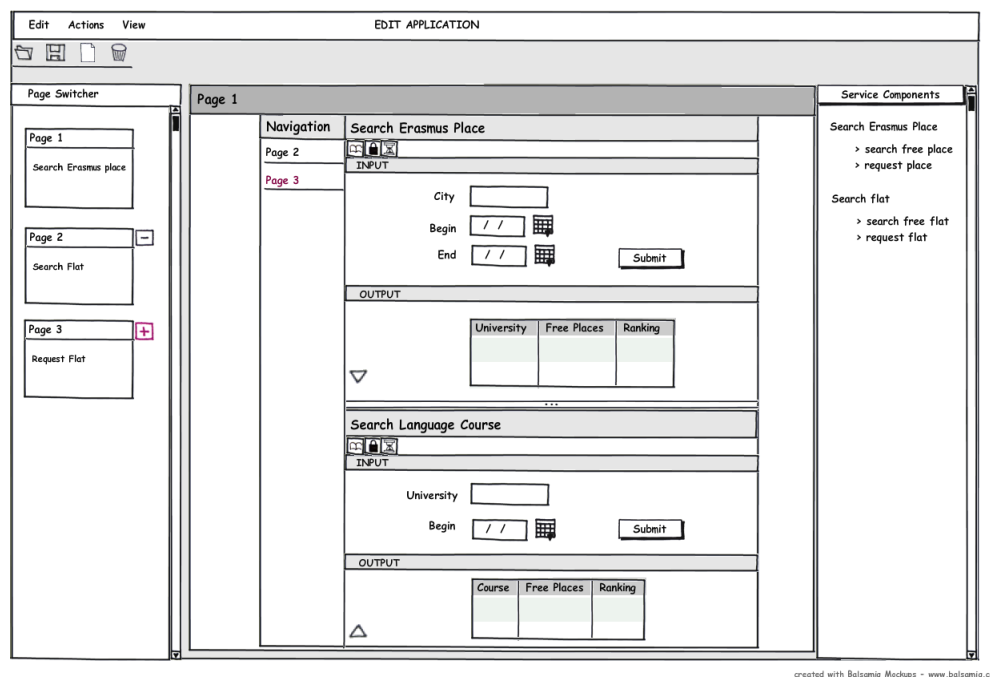


Abbildung 5.1.: Simple Mode

Im linken Randbereich wird der Pageswitcher mit einer Übersicht über alle vorhandenen Seiten dauerhaft angezeigt. Jeder Seite ist ein Button zugeordnet, mit Hilfe dessen man von der aktuell geöffneten Seite aus eine Transition zu der ausgewählten Seite erstellen oder entfernen kann. Optional kann dem Pageswitcher der Layoutmanager hinzugefügt

¹⁹ <http://office.microsoft.com/en-us/powerpoint/>

werden. Im rechten Randbereich befindet sich eine Leiste, die den Service Browser mit einer Auflistung von vorhandenen Services enthält. Desweiteren kann eine Zwei-Seiten-Ansicht geöffnet werden, um einen Interpage-Datenfluss zu erstellen oder den Ausgabe-Bereich eines Frontends auf eine andere Seite zu verschieben (Abbildung 5.2).

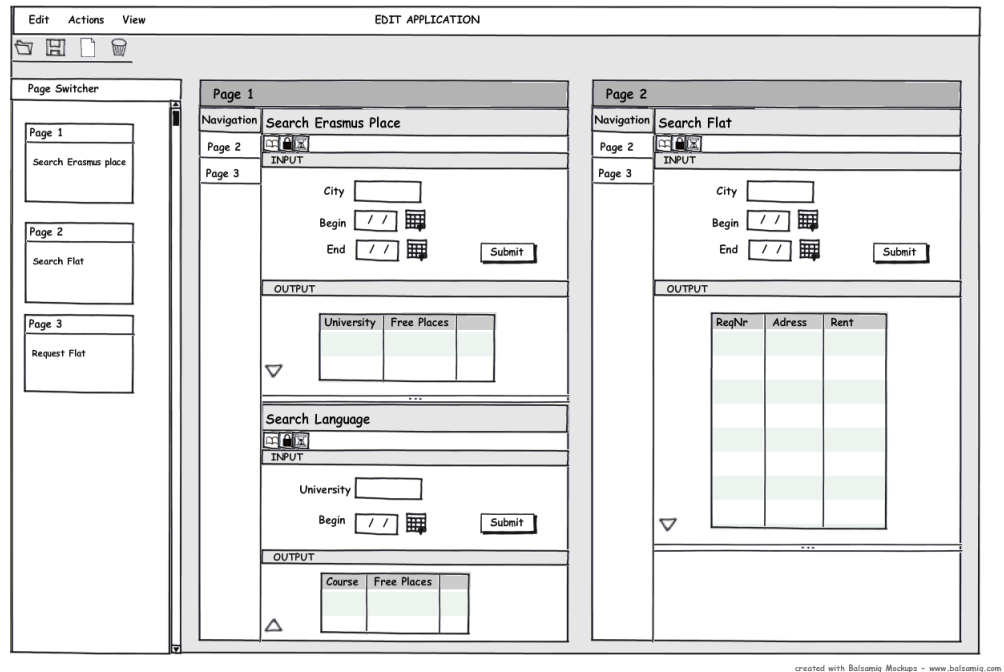


Abbildung 5.2.: Zwei-Seiten-Ansicht

Der zweite Modus **Advanced Mode** bietet eine Übersicht über alle Seiten sowie deren Beziehungen untereinander und ist für erfahrene Benutzer geeignet. In diesem Modus können ebenfalls unkompliziert neue Transitionen erzeugt und gelöscht werden. Es gibt unterschiedliche Möglichkeiten, die Prozessansicht darzustellen. Auf die Ansicht des **Advanced Mode** wird im Unterkapitel 5.3 eingegangen.

5.2 FRONTEND- UND SEITENDARSTELLUNG

Die bisherige Modellierungsansicht im ServFace Builder mit der Visualisierung der Service Operationen als frei bewegliche Widgets folgt keiner gängigen Layoutkonvention und bereitete den Testnutzern Schwierigkeiten beim generellen Verständnis der Funktionalität. In der Anforderungsanalyse wurde einerseits gefordert, eine der gewählten Zielplattform (mobile oder Webanwendung) entsprechenden Darstellung zu entwickeln (Unterkapitel 5.2.1) sowie andererseits die präsentationsorientierte Darstellung zu optimieren (Unterkapitel 5.2.2). Zu-

letzt werden die konfigurierbaren Parameter des Seitenlayouts, die über den Layoutmanager manipuliert werden können, vorgestellt (Unterkapitel 5.2.3).

5.2.1 PLATTFORMSPEZIFISCHE DARSTELLUNG

Die im ServFace Builder erstellte Anwendung soll sich gemäß der gewählten Zielplattform verhalten. Das Layout soll sich an bekannten vergleichbaren Anwendungen der entsprechenden Plattform orientieren [A3].

Dazu müssen die in der Anforderungsanalyse genannten plattformspezifischen Layouteigenschaften, die auf den in der Analyse vorgestellten Konventionen (Abschnitt 3.2) beruhen, in der Darstellung berücksichtigt und die Integrationsweise der Frontends überarbeitet werden [A4]. Um dem Benutzer das Seitenkonzept näher zu bringen, soll die Seite weniger abstrakt erscheinen. Die bisherige Darstellung kann laut Evaluation optimiert werden.

Bisher werden die Service Operationen als frei bewegliche Widgets visualisiert. Zwar kann dadurch der zur Verfügung stehende Platz optimal ausgenutzt werden, jedoch kann diese Flexibilität zu einem ungünstigen bzw. schlecht lesbaren Layout führen. Zudem orientiert sich die Widget-Darstellung an keinem geläufigem Layout. Eine alternative Möglichkeit besteht darin, in Anlehnung an formularbasierte Anwendungen bzw. Portlets die Frontends in die Seite zu integrieren und sie automatisch in einem definierten Bereich untereinander anzuordnen. Der Benutzer kann lediglich bestimmen, in welcher Reihenfolge die Frontends abgebildet sind.

In der Anforderungsanalyse wurde gefordert, dass verschiedene Navigationsstrukturen (freie, hierarchische, sequentielle) unterstützt und als Template vorgegeben werden. Somit ist auch eine Anpassung des Layouts abhängig von der Navigationsstruktur nach den in der Analyse vorgestellten Konventionen (Abschnitt 3.1) vorzunehmen [A11].

In folgenden Abschnitten werden die Layouts für Webanwendungen und mobile Anwendungen nach allgemeinen Layoutkonventionen und je nach Navigationsstruktur festgelegt.

5.2.1.1 DARSTELLUNG EINER WEBANWENDUNG

Für die drei verschiedenen Navigationsstrukturen wurde je ein mögliches Layout entwickelt (siehe Abbildung 5.3):

- **Freie Navigation:** Die Navigationselemente befinden sich im linken Randbereich der Seite
- **Sequentielle Navigation:** Es ist eine lineare Schritt-für-Schritt Navigation vorgegeben, dementsprechend befindet sich ein Interaktionselement im unteren Bereich der Seite,

das auf die nachfolgende Seite verlinkt. Zudem bietet die Pfadnavigation im oberen Bereich eine Orientierungsmöglichkeit, indem sie alle durchlaufenden Schritte, den aktuellen Stand eingeschlossen, anzeigt.

- **Hierarchische Navigation:** Der Aufbau der Anwendung ist hierarchisch organisiert, es gibt verschiedene Themengebiete (globale Navigation), die jeweils ein oder mehrere Unterkategorien (lokale Navigation) besitzen. Die globalen Navigationselemente werden permanent angezeigt, der lokale Navigationsbereich wechselt je nach ausgewählter Kategorie. Nach einer gängigen Konvention werden der globale Navigationsbereich in der oberen horizontalen Seitenleiste, die lokalen temporären Interaktionselemente im linken vertikalen Randbereich angezeigt.

Die bisherige abstrakte Darstellung der Seite soll wenn möglich angepasst werden, um eine Umgebung entstehen zu lassen, die den Voraussetzungen der Endanwendung entsprechen. Eine Webanwendung wird mit einem Browser ausgeführt, d.h. die Seiten werden innerhalb des Browserfensters dargestellt. Um eine Assoziation zu einem Browser herzustellen, wird eine „Browserleiste“ in den Rahmen der Seite integriert. Die Browserleiste enthält typische Elemente wie zum Beispiel einen Back- und Forward-Button sowie eine URL-Leiste.

The mockup shows a web page layout with a left sidebar for navigation and a main content area. The sidebar includes a home icon and three links: 'Erasmus place', 'Language course', and 'Accommodation'. The main content area has a header 'Search Erasmus Place' followed by an 'INPUT' section with three text boxes labeled 'City', 'Begin', and 'End', each with a calendar icon, and a 'Submit' button. Below this is an 'OUTPUT' section containing a table with four columns: 'ID', 'University', 'Free Places', and 'Ranking'. The table has one data row. Below the table is a 'Request Erasmus Place' section with an 'INPUT' section containing an 'ID' text box. The entire mockup is enclosed in a browser window frame with a 'PAGE 1' title bar and a URL bar showing 'http://www'.

created with Balsamiq Mockups - www.balsamiq.com

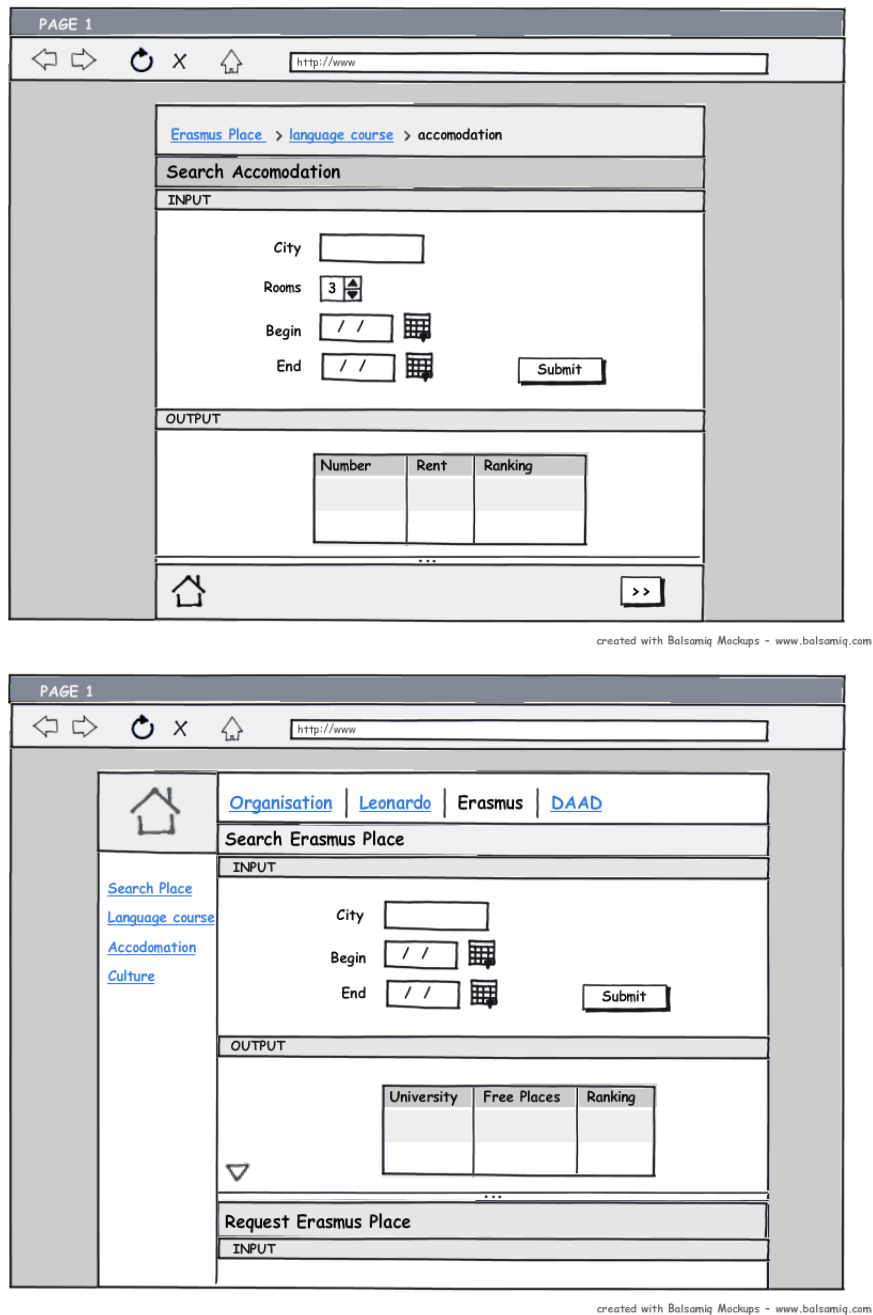


Abbildung 5.3.: Mockups für eine Webanwendung mit drei unterschiedlichen Navigationsstrukturen

5.2.1.2 DARSTELLUNG EINER MOBILEN ANWENDUNG

Mobile Bildschirme haben eine niedrigere Auflösung sowie eine kleinere Bildschirmgröße als ein Desktop-bzw. Laptop-Bildschirm, der die Ausführungsumgebung für eine Webanwendung darstellt. Um dem Benutzer einen kleineren Bildschirm mit niedriger Auflösung zu suggerieren und ihm damit das Platzverhältnis näher zu bringen, werden die Elemente einer Seite größer skaliert, zudem steht weniger Platz für die Modellierung zur Verfügung als bei der Erstellung einer Webanwendung.

Gemäß der Konvention sind die Elemente eines Frontends untereinander angeordnet im Gegensatz zu einer Webanwendung, wo zusammengehörige UI-Elemente nebeneinander angezeigt werden.

Das Drill-Down Menu für die hierarchische Navigation, die Interaktionselemente für die freie oder der „Next“-Button für die sequentielle Navigation werden alle direkt unterhalb der Service Frontends positioniert. Zur Orientierungshilfe kann eine Pfadnavigation in der oberen Horizontalen integriert werden. Somit entspricht der Modellierungsbereich dem Bildschirm des mobilen Gerätes (siehe Abbildung 5.4).

created with Balsamiq Mockups - www.balsamiq.com

Abbildung 5.4.: Seitendarstellung für eine mobile Anwendung

5.2.1.3 EVALUATION DER VERSCHIEDENEN FRONTEND-DARSTELLUNGEN

Im Mai 2010 wurde eine 30-minütige Benutzerstudie durchgeführt, die die alternativen Darstellungen mit frei positionierbaren Frontends und der im Rahmen dieser Arbeit ausgearbeiteten Variante mit einem strukturiertem Aufbau der Seite (Abschnitt A.2) vergleicht. Dazu haben 12 Teilnehmer mit dem ServFace Builder ein Anwendungsszenario mit beiden Layoutversionen durchgeführt und anschließend Fragen zu unterschiedlichen Aspekten der Bedienbarkeit und Benutzerfreundlichkeit beantwortet. Die Mehrheit (7 von 12 Teilnehmern) bevorzugte aufgrund der strukturierten Darstellung die Layoutvariante mit den integrierten Frontends. Anhand dieser Repräsentation konnte die Mehrheit sich die Endanwendung vorstellen und bewertete den Umgang ebenfalls überwiegend als positiv.

5.2.2 PRÄSENTATIONSORIENTIERTE DARSTELLUNG

Dem Benutzer fällt es schwer, zwischen Entwurfs- und Laufzeit zu unterscheiden [A5]. Innerhalb der Evaluation (Abschnitt 2.4) haben einige der Benutzer versucht, die Services in der Entwurfsansicht zu benutzen. In Bezug auf die im ServFace Builder angewandte Darstellungsweise nach dem WYSIWYG-Prinzip kann der ServFace Builder mit HTML-Editoren wie Adobe Dreamweaver ²⁰ oder MS Frontpage ²¹ verglichen werden, wo der Benutzer das Layout der Anwendung definiert und die UI-Elemente selbst konfiguriert. Die Darstellung in der Entwurfsansicht eines HTML-Editoren gleicht dem der Endanwendung, jedoch sind interaktive Elemente zum Teil noch deaktiviert oder mit Tags gekennzeichnet (siehe Abschnitt 3.4).

Das Konzept knüpft an dieses Prinzip an: Die UI-Elemente sind nicht mehr nutzbar, um dem Anwender die Differenzierung zur Laufzeitanwendung zu erleichtern. Interaktive UI-Elemente wie z.B. ein Textfeld oder eine Combobox werden zudem mit Beispieldaten gefüllt, damit der Benutzer eine Vorstellung bekommt, welche Daten zu benutzen sind.

Das WYSIWYG-Prinzip wird nicht vollkommen erfüllt, da innerhalb der Frontends Konfigurationswerkzeuge vorhanden sind, die zur Laufzeit nicht mehr angezeigt werden und nur für die Entwurfsansicht relevant sind [A6]. Alle Konfigurationswerkzeuge (Entfernen eines Frontends, Verstecken/Umbenennen eines Elementes etc.) werden deswegen in einem unabhängigen Inspektor – eine losgelöste Werkzeugpalette zur Konfiguration des aktuell markierten UI-Elements [Inc] – ausgelagert, um das Layout der späteren Endanwendung von den Konfigurationswerkzeugen klar trennen zu können. Der Inspektor erscheint, sobald der Benutzer ein Element, was bearbeitet werden kann, selektiert. Dieses Prinzip wird in

²⁰ <http://www.adobe.com/products/dreamweaver/>

²¹ <http://office.microsoft.com/de-de/frontpage-help/>

Editoren wie Balsamiq ²² angewandt, wo das Layout in der Entwurfsansicht ebenfalls mit dem Endprodukt übereinstimmt.

5.2.3 LAYOUTMANAGER

Um das Layout der Anwendung individuell gestalten zu können, soll dem ServFace Builder ein Layoutmanager hinzugefügt werden [A7]. Dabei soll dem Benutzer nicht zu viel Freiheit gelassen werden, da aus der Zielgruppe die meisten höchstwahrscheinlich über kein großes Hintergrundwissen bezüglich Layout verfügen. Vielmehr soll der Layoutmanager dazu dienen, das Design in eingeschränkten Maße individuell zu gestalten und ein Corporate Design bzw. firmenspezifische Merkmale wie z.B. ein Logo zu integrieren. Mittels einer „Masterpage“ soll das einheitliche Aussehen aller Seiten bestimmt werden. Dabei beinhaltet eine **Masterpage** folgende konfigurierbare Parameter:

- Freie Widgets oder feste Positionierung
- Hintergrundfarbe
- Hintergrundmedium

Auch die einzelnen Frontends sollen nach dem jeweiligen Inhalt der Seite bzw. dem vorhandenen Platz angepasst werden können. Der Benutzer kann demzufolge bestimmen, ob die Inhalte eines Frontends in ein oder zwei Spalten angezeigt werden sollen:

- Spaltenanzahl eines Frontends (ein oder zweispaltig)

Der Umfang eines Rückgabewerts kann insbesondere bei dem Datentyp Tabelle schlecht abgeschätzt werden. Um die Ausmaße einer Tabelle zu beschränken, kann der Benutzer folgende Parameter konfigurieren:

- Feste Anzahl an Zeilen
- Anzahl der angezeigten Spalten
- Ausblendung von Spalten
- Anpassung auf Seitengröße

5.3 DIALOGSTRUKTUR

Die Dialogstruktur soll in der Prozessansicht repräsentiert werden. In der Prozessansicht wird dem Benutzer einerseits eine Übersicht geboten, andererseits kann er neue Transitionen erstellen. Dafür benötigt er eine möglichst informative und – durch das begrenzte Platzverhältnis bedingt – zugleich kompakte Darstellung. Diese Ansicht ist primär für die „Freie

²² <http://www.balsamiq.com/>

Navigationsstruktur“ von Bedeutung, da bei den anderen Navigationsarten die Struktur bereits bekannt ist, so dass es unter Umständen gar keiner Visualisierung bedarf.

Es sollen verschiedene Darstellungsmöglichkeiten ausgearbeitet werden, die die in der Anforderungsanalyse genannten Aspekte und Kriterien berücksichtigen [A9].

5.3.1 DARSTELLUNGSMÖGLICHKEITEN

In der Analyse wurde untersucht, wie die Dialogstruktur visualisiert werden kann. Die Struktur ist auf einen Graphen zurückzuführen, dementsprechend bieten sich bereits bekannte Visualisierungen von Graphen an. Für eine abstraktere Sichtweise wurden verschiedene Möglichkeiten zur Visualisierung von Beziehungen aus dem Bereich der Visuellen Programmierung miteinbezogen. Abschließend wird bewertet, welche Darstellung die Anforderungen am besten erfüllt. Die Kriterien wurden in der Anforderungsanalyse (Abschnitt 4.3.1) definiert, zudem sollen die Navigationsstrukturen abgebildet werden. Im Folgenden werden mögliche Darstellungen erläutert:

Matrix

Da der typische Anwender in dem Umfeld Büro beschäftigt ist, kann eine gewisse Erfahrung im Umgang mit Microsoft Office Produkten vorausgesetzt werden. Eine Option ist, die Übersicht in Anlehnung an Microsoft Office Excel ²³ in einer Tabelle bzw. Matrix abzubilden (Abbildung 5.5).

Alle vorhandenen Seiten werden in Thumbnailgröße mit einer optional anzeigbaren Beschreibung des Seiteninhalts in den Zeilen- und Spaltenköpfen eingetragen. Ein Kreuz in einem Kästchen innerhalb der Matrix steht für eine Verbindung zwischen den entsprechenden Seiten. Eigenschaften einer Transition und Datenflüsse können durch verschiedene Symbole (Kreuz, Kreis..) gekennzeichnet werden. Für beide möglichen Ausrichtungen einer Transition ist je ein Kästchen vorhanden. Durch Deaktivierung eines Kästchens können unerwünschte Transitionen unterbunden werden. Die Gruppierung von Seiten innerhalb einer hierarchischen Navigationsstruktur kann durch eine entsprechende Einfärbung der Seiten repräsentiert werden. Der Gesamtkontext der Dialogstruktur ist dieser Ansicht jedoch schwer zu entnehmen. Zudem kann diese Darstellung bei einer großen Anzahl von Seiten schnell unübersichtlich werden. Alle Navigationsmuster können zwar erstellt werden, jedoch fehlt es an einer guten Visualisierung.

Flussgraph

Eine andere Möglichkeit ist, die Darstellung an gängigen Sitemap-Darstellungen, wie sie auf Webseiten oder innerhalb des Erstellungsprozesses einer Webanwendung in HTML-

²³ <http://office.microsoft.com/de-ch/excel/>

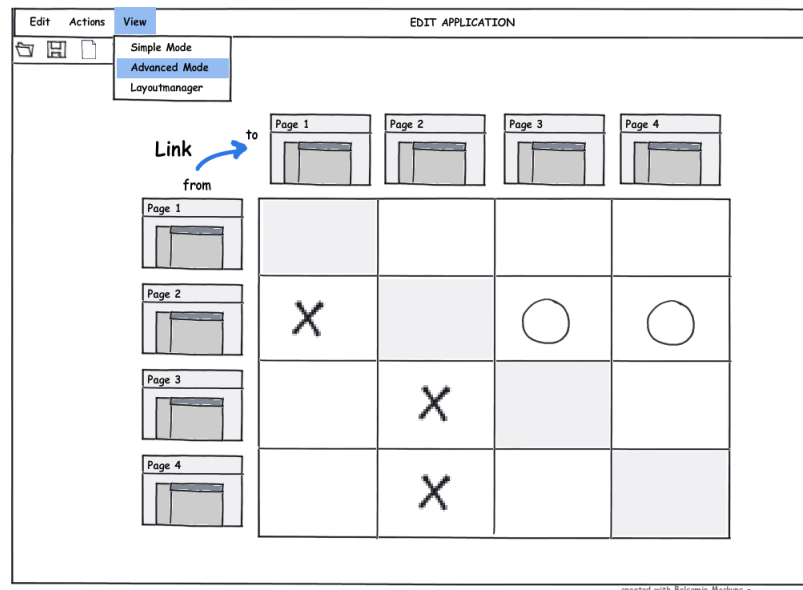


Abbildung 5.5.: Dialogstruktur als Matrix

Editoren wie Adobe Dreamweaver ²⁴ üblich sind, zu orientieren. Diese Darstellung ähnelt der aktuellen Prozessansicht. Der Dialogstruktur liegt eine Seiten- und Verbindungsmenge zugrunde, dementsprechend kann sie als Flussgraph mit den Seiten als Knoten und den Transitionen als Kanten abgebildet werden.

Die Seiten werden als Thumbnail repräsentiert, um den Bezug zur modellierten Seite innerhalb der Anwendung deutlich zu machen. Da diese Darstellung aufgrund der schlechten Lesbarkeit keinen großen Informationsgehalt besitzt, hat der Benutzer zusätzlich die Möglichkeit, Kontextinformationen in Form eines Tooltips zu einer Seite abzufragen. Die Kontextinformation enthält die Namen der integrierten Service Operationen.

Die Graphen-Ansicht orientiert sich an der Whiteboard-Metapher: Die Seiten werden an die Wand „gepinnt“ und durch einen Kreidestift visuell verbunden (Abbildung 5.6). Die Ausrichtung einer Transition kann durch einen Pfeil visualisiert werden. Weitere Transitionseigenschaften oder Datenflüsse können durch verschiedenfarbige Verbindungen bzw. verschiedene Linienarten repräsentiert werden. Die Navigationsstruktur wird durch den Aufbau des Graphen vermittelt. Bei komplexen, mehrdimensionalen Transitionen kann die Darstellung überladen wirken. Ansonsten bietet diese Darstellung eine gute Übersicht über den Gesamtkontext.

²⁴ <http://www.adobe.com/products/dreamweaver/>

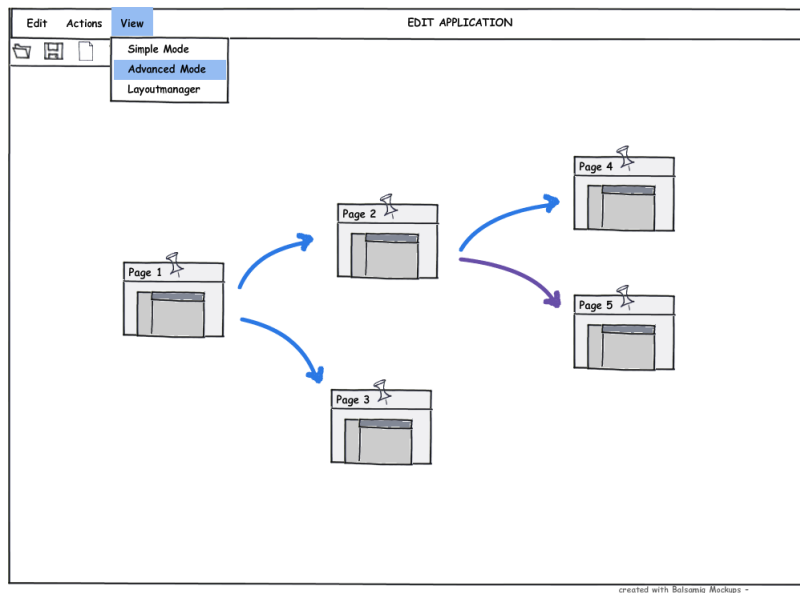


Abbildung 5.6.: Dialogstruktur als Flussgraph

Haus-Raum Varianten

In Anlehnung an das Autorensystem ShareMe (Abschnitt 3.5.3.2) kann die Haus-Raum Metapher eingesetzt werden. Ein Raum repräsentiert eine Seite, das Haus, welches aus den Räumen besteht, ist die gesamte Anwendung. In einem Raum können verschiedene Aspekte der Seite platziert werden, die Service Operationen sind Objekte innerhalb des Raumes. An einer Wand ist das Erscheinungsbild der späteren Anwendungsseite projiziert. Es wurden 2 Varianten für die Haus-Raum Metapher entwickelt:

Haus-Raum Variante 1: Alle Räume, die aneinander angrenzen, sind in der Endanwendung miteinander verbunden. Eigenschaften einer Transition können über verschiedene Farben oder durch die Art(dick, gestrichelt) der gemeinsamen Kanten visualisiert werden. Ein Datenfluss zwischen zwei Seiten kann durch ein Objekt, welches sich in beiden Räumen befindet, repräsentiert werden. Neben einer 3D-Ansicht des Hauses stehen dem Benutzer die Grundrisse der Stockwerke zur Verfügung, so dass er sehen kann, wie die Räume zusammenhängen und somit eine Übersicht erhält. Innerhalb der 3D-Übersicht kann der Benutzer in einen Raum hereinzoomen. Bei dieser Variante sind die Verbindungen auf die Anzahl der Außenwände, d.h. bei einem kubischen Raum auf 6 Transitionen, begrenzt.

Haus-Raum Variante 2: Eine alternative Möglichkeit, die keine Einschränkung bei der Anzahl von Verbindungen vorgibt, besteht darin, die Räume durch Türen zu verbinden (Abbildung 5.7). Durch Öffnen einer Tür kann zu dem entsprechenden Raum gewechselt werden. In der Gesamtübersicht des Hauses (3D-Ansicht, Grundrisse) steht für jede Verbindung ein

Vermerk in beiden Räumen. Die hierarchische Navigationsstruktur kann mit Hilfe einer homogenen Einfärbung von zusammengehörigen Räumen visualisiert werden. In beiden Varianten kann der Gesamtaufbau zwar repräsentiert werden, erscheint jedoch eventuell unübersichtlich. Dafür wird eine anschauliche Darstellung der Seiten geboten.

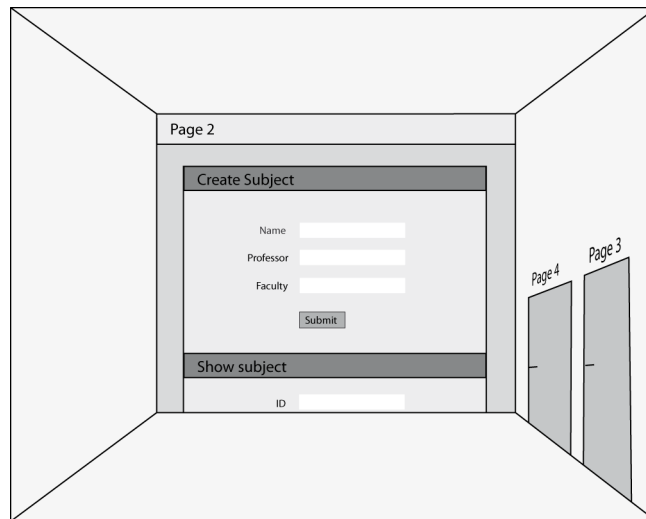


Abbildung 5.7.: Dialogstruktur als Haus-Raum Metapher, Variante 2

Film

Analog zur Film-Metapher, die für zeitliche Abhängigkeiten von Medienobjekten innerhalb Autorenwerkzeugen verwendet wird, kann der zeitliche Ablauf einer seitenbasierten Anwendung mit Hilfe einer Zeitleiste gesteuert werden (Abbildung 5.8).

Die Zeitleiste beinhaltet sogenannte Bilder (Zeiteinheiten), ein Bild steht für die Präsentation einer Seite in der Endanwendung. In der Zeitleiste ist für jede Seite eine Ebene vorhanden, in welcher die Lebensdauer der Seite durch einen gerichteten Pfeil definiert wird. Die Lebensdauer bestimmt, wann die Seite in der Endanwendung angezeigt wird. Grenzt die Lebensdauer einer Seite an die gerichtete Pfeilspitze einer anderen Seite an, besteht eine Transition zwischen den beiden. Auf diese Weise können beliebige Transitions für die freie, sequentielle und hierarchische Navigation abgebildet werden. Um die Zusammengehörigkeit von Seiten der gleichen Kategorie bei einer hierarchischen Navigationsstruktur zu vermitteln, können die entsprechenden Ebenen gleich eingefärbt werden. Ein Datenfluss wird durch jeweils ein Icon innerhalb der Bilder repräsentiert. Die Eigenschaften einer Transition werden durch verschiedene Linienarten abgebildet. Auf der Bühne wird das Abbild der Seite präsentiert. Diese Visualisierung bietet eine gute Darstellung des Ablaufes der Anwendung, jedoch kann sie bei komplexen Transitions schnell unübersichtlich werden.

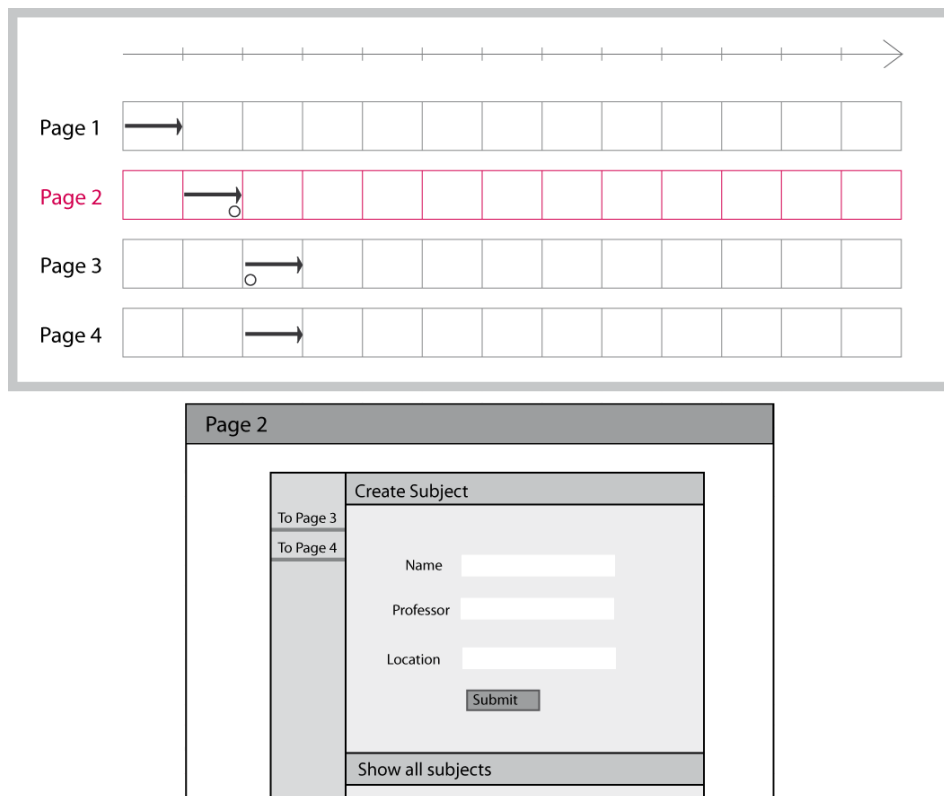


Abbildung 5.8.: Dialogstruktur als Film

5.3.1.1 BEWERTUNG ANHAND DER KRITERIEN UND EINER BENUTZERSTUDIE

Die vorgestellten Konzepte für die Darstellung der Dialogstruktur sollen nun nach den in der Anforderungsanalyse spezifizierten Kriterien bzw. Aspekte bewertet werden. Die Vor- und Nachteile, die unter anderem in der Beschreibung angegeben sind, werden in Tabelle 5.1 ausgewertet.

Die Erstellung von Transitionen soll bereits im Simple Mode möglich sein. Die Prozessansicht soll vordergründig eine Übersicht bieten und den Gesamtkontext der Navigationsstruktur visualisieren. Kriterien wie die Übersichtlichkeit des Gesamtkontextes und die Visualisierung der Transitionseigenschaften müssen stärker gewichtet werden. Auch eine angemessene Repräsentation der verschiedenen Navigationsstrukturen hat Priorität. Nach den Bewertungskriterien erscheint die Alternative „Graph“ geeignet. Diese Alternative erfüllt die wichtigsten Kriterien weitgehendst. Die Konzeption der anderen Möglichkeiten wird nicht weiter verfolgt.

Im Februar 2010 wurde an der Universität Manchester eine Benutzerstudie durchgeführt, die unter anderem die verschiedenen Darstellungen der Dialogstruktur durch eine Matrix

KRITERIUM	Matrix	Graph	Haus- Raum 1	Var. 2	Haus- Raum 2	Var.	Film
<i>Darstellung aller Seiten und Transitionen</i>	+	+	+		+		+
<i>Gesamtkontext deutlich</i>	-	++	+		+		+
<i>Seitendarstellung</i>	+	+	++		++		+
<i>Flexible Erstellung von Transitionen</i>	+	+	-		+		+
<i>Übersichtliche Darstellung komplexer Transitionen</i>	+	-	+		+		-
<i>Transitionseigenschaften</i>	++	++	+		+		++
<i>Datenflüsse</i>	+	+	+		+		+
<i>Navigationsstruktur</i>	+	++	+		+		+

Tabelle 5.1.: Bewertung der verschiedenen Aspekte der Darstellung der Dialogstruktur: +(erfüllt), ++ (sehr gut erfüllt), - (nicht erfüllt)

und einen Graphen anhand von Mockups sowie Befragungen evaluiert hat (ausführliche Beschreibung in Abschnitt A.1). Insgesamt haben 12 Personen aus der Zielgruppe an der 2-stündigen Benutzerstudie teilgenommen. Nach Prüfung des Vorwissens der Teilnehmer führten sie ein Anwendungsszenario durch. Anschließend wurden sie zu unterschiedlichen Design-Alternativen befragt. Die Mehrheit der Testpersonen haben die Repräsentation durch einen Graphen bevorzugt. So erschien ihnen diese Darstellung natürlich und übersichtlich, die Unterscheidung der Transitionen und Datenflüsse durch verschiedene Farben wurde ebenfalls als positiv aufgefasst. Für die Bewertung von Aspekten wurde die Likert-Skala mit Werten zwischen 1 („Ich stimme nicht zu“) und 5 („Ich stimme zu“) eingesetzt. Die Graphen-Darstellung bewerteten die Teilnehmer mit 4,2 für das Kriterium „leicht zu verstehen“, die Matrix-Darstellung dahingegen nur mit 3,4. Auch bei dem Kriterium „Effektivität“ schnitt erstere besser ab (Graphen-Darstellung:4,3 Matrix-Darstellung: 2,9). Viele Nutzer hielten die Graphen-Darstellung bei einer größeren Anwendung mit vielen Seiten oder komplexen Verbindungen für ungünstig, was sie jedoch ebenfalls für die Matrix-Darstellung als Problem identifizierten. Zudem betrachtete die überwiegende Mehrheit der Nutzer die Graphen-Ansicht als effektiv und erkannten diese Ansicht als sinnvoll an.

5.3.1.2 SPEZIFIZIERUNG DER GRAPHEN-DARSTELLUNG

Da die Graphen-Darstellung die präferierte Ansicht ist, wird die Untersuchung des Konzepts vertieft. Um die Darstellung als Graph zu verfeinern und zu optimieren, sollen folgende Aspekte berücksichtigt werden:

- Linien- bzw. Graphendarstellung: geradlinig, orthogonal
- Position der eingehenden bzw. ausgehenden Pfeile: Seitenrand, Ecke, Seitenmitte
- Berücksichtigung eines Leseflusses
- Struktur der Darstellung

In der Anforderungsanalyse wurde gefordert, dem Benutzer zu Anfang der Modellierung verschiedene Templates anzubieten, die eine bestimmte Navigationsstruktur vorgeben: hierarchische, sequentielle und freie Struktur. Davon abhängig soll die Dialogstruktur aufgebaut werden [A12]. Im Analysekapitel wurden bereits bekannte Darstellungen von Graphen erläutert. Es gilt, für jede Navigationsstruktur eine oder mehrere Graphen-Darstellungen auszuwählen und die definierten Kriterien anzupassen.

Sequentielle Navigationsstruktur

Die sequentielle Navigationsstruktur orientiert sich an der linearen Abfolge der Seiten im Pageswitcher. Die Navigationsstruktur wird somit im Simple Mode erstellt und bedarf keiner weiteren Ansicht, da diese keine neuen Informationen enthält und somit redundant wäre.

Hierarchische Navigationsstruktur

Der Benutzer kann mittels eines Interfaces die globalen und lokalen Kategorien auswählen. Die Visualisierung erfolgt durch eine Baum-Darstellung. Unter dem Wurzelknoten (Startseite) befinden sich mindestens zwei Ebenen. Die erste Ebene enthält alle globalen Elemente, wobei diese alle verbunden sind. Die Unterpunkte einer Kategorie befinden sich in derselben Ebene und sind ebenfalls alle miteinander verbunden. Dieses Prinzip wird für alle weiteren Unterkategorien fortgesetzt. Gewöhnlich existieren bei hierarchischen Navigationsmustern keine gerichteten Verbindungen, alle Transitionen sind bidirektional. Insofern werden die Transitionen als Linien visualisiert, wobei der Anknüpfungspunkt die jeweilige Mitte des Seitenrands ist. Datenflüsse können entweder über farbige, polygone Pfeile (gemäß Abbildung 3.1 „Down-to-Grandchild“) oder über eine Einfärbung der betroffenen Seitenrahmen visualisiert werden. Neben der bidirektionalen Verbindung von Überkategorie zu allen Unterkategorien sollen auch Seiten, die sich innerhalb derselben Ebene befinden und zu einer Kategorie gehören, miteinander verbunden sein. Somit ist die Darstellung nicht planar und es empfiehlt sich, orthogonale Verbindungen einzusetzen (gemäß Abbildung 3.1, „Across-to-Sibling“).

Der Benutzer kann zwischen einer radialen oder nach dem Top-Down Prinzip angeordneten Baumstruktur wählen, wobei von dem Wurzelknoten aus die Leserichtung zu sehen ist.

Freie Navigation

Nachdem der Benutzer im Simple Mode Transitionen erstellt hat, erhält er im Advanced Mode eine Übersicht über die Dialogstruktur sowie über die Seiten, die noch nicht verbunden sind. Seiten, die in keine Transition integriert sind oder von denen nur eine Transition ausgeht (die aber nicht die Startseite ist), sind in der Endanwendung nicht erreichbar. Diese Seiten werden visuell hervorgehoben, um den Anwender aufzufordern, diesen Zustand zu ändern. Um die Lesbarkeit der Visualisierung zu erhöhen, werden einige der vorgestellten ästhetischen Kriterien eingehalten. Es werden Algorithmen angewendet, die die Kantenlängen möglichst einheitlich halten sowie eine ausgewogene Verteilung der Seiten bewirken. Der Benutzer kann die Transitionen völlig frei konfigurieren, somit sind die Verbindungen gerichtet und werden als Pfeile visualisiert. Der Benutzer hat die Möglichkeit, zwischen einer orthogonalen und einer geradlinigen Darstellung zu wählen. Bei einer orthogonalen Darstellung sind die Seiten in einem Raster angeordnet, wobei die Rasterelemente für die Visualisierung der Verbindungen einen gewissen Abstand besitzen. Die Kanten bestehen aus einer Kette von horizontalen und vertikalen Segmenten entlang des Rasters. Der Benutzer kann neue Seiten in die Dialogstruktur per Drag & Drop integrieren bzw. bestehende verschieben, die entsprechende Seite rastet ein. Bei einer geradlinigen Darstellung knüpfen die Verbindungen an der jeweiligen äußeren Seitenmitte an, bei einer bidirektionalen Verbindung werden die Anknüpfungspunkte entsprechend versetzt. Um einen Lesefluss von der Startseite aus darzustellen, benötigt man die Festlegung der Startseite. Diese kann entweder automatisch identifiziert werden (Seite, welche in der chronologischen Reihenfolge am vordersten steht und von der nur Transitionen ausgehen) oder vom Benutzer explizit festgelegt werden. Die geradlinige Darstellung wird im Allgemeinen als ästhetischer empfunden, jedoch kann eine orthogonale Darstellung bei komplexen Transitionen übersichtlicher wirken.

5.3.2 UNTERSTÜTZUNG BEI DER MODELLIERUNG VON BEZIEHUNGEN

Abhängig von verschiedenen Aspekten können Transitionen automatisch erzeugt oder vom System unterbunden werden. In der Anforderungsanalyse wurde Unterstützung aus Gründen der Benutzerfreundlichkeit gefordert sowie die Einflussfaktoren spezifiziert. Auch die definierten Navigationsstrukturen müssen berücksichtigt werden. Das Ziel ist dabei, der Endanwendung ein möglichst natürliches Verhalten zu verleihen. Im Folgenden wird die Unterstützung in Form von Unterbindung oder Generierung von Transitionen erläutert:

Unterbindung von Transitionen

Wenn eine Seite eine nicht idempotente Service Operation beinhaltet, sollte keine Transition auf eine der Vorgängerseiten möglich sein [A15]. In diesem Fall wäre dem Benutzer nicht bewusst, dass die Service Operation ausgeführt wurde und die Aktion nicht rückgängig machbar ist. Solche Transitionen werden vom System unterbunden.

Generierung von Transitionen

Abhängig von den Navigation-Templates können automatisch Transitionen erzeugt werden, die den Konventionen der ausgewählten Navigationsart entsprechen. Generell sollte aus Sicht der Benutzerfreundlichkeit unabhängig vom ausgewähltem Navigations-Template die Startseite erreichbar sein. Dies wird durch die automatische Generierung eines Navigations-elements mit einem entsprechenden Layout (Home-Icon, Logo), das in das Navigationsmenu einer Seite integriert ist, realisiert.

Die weitere Unterstützung des Systems wird abhängig von der Navigationsstruktur definiert:

Bei der freien Navigation kann der Benutzer die Anwendung beliebig konfigurieren, jedoch sollen Transitionen, die aus Sicht der Benutzerfreundlichkeit sinnvoll sind und die der Benutzer nicht unbedingt selbst erstellt, automatisch modelliert werden.

Um eine Anwendung unkompliziert durchlaufen zu können, sind Transitionszyklen innerhalb eines Navigationsgraphen förderlich [A13]. Wenn eine Seite mehrere Transitionen zu anderen Seiten besitzt, entsteht eine Verzweigung im Navigationsgraphen. Sobald der Benutzer einen Zweig ausgewählt hat, bleiben alle anderen unerreichbar. Das System generiert in diesem Fall für jede Seite einer Verzweigung eine Transition zur Ausgangsseite der Verzweigung. Ein Datenflusszyklus hingegen wird vom System unterbunden.

Bei der sequentiellen Navigation wird die Anwendung linear durchlaufen, d.h. die Navigationsstruktur ist festgelegt. In diesem Fall kann die Dialogstruktur durch das System erstellt werden, gemäß Empfehlung wird einerseits ein Interaktionselement für die Navigation auf die Folgeseite und andererseits eine Pfadnavigation für die Orientierung sowie zur Navigation zu den durchlaufenen Seiten benötigt.

Bei der hierarchischen Navigation ist der Aufbau vorgegeben, die Festlegung von lokalen und globalen Seiten ergibt die Dialogstruktur. Dabei können die Verbindungen zwischen Seiten, die zu einer Kategorie gehören, sowie zwischen den Hauptkategorien automatisch generiert werden.

Im Rahmen dieser Arbeit wird die Unterstützung des Systems abhängig vom Einflussfaktor Datenfluss nicht weiter betrachtet. Bei näherer Untersuchung hat sich eine größere Komplexität herausgestellt, die der Gewichtung der Themen nicht gerecht werden würde und

der Ausrichtung der Arbeit nicht entspricht. Abhängig von den definierten Dimensionen müssen verschiedene Interaktionselemente definiert sowie eine noch nicht absehbare Anzahl von Fallunterscheidungen getroffen bzw. Entscheidungsbäume erstellt werden, um einen realitätsnahen Einsatz zu ermöglichen. Im Anhang sind die Interaktionselemente, die für notwendig befunden wurden, definiert sowie ein paar erste Fallunterscheidungen tabellarisch aufbereitet (Abschnitte A.3, A.4).

5.4 ZUSAMMENFASSUNG

Es wurden zwei verschiedene Modi entwickelt, mit Hilfe deren die Funktionalität klar aufgetrennt ist. Der Simple Mode dient vorrangig zur Bearbeitung einer Seite, der Advanced Mode bietet eine Übersicht und Möglichkeit zur Bearbeitung der Navigationsstruktur. Desweiteren wurden unterschiedliche Seitendarstellungen abhängig von Navigationsstruktur und Zielplattform mit einer strukturierte Eingliederung der Frontends in die Seite vorgestellt. Laut einer Benutzerstudie wird die alternative Darstellung mit den integrierten Frontends im Gegensatz zu der bisherigen Darstellung mit frei positionierbaren Frontends bevorzugt. Für die präsentationsorientierte Darstellung empfiehlt sich, eine klare Trennung von Entwurfs- und Laufzeitansicht in Form von der Auslagerung der Konfigurationswerkzeuge in einen Inspektor vorzunehmen. Um dem Benutzer zu verdeutlichen, dass er sich in der Entwurfsansicht befindet, sollen die interaktive UI-Elemente mit Beispieldaten gefüllt und gleichzeitig nicht benutzbar sein. Damit der Benutzer das Layout der Seite in geringem Maße anpassen kann, wurden die Konfigurationsparameter definiert. Für die Darstellung der Dialogstruktur wurden mehrere Alternativen vorgestellt und diskutiert. Anhand einer Benutzerstudie und den Kriterien aus der Anforderungsanalyse wurde die Repräsentation durch einen Graphen ausgewählt und für die Navigationsstrukturen spezifiziert. Die Unterstützung bei der Modellierung von Beziehungen wurde abhängig von den Navigationsstrukturen und Aspekten der Benutzerfreundlichkeit festgelegt.

6 IMPLEMENTIERUNG

In diesem Kapitel wird erläutert, wie die im vorherigen Kapitel spezifizierten Konzepte umgesetzt wurden. Die Konzepte wurden beispielhaft innerhalb des ServFace Builders in angepasster Form implementiert. Als Zielplattform wird neben einer Webanwendung eine Applikation für Google Android, stellvertretend für eine mobile Anwendung, angeboten. Der hierarchische Navigationsmodus wurde nicht umgesetzt, da der ServFace Builder durch seine eingeschränkte Komplexität bedingt nicht dafür ausgelegt ist.

Zunächst wird eine Führung durch den derzeitigen Stand des Tools (Unterkapitel 6.1 mit Screenshots gegeben. Anschließend werden die notwendigen Erweiterungen der Software-Architektur des ServFace Builders im Abschnitt 6.2 sowie die Implementierung der wichtigsten Komponenten erläutert.

6.1 ANWENDUNG

Im ersten Schritt muss der Benutzer die zu erstellende Anwendung konfigurieren. In einem Anfangsdialog kann er die grundsätzlichen Einstellungen bzgl. der Zielplattform und des Navigationsmodus vornehmen. Anschließend beginnt die eigentliche Modellierung der Anwendung. Der Anfangszustand des ServFace Builders ist der „Simple Mode“, in dem eine Großansicht der Startseite angezeigt wird. Der Benutzer kann beliebig Seiten erstellen und Service Operationen per Drag & Drop in eine Seite einfügen. Ausgehend von der gewählten Zielplattform und dem Navigationsmodus wird nun ein entsprechendes Layout angeboten. Im Folgenden werden die Eigenschaften der Layoutalternativen kurz zusammengefasst:

- **Webanwendung:** Im Rahmen der Seite ist eine Browserleiste integriert. Die Service Frontends werden in einem vordefinierten Bereich untereinander eingefügt. Der Navigationsbereich variiert entsprechend des ausgewählten Navigationsmodus. Bei einer sequentiellen Navigation wird einerseits eine Pfadnavigation zur Orientierung im oberen Bereich sowie ein „Next“ Button zur sequentiellen Navigation im unteren Bereich der Seite angezeigt (Abbildung 6.1). Bei der freien Navigation werden die Interaktions-

elemente im linken Bereich angeordnet. Zudem wird von jeder Seite aus automatisch eine Verbindung auf die Startseite generiert, das Interaktionselement wird durch ein „Home“ Icon repräsentiert (Abbildung 6.1).

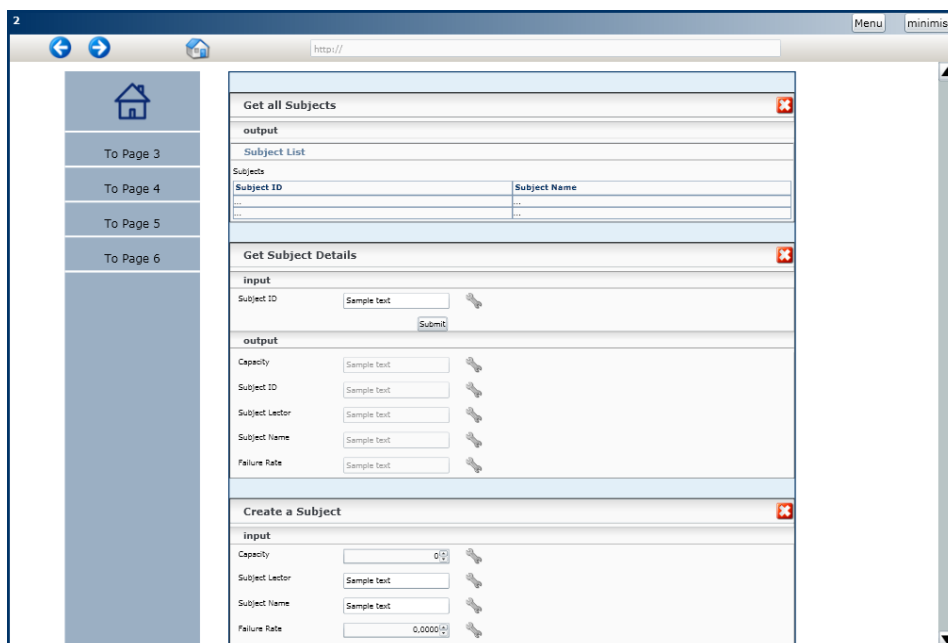
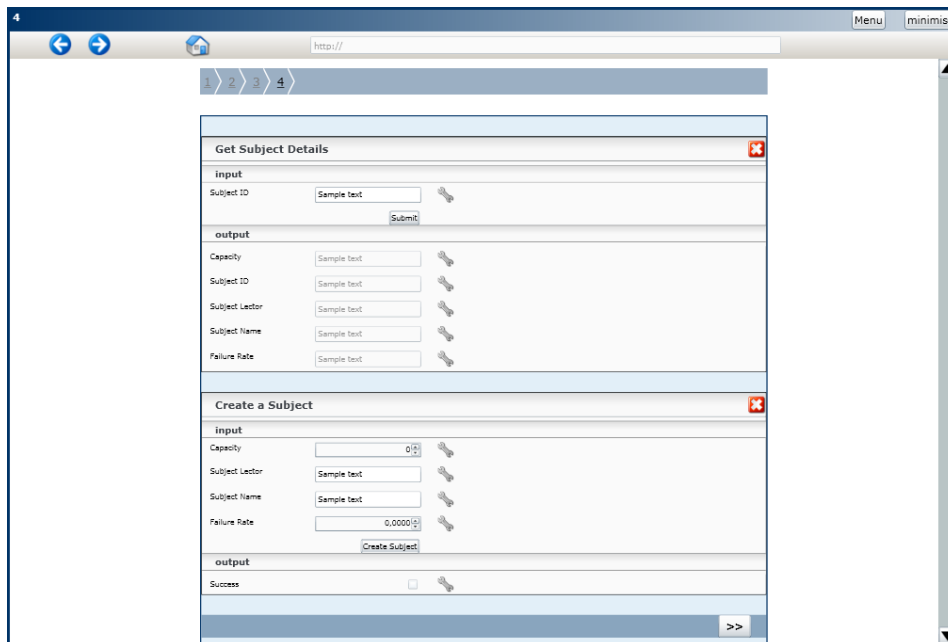


Abbildung 6.1.: Modellierungsansicht zur Erstellung einer Webanwendung mit sequentieller und freier Navigation

- **Google Android:** Der Modellierungsbereich repräsentiert den Bildschirm des Google Android Gerätes (Abbildung 6.2). Die Service Frontends werden untereinander eingefügt, der Navigationsbereich befindet sich im unteren Bereich. Insgesamt sind die Elemente größer skaliert als bei der Webanwendung.

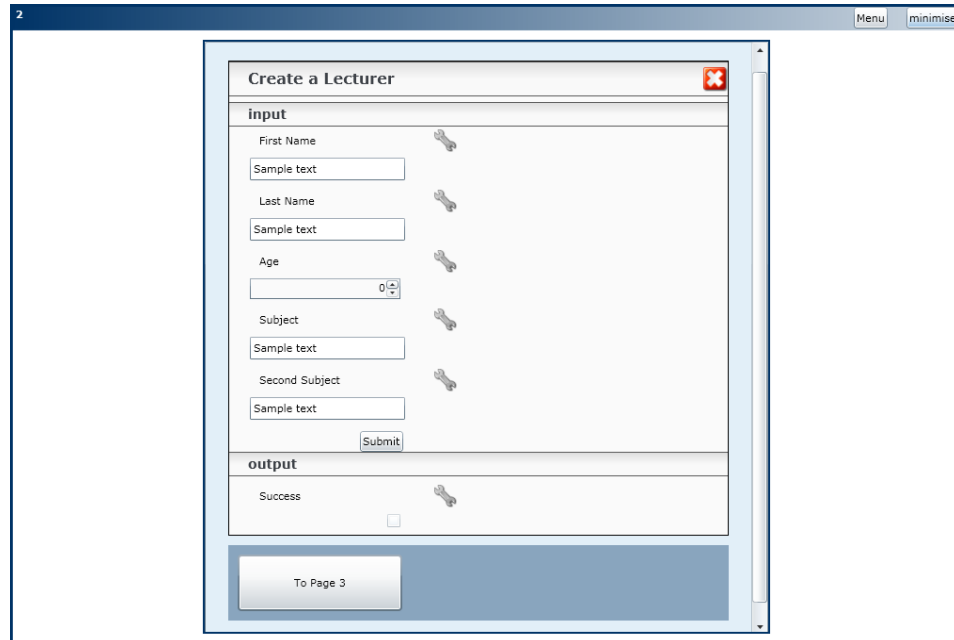


Abbildung 6.2.: Modellierungsansicht zur Erstellung einer mobilen Anwendung für Google Android

Anstatt die Navigation selbständig zu erstellen, wird im sequentiellen Navigationsmodus automatisch eine lineare Dialogstruktur entsprechend der Reihenfolge im Pageswitcher vorgegeben.

Im Navigationsmodus „Freie Navigation“ gibt es zwei Ansichten. Der Simple Mode ist die Großansicht einer Seite, neben der Konfiguration einer Seite können ebenfalls Transitionen erstellt werden. Jeder Seite im Pageswitcher ist ein Button beigegefügt, durch den eine Transition von der geöffneten Seite zu der entsprechenden Seite generiert werden kann. Nach Aktivierung des Buttons wird der geöffneten Seite automatisch ein entsprechendes Navigationsselement hinzugefügt und der Button, über den die Erstellung vorgenommen wird, nimmt ein anderes Layout an. Eine nochmalige Selektion des Buttons löscht die Transition. Wenn eine Transition unterbunden werden soll wie z.B. eine Transition, deren Start- und Zielseite identisch sind, ist der entsprechende Button deaktiviert (Abbildung 6.3).

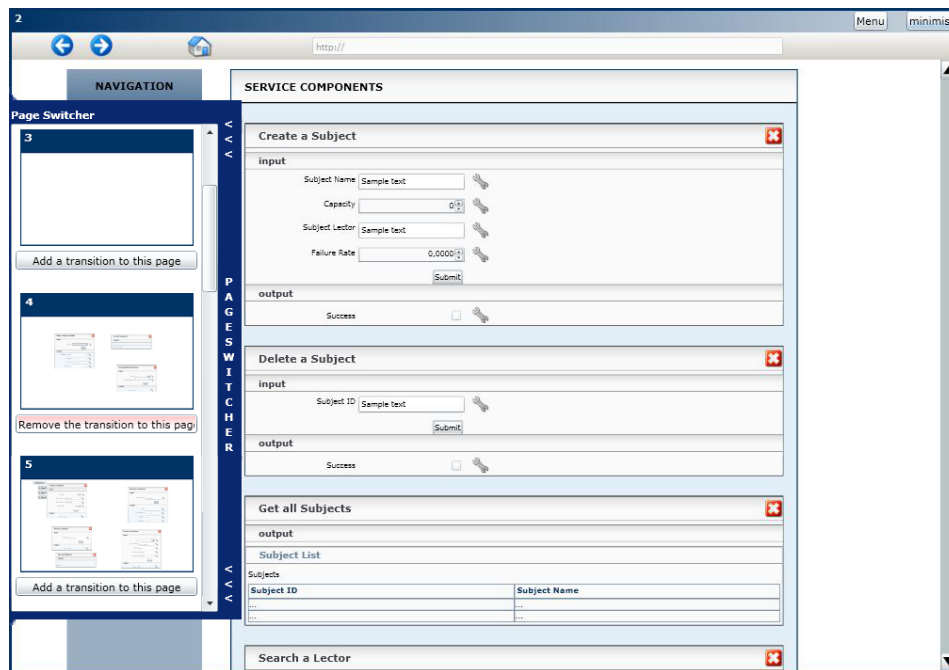


Abbildung 6.3.: Simple Mode: Erstellung einer Transition über entsprechenden Button

Um eine Übersicht über die erstellte Dialogstruktur zu bekommen, kann der Benutzer in den „Advanced Mode“ wechseln. In dieser Ansicht ist ein Graph mit den Seiten als Knoten und den Transitionen als gerichtete Verbindungen zu sehen. Die Seiten werden in Thumbnailgröße dargestellt, zusätzlich sind Kontextinformationen über den Inhalt der Seite in Form eines Tooltips abrufbar. Transitionen können nach wie vor in der Prozessansicht erstellt bzw. gelöscht werden. Seiten, die noch kein Teil des Navigationsgraphen sind, werden visuell hervorgehoben.

6.2 UMSETZUNG

Die Änderungen und Erweiterungen an dem ServFace Builder im Rahmen der Diplomarbeit betreffen vornehmlich das UI. Die Abbildung 6.4 zeigt ein vereinfachtes Klassendiagramm für die View-Komponente der MVC-Architektur. Die orange eingefärbten Klassen wurden im Rahmen dieser Arbeit verändert bzw. neu hinzugefügt.

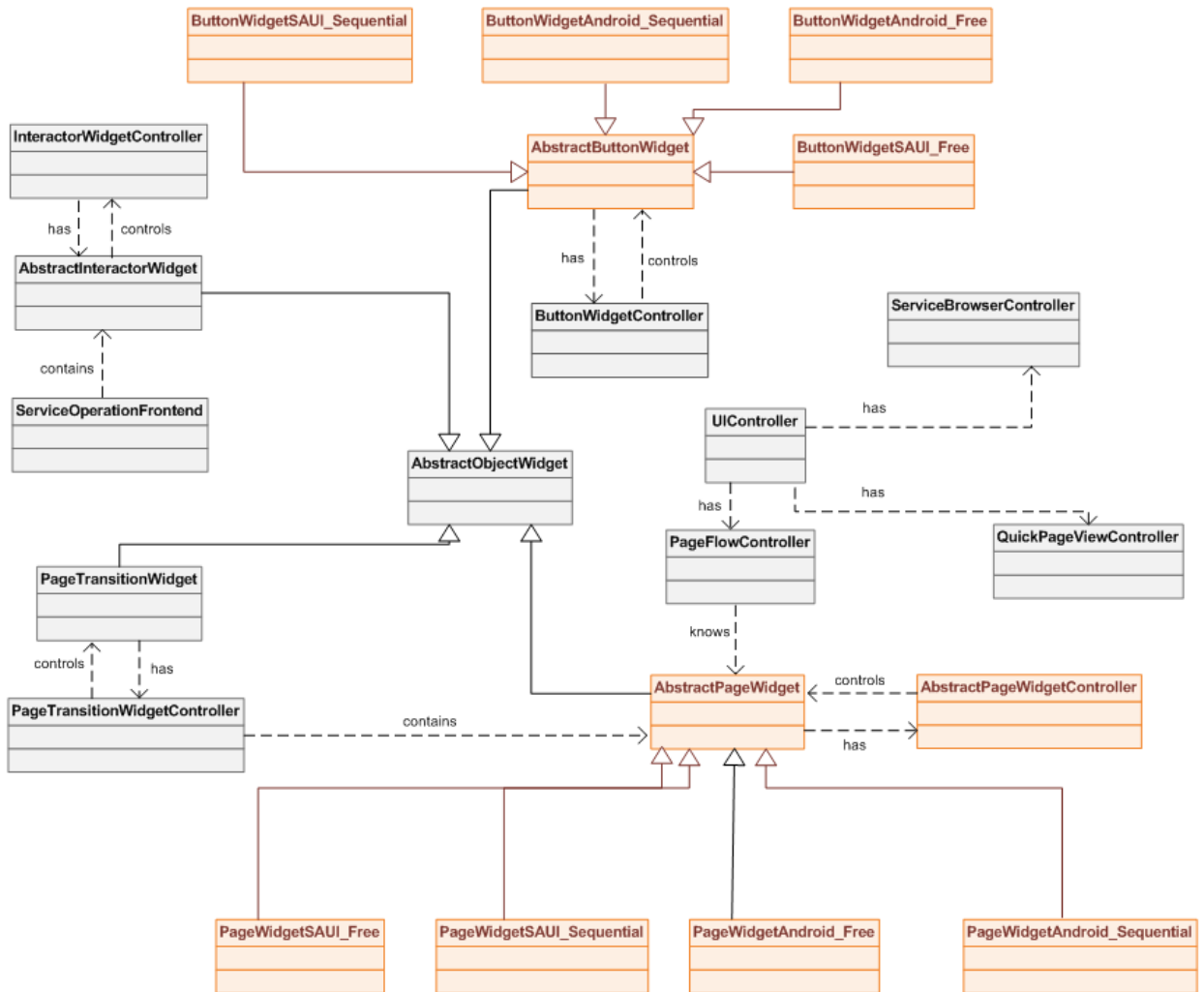


Abbildung 6.4.: Vereinfachtes Klassendiagramm der View

Es folgt eine genauere Beschreibung der wichtigsten Komponenten:

- **AbstractPageWidget:** Repräsentation einer Seite, enthält UI-Container für die Navigationselemente sowie für die Service Frontends.
- **PageTransitionWidget:** Repräsentation einer Transition zwischen zwei Seiten, enthält Informationen über Quell- und Zielseite.
- **AbstractInteractorWidget:** Alle Elemente eines Service Frontends, mit denen der Nutzer zur Laufzeit interagieren kann, erben von dieser Klasse. Alle Interaktionselemente verfügen über ein Interaktionsfeld wie beispielsweise ein Textfeld oder eine Datumsangabe, ein Label zur Beschreibung sowie ein Werkzeug-Icon zur Bearbeitung des Elementes.

- **UIController**: Verwaltung der Ansichten im ServFace Builder: den **PageFlowController**, der für die Darstellung der Prozessansicht zuständig ist, den **ServiceBrowserController**, der eine Auflistung aller verfügbaren Services enthält und den **QuickPageViewController**, der in der Seitenleiste alle Seiten in Thumbnailgröße anzeigt.

6.2.1 PLATTFORM- UND NAVIGATIONSSPEZIFISCHE SEITENDARSTELLUNG

Die Umsetzung der plattform- und navigationsspezifischen Darstellung erforderte neben einer Erweiterung auch eine Umstrukturierung des Klassenmodells des ServFace Builders. Alle Seitenlayouts verfügen über einen Modellierungsbereich, in dem die Service Frontends integriert werden, und einen Navigationsbereich. Jedoch unterscheiden sich die Gestaltung einzelner Elemente sowie die Positionierung bzw. Anordnung. Je Plattform und Navigationsmodus gibt es ein „PageWidget“, welches eine individuelle Anordnung des Modellierungs- und Navigationsbereichs definiert. Zudem existieren unterschiedliche „ButtonWidgets“, die verschiedene Interaktionselemente für die Navigationsmodi spezifizieren. Gemäß dem Factory Pattern entscheidet die „WidgetFactory“, welche konkreten UI-Elemente erzeugt werden. Somit existiert für jede Plattform eine „WidgetFactory“. Die gemeinsamen Eigenschaften der „PageWidgets“, „ButtonWidgets“ und „WidgetFactories“ werden in je einer abstrakten Klasse zusammengefasst. Abhängig von Zielplattform und Navigationsmodus existieren mehrere Widgets, die die entsprechenden abstrakten Klassen spezifizieren. Wenn möglich wurde nur ein Controller je abstrakte Klasse definiert, da das Verhalten oftmals äquivalent ist. Diese Erweiterungen sind in dem UML-Klassendiagramm (Abbildung 6.4) visuell hervorgehoben.

Nachdem der Benutzer am Anfang Zielplattform sowie Navigationsstruktur ausgewählt hat, werden diese Informationen global gespeichert. Abhängig von der Wahl der Zielplattform und Navigationsstruktur wird eine entsprechende „WidgetFactory“ instanziiert, die spezifische UI-Elemente generiert. Das Sequenzdiagramm (Abbildung 6.5) schildert den Ablauf für die Erstellung einer neuen Seite, wobei der Benutzer zuvor den freien Navigationsmodus und als Zielplattform Google Android ausgewählt hat. Die ausgewählte Plattform wird in der „RuleEngine“ gespeichert, der Navigationsmodus im „UIController“ gesetzt.

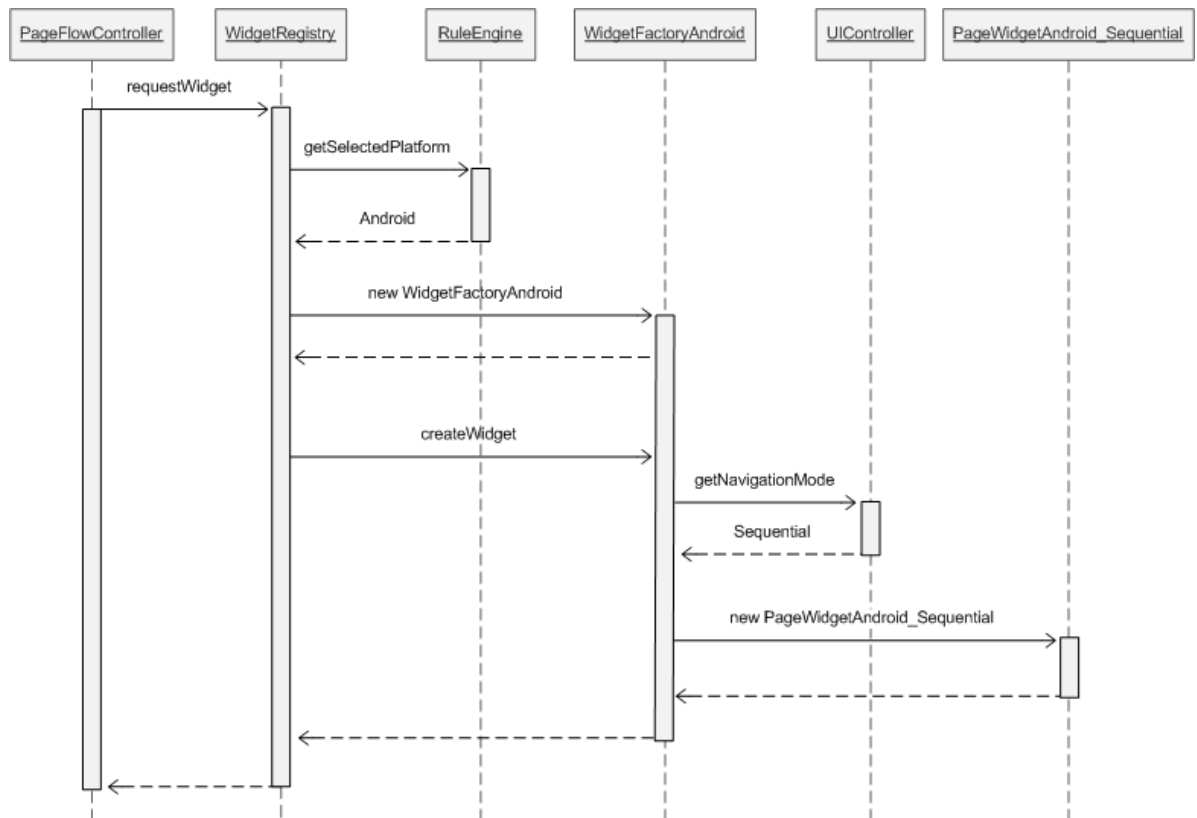


Abbildung 6.5.: Erstellung einer Anwendung für Google Android mit dem sequentiellen Navigationsmodus

6.2.2 PROZESSANSICHT

Die Prozessansicht wird für den freien Navigationsmodus angeboten und zeigt eine Übersicht über die Dialogstruktur. Um eine möglichst übersichtliche Darstellung des im Simple Mode erstellten Graphen zu gewährleisten, wird der „Force Directed Algorithm“ angewandt, der dafür sorgt, dass die Kanten möglichst gleichlang sind und dass eine ausgewogene Verteilung der Seiten besteht. Der Force Directed Algorithm betrachtet den Graphen als virtuelles physikalisches System und berechnet mit Hilfe physikalischer Gesetze die Kräfte, die von Knoten und Kanten eines Graphen ausgehen [Tun99]. Beim „Hookschen Gesetz“ werden die Kanten als Springfedern betrachtet. Dementsprechend stoßen sich die Knoten, die durch Springfedern miteinander verbunden sind, gegenseitig ab bzw. ziehen sich an. Zunächst werden alle Kräfte, die nach dem Hookschen Gesetz auf einen Knoten wirken, summiert. Anschließend wird das „Coulombsche Gesetz“ angewandt. Dieses ist aus dem Gebiet der Elektrostatik und behandelt die Anziehung und Abstoßung zwischen elektrisch geladenen Körpern. Die Knoten werden dabei als Körper mit der gleichen Ladung (1) betrachtet. Die

aus dem Coulombschen Gesetz resultierenden Kräfte werden addiert. Nachdem die Kräfte berechnet wurden, werden die Knoten entsprechend bewegt. Dieser Algorithmus wird iterativ wiederholt, bis ein ausgewogenes Verhältnis entsteht.

6.3 ZUSAMMENFASSUNG

Innerhalb dieses Kapitels wurde der aktuelle Stand des ServFace Builders anhand von Screenshots erläutert und die Implementierung der wichtigsten Komponenten beschrieben. Das Klassenmodell des ServFace Builders musste für die Umsetzung erweitert und umstrukturiert werden. Der Prozess für die Erstellung einer plattform- und navigationsabhängigen Seitendarstellung wurde beispielhaft mit Hilfe eines Sequenzdiagramms erklärt. Für die Visualisierung des Graphen, der die Navigationsstruktur in der Prozessansicht repräsentiert, wurde der „Force Directed Algorithm“ eingesetzt. Zusätzliche Beschreibungen zu der Implementierung sind im Anhang zu finden Abschnitt A.5. Für den ServFace Builder nicht relevante Erweiterungen wie z.B. eine hierarchische Navigation wurden nicht umgesetzt.

7 ZUSAMMENFASSUNG UND AUSBLICK

Im Rahmen dieser Arbeit wurden Konzepte für verschiedene Aspekte einer seitenbasierten Anwendung vorgestellt. Um eine benutzerfreundliche Interaktion zu erreichen, sollte einerseits eine angemessene Seitendarstellung entwickelt werden, die den präsentationsorientierten Ansatz verfolgt und für verschiedene Zielplattformen angepasst ist. Andererseits sollte der Benutzer einen Überblick über den Gesamtkontext, d.h. die Dialogstruktur der Anwendung, erhalten. Die Anforderungen und Konzepte knüpfen an die Referenzimplementierung „ServFace Builder“ an. Konflikte, die in einer Evaluation des ServFace Builders und damit stellvertretend für seitenbasierte Anwendungen deutlich geworden sind, sollten gelöst werden. Neben Optimierung des benutzerfreundlichen Umgangs wurden neue Zielplattformen und Navigationsstrukturen in den ServFace Builder eingeführt. Im Folgenden wird die Arbeit kurz zusammengefasst und ein Ausblick für zukünftige Arbeiten gegeben.

7.1 ZUSAMMENFASSUNG

Die Modellierungsansicht, in der die Seite konfiguriert wird, soll dem präsentationsorientierten Ansatz gemäß möglichst der späteren Endanwendung mit den möglichen Zielplattformen Webanwendung oder mobile Anwendung entsprechen. Im Analysekapitel wurden Layoutkonventionen bzgl. Seitenaufbau und Navigation für beide Plattformen vorgestellt. Darauf aufbauend konnte ein Konzept für die Darstellung festgelegt werden. Für eine präsentationsorientierte Darstellung wurden Prinzipien aus WYSIWYG-Werkzeugen untersucht, die in angepasster Form übernommen werden können. Um eine benutzerdefinierte Anpassung der Seitenansicht zu ermöglichen, wurde eine kleine Menge von Konfigurationsparametern festgelegt, die mit Hilfe eines Layoutmanagers modifiziert werden können.

Neben der Seitendarstellung war eine angemessene Erstellung und Repräsentation der Dialogstruktur gefordert. Die Visualisierung der Dialogstruktur soll dem Benutzer einerseits einen Überblick geben und andererseits die Möglichkeit bieten, sie zu modifizieren.

Um verschiedene Darstellungsmöglichkeiten zu finden, wurde eine umfassende Analyse durchgeführt, die neben verschiedener Graphen-Darstellungen auch in den Bereich der

Visualisierung von Beziehungen reicht. Anschließend wurde spezifiziert, welche Kriterien die Darstellung erfüllen muss. Auf der Analyse aufbauend wurden verschiedene Konzepte erarbeitet und diskutiert. Anhand der spezifizierten Kriterien und einer Benutzerstudie konnte eines ausgewählt werden.

Transitionen werden zunächst durch das ausgewählte Navigationsmuster festgelegt, sie können jedoch ebenfalls aus Gründen der Benutzerfreundlichkeit erstellt oder von Faktoren wie beispielsweise einem Datenfluss beeinflusst werden. In diesen Fällen soll das System unterstützend eingreifen. In der Konzeption wurden Transitionen vorgestellt, die abhängig von Kriterien der Benutzerfreundlichkeit und dem jeweiligen Navigationsmuster automatisch erstellt werden können. Um den Einsatz für alle möglichen Fälle der in der Anforderungsanalyse festgelegten Einflussfaktoren zu definieren, ist noch eine umfassende Untersuchung erforderlich.

Die Konzepte wurden zum größten Teil im ServFace Builder umgesetzt, wobei die bisherige modellorientierte Architektur eingehalten wurde.

7.2 AUSBLICK

Der aktuelle Stand des ServFace Builders ermöglicht eine intuitive Erstellung einer Anwendung für verschiedene Zielplattformen.

Wie bereits im Konzept angesprochen, kann die Graphen-Ansicht um das Interaktionsprinzip „Semantisches Zooming“ erweitert werden. Dadurch kann der Benutzer schrittweise mehr über eine Seite und über Verbindungen erfahren. Dabei wird bei näherem Heranzoomen die Seite nicht vergrößert, sondern es werden mehr informative Daten preisgegeben. Um den Kontext nicht aus den Augen zu verlieren, soll dem Benutzer jederzeit eine kleine Gesamtübersicht über die Dialogstruktur zur Verfügung stehen, in der der Benutzer ebenfalls sieht, wo er sich gerade befindet.

Da der hierarchische Navigationsmodus nicht den Voraussetzungen des derzeitigen Stands des ServFace Builders entspricht, wurde dieser nicht integriert. Für die Umsetzung müssen jedoch noch Bedienungskonzepte erstellt werden. Dem Nutzer soll eine Möglichkeit geboten werden, die globalen und lokalen Elemente über ein Interface festzulegen. Dabei ist zu überlegen, ob dies zu Anfang im Simple Mode oder im Advanced Mode entschieden werden soll. Eventuell lässt sich aus semantischen Zusammenhängen eine Empfehlung erstellen. Ziel dabei ist, die Bedienung möglichst intuitiv zu gestalten.

Für eine aussagekräftige Gestaltung der Dialogstruktur muss zunächst die Startseite identifiziert werden, um anschließend einen Lesefluss modellieren zu können. Die Startseite ist derzeit von vornherein festgelegt, jedoch soll diese vom Benutzer oder automatisch definiert

werden können. Die Startseite kann anhand von typischen Merkmalen wie z.B. nur ausgehende Transitionen identifiziert werden. Jedoch soll der Benutzer immer die Möglichkeit haben, diese zu verändern. Für die Darstellung eines Leseflusses bei der freien Navigation müssen noch passende Algorithmen untersucht werden.

Die automatische Erstellung von Transitionen wurde in Abhängigkeit von Navigationsstrukturen und Aspekten der Benutzerfreundlichkeit definiert. Zukünftig kann die Unterstützung abhängig vom Einflussfaktor Datenfluss und den in der Anforderungsanalyse definierten Dimensionen erweitert werden. Die Spezifizierung der Fallunterscheidung hat sich bei näherer Untersuchung als sehr komplex erwiesen, es muss eine detaillierte Fallunterscheidung mit Entscheidungsbäumen vorgenommen werden. Desweiteren müssen für die verschiedenen Einsätze spezielle Interaktionselemente definiert werden. Eine grobe Spezifikation der Interaktionselemente und Fallunterscheidung ist im Anhang (Abschnitt A.3, A.4) zu finden.

Der Layoutmanager kann anhand einer Benutzerstudie spezifiziert werden. Das Ziel ist herauszufinden, welche Konfigurationsparameter von den Benutzern erwünscht bzw. nicht benötigt werden. In einer früheren Evaluation wurde die Erweiterung um einen Layoutmanager befürwortet.

A ANHANG

A.1 EVALUATION 02/2010

Im Februar 2010 wurde an der Universität Manchester eine Benutzerstudie durchgeführt, die zum Ziel hatte, die Benutzerfreundlichkeit der aktuellen Version des ServFace Builders zu testen. Zudem sollten UI-Alternativen bewertet werden, um richtungsweisend für die Konzeptfindung zu wirken.

A.1.1 DURCHFÜHRUNG DER EVALUATION

An der Studie haben 12 Personen ²⁵ zwischen 25 und 34 Jahren teilgenommen, darunter 9 Studenten und 3 Angestellte des Verwaltungspersonals der Manchester Business School. Keine der Testpersonen verfügte über Programmier- oder Modellierungserfahrungen. Die Evaluation dauerte 2 Stunden und durchlief folgende Schritte:

1. Zu Anfang wurde das Vorwissen der Nutzer untersucht, indem sie zu mentalen Modellen und zu ihren Erfahrungen mit aktuellen Software Entwicklungsumgebungen befragt wurden.
2. Anschließend wurde mit einem Video-Tutorial und persönlicher Betreuung eine 15-minuten lange Einführung in den ServFace Builder gegeben.
3. Die Nutzer erhielten ein Szenario (Erstellung eines administrativen Verwaltungssystems für die Bearbeitung von Studentenbewerbungen mit Hilfe einer Menge von vorgegebenen Webservices) und sollten anschließend 12 Aufgaben mit dem ServFace Builder durchführen.
4. Danach haben die Benutzer einen Fragebogen ausgefüllt, der die Interaktion mit dem ServFace Builder nach verschiedenen Aspekten qualitativ bewerten sollte.

²⁵ 12 Teilnehmer ist die Mindestanzahl für eine aussagekräftige Benutzerstudie laut den Usability-Experten des ServFace Projektes

5. Zuletzt wurde den Benutzern verschiedene UI Alternativen anhand von Mockups vorgestellt. Sie sollten sich ausführlich zu den Vor- und Nachteilen äußern sowie welche Alternative sie bevorzugen.

A.1.2 RESULTATE

Die Benutzer wurden zu mehreren Erweiterungen (Layoutmanager, Zwei-Seiten-Ansicht, Simple Mode) sowie Design-Alternativen (Dialogstruktur als Flussgraph oder Matrix) befragt, die ihnen anhand von Mockups erläutert wurden. Zudem wurden jeweils Vor- und Nachteile genannt. Im Folgenden werden jeweils die von den Benutzern genannten Vor- bzw. Nachteile erläutert sowie die durchschnittliche Bewertung der 12 Teilnehmer von verschiedenen Aspekten der Benutzerfreundlichkeit nach der Likert-Skala mit Werten zwischen 1 („Ich stimme nicht zu“) und 5 („Ich stimme zu“). Dabei werden jeweils folgende Kriterien bewertet:

- Leicht zu erlernen
- Effektiv
- Nützliche Erweiterung

In den Tabellen sind jeweils der Durchschnittswert sowie die Standardabweichung angegeben.

Layoutmanager

Die Teilnehmer haben folgende Vorteile identifiziert:

- Konfigurationsmöglichkeit
- Individuelles Design, Anpassung
- Spaßfaktor

In der Tabelle A.1 sind die Bewertungen nach der Likert-Skala aufgelistet.

KRITERIUM	Durchschnitt	Standardabweichung
<i>Leicht zu verstehen</i>	4,6	0,7
<i>Effektiv</i>	4,2	1,1
<i>Nützliche Erweiterung</i>	4,5	0,9

Tabelle A.1.: Bewertung des Layoutmanagers nach der Likert-Skala

Zwei-Seiten-Ansicht

Die Teilnehmer haben folgende Vorteile identifiziert:

- Nützlich für Interpage-Datenfluss zwischen zwei Seiten
- Leicht zu verstehen
- Verbesserung

Die Teilnehmer haben folgende Nachteile identifiziert:

- Bei vielen Services ev. überladen
- Kann Fehler verursachen
- Ev. verwirrend

In der Tabelle A.2 sind die Bewertungen nach der Likert-Skala aufgelistet.

KRITERIUM	Durchschnitt	Standardabweichung
<i>Leicht zu verstehen</i>	4,4	1,2
<i>Effektiv</i>	4,0	1,3
<i>Nützliche Erweiterung</i>	3,8	1,6

Tabelle A.2.: Bewertung der Zwei-Seiten-Ansicht nach der Likert-Skala

Dialogstruktur als Flussgraph

Die Teilnehmer haben folgende Vorteile identifiziert:

- Gute Übersicht über alle Seiten und Verbindungen
- Natürliches Aussehen
- Navigationsfluss gut sichtbar
- Unterscheidung von Datenfluss und Transition durch Farben
- Einfach, leicht zu verstehen

Die Teilnehmer haben folgenden Nachteil identifiziert:

- Unübersichtlich bei vielen Seiten

In der Tabelle A.3 sind die Bewertungen nach der Likert-Skala aufgelistet.

KRITERIUM	Durchschnitt	Standardabweichung
<i>Leicht zu verstehen</i>	4,2	1,3
<i>Effektiv</i>	4,3	1,0
<i>Nützliche Erweiterung</i>	4,4	1,2

Tabelle A.3.: Bewertung der Graphen-Ansicht nach der Likert-Skala

Dialogstruktur als Matrix

Die Teilnehmer haben folgende Vorteile identifiziert:

- Verbindung zwischen einzelnen Seiten gut erkennbar
- Einfach zu bedienen
- Schönes Design, organisiert

Die Teilnehmer haben folgende Nachteile identifiziert:

- Überladen bei vielen Seiten
- Kein natürliches Aussehen
- Schwierig, Daten zu interpretieren
- Schwierig, Start- und Endseite zu identifizieren

In der Tabelle A.4 sind die Bewertungen nach der Likert-Skala aufgelistet.

KRITERIUM	Durchschnitt	Standardabweichung
<i>Leicht zu verstehen</i>	3,4	1,3
<i>Effektiv</i>	2,9	1,4
<i>Nützliche Erweiterung</i>	2,7	1,5

Tabelle A.4.: Bewertung der Matrixansicht nach der Likert-Skala

Simple Mode

Die Teilnehmer haben folgende Vorteile identifiziert:

- Unkomplizierte Möglichkeit, eine Verbindung zu modellieren
- Leicht zu verstehen
- Zeitsparend
- Insbesondere geeignet für Personen, die nicht aus dem IT-Bereich kommen

Die Teilnehmer haben folgende Nachteile identifiziert:

- Keine Gesamtübersicht über die Beziehungen
- Einschränkung
- Zu wenig Funktionalität

In der Tabelle A.5 sind die Bewertungen nach der Likert-Skala aufgelistet.

KRITERIUM	Durchschnitt	Standardabweichung
<i>Leicht zu verstehen</i>	4,2	1,1
<i>Effektiv</i>	2,8	1,5
<i>Nützliche Erweiterung</i>	2,8	1,6

Tabelle A.5.: Bewertung des Simple Modes nach der Likert-Skala

A.1.3 FAZIT

Der Layoutmanager sowie die Zwei-Seiten-Ansicht wurden als Erweiterungsmöglichkeiten überwiegend positiv aufgenommen. Im Hinblick auf die Bewertungen für die zwei Darstellungsalternativen der Dialogstruktur wird die Graphen-Ansicht klar bevorzugt. Die Ergebnisse lassen darauf schließen, dass der Simple Mode zwar nützlich, jedoch alleine nicht ausreichend ist und eine zusätzliche Ansicht für den Gesamtkontext verlangt wird.

A.2 EVALUATION 05/2010

Im Mai 2010 wurde eine Benutzerstudie durchgeführt, um die zwei Darstellungsalternativen für die Frontends und das Layout einer Seite („Freie Frontends“: freie Positionierung von Frontends, „Integrierte Frontends“: vordefinierte Integration der Frontends in die Seite) zu vergleichen. Das Ziel war herauszufinden, welche Darstellung von den Benutzern bevorzugt

wird und ob sie sich die Endanwendung anhand der Entwurfsansicht vorstellen können. Dazu wurden unterschiedliche Aspekte der Bedienbarkeit und Benutzerfreundlichkeit betrachtet.

A.2.1 DURCHFÜHRUNG DER EVALUATION

Die Benutzerstudie dauerte insgesamt 30 Minuten und wurde mit 12 Teilnehmern²⁶, die über unterschiedliches Vorwissen verfügten, durchgeführt. Die Evaluation lässt sich in folgende Schritte gliedern:

1. Zunächst wurde eine grobe Einordnung anhand des Vorwissens der Benutzer bzgl. Webanwendungen vorgenommen.
2. Anschließend wurden die Konzepte anhand einer Demonstration im ServFace Builder erläutert.
3. Den Benutzern wurde ein Anwendungsszenario vorgelegt, wobei sie eine Anwendung mit 5 Service Operationen erstellen sollten.
4. Die Benutzer führten das Anwendungsszenario mit beiden Layoutversionen durch.
5. Abschließend wurden die Teilnehmer befragt und füllten einen Fragebogen aus.

A.2.2 RESULTATE

Die Teilnehmer wurden zu beiden Varianten befragt. Die Mehrheit (7) bevorzugte die „Integrierten Frontends“.

Freie Frontends

Für die „Freien Frontends“ identifizierten die Teilnehmer folgende Vorteile:

- Spaßfaktor
- Flexibilität
- Individuelle Anordnung
- Mehr Gestaltungsmöglichkeiten
- Platzsparend

²⁶ 12 Teilnehmer ist die Mindestanzahl für eine aussagekräftige Benutzerstudie laut den Usability-Experten des ServFace Projektes

Als Nachteile wurden genannt:

- Flexibilität unnötig
- Unpraktisch
- Keine Hilfestellung bei der Positionierung

Integrierte Frontends

Für die „Integrierten Frontends“ identifizierten die Teilnehmer folgende Vorteile:

- Effizient
- Strukturiert, geradlinig
- Arbeitsfluss sichtbar
- Praktisch
- Zeitsparend
- Übersichtlich
- Insbesondere für Einsteiger geeignet
- Vorstellung einer Endanwendung vorhanden

Als Nachteile wurden genannt:

- Einschränkend, starr
- Keine Konfigurationsmöglichkeiten

Bei der Befragung zu Aspekten wie „Vorstellung einer Endanwendung“ sowie zur Qualität der Benutzung wurde die Likert-Skala eingesetzt mit Werten von 1 bis 7, wobei 1 für „Ich stimme überhaupt nicht zu“ und 7 für „Ich stimme vollkommen zu“ steht. Aus den Bewertungen der 12 Teilnehmer wurde jeweils der Durchschnitt sowie die Standardabweichung berechnet (Tabelle Bewertung der „Freien Frontends“ A.6, Tabelle Bewertung der „Festen Frontends“ A.7).

	Durchschnitt	Standardabweichung
<i>Endanwendung vorstellbar</i>	4,8	1,8
<i>Leicht zu erlernen</i>	5,3	1,3
<i>Leicht zu benutzen</i>	5,2	1,1
<i>Angenehm zu benutzen</i>	5,3	1,0
<i>Effizient zu benutzen</i>	5,2	1,4

Tabelle A.6.: Bewertung der Freien Frontends nach der Likert-Skala

	Durchschnitt	Standardabweichung
<i>Endanwendung vorstellbar</i>	6	1
<i>Leicht zu erlernen</i>	5,9	1,2
<i>Leicht zu benutzen</i>	5,7	1,0
<i>Angenehm zu benutzen</i>	5,3	1,0
<i>Effizient zu benutzen</i>	5,2	1,3

Tabelle A.7.: Bewertung der Integrierten Frontends nach der Likert-Skala

A.2.3 FAZIT

Die Mehrheit bevorzugte die Integrierten Frontends, wobei vor allem der strukturierte Aufbau geschätzt wurde. Die Endanwendung konnte zum Zeitpunkt der Evaluation mit dem ServFace Builder noch nicht erstellt werden, so dass die Bewertung des Layouts unter dem Gesichtspunkt der Interaktion mit der Endanwendung in dieser Studie nicht zum Tragen kommt. Es kann jedoch angenommen werden, dass bei der Nutzung ein strukturierter Aufbau, der mit den Freien Frontends nicht zwangsläufig erstellt wird, von Vorteil ist.

A.3 INTERAKTIONSELEMENTE

Ein Datenfluss oder eine Transition kann durch verschiedene Events ausgelöst werden. Manche Datenflüsse werden bereits nach Bestätigung der Eingabewerte initialisiert, andere benötigen die Definition eines eigenen Navigationselementes. Im Folgenden werden verschiedene Interaktionselemente spezifiziert, wobei unterschieden wird, ob ein Interaktionselement

an ein Event gebunden ist und ob das entsprechende Element automatisch vom System oder manuell vom Benutzer zu generiert wird.

- *Submit Button*: Der *Submit Button* ist im Eingabe-Bereich integriert und wird automatisch generiert. Durch Betätigung des *Submit Buttons* wird die Eingabe bestätigt und damit der Service aufgerufen. Ausserdem kann er einen Datenfluss ($I \rightarrow I$) auslösen.
- *Forward Button*: Der *Forward Button* wird automatisch vom System bei einem Datenfluss vom Typ $O \rightarrow I$ generiert, wenn das Quell-Ausgabeelement eine (komplexe) Tabelle ist. Die Aktivierung des *Forward Buttons* bestätigt die ausgewählten Ausgabedaten als Eingabe für einen definierten Datenfluss. Somit wird ein Datenfluss initialisiert und im Fall eines Interpage-Datenflusses wird ebenfalls eine Transition ausgelöst.
- *Multiple Forward Button*: Dieses Interaktionselement wird nur bei der Frontend-Abhängigkeit 1:N eingesetzt und ansonsten unter den gleichen Bedingungen wie ein *Forward Button* generiert. Für jeden Datenfluss, der von dem Quell-Frontend ausgeht, wird ein *Forward Button* generiert. Im Label eines *Multiple Forward Button* ist ebenfalls der Seitenna-me vermerkt, auf den die Transition zeigt. Durch einen *Multiple Forward Button* wird im Fall eines Interpage-Datenflusses eine Transition ausgelöst sowie ein Datenfluss initialisiert.
- *On Click*: Der Benutzer kann durch Klicken einer Zeile innerhalb einer Ausgabetabelle einen $O \rightarrow I$ Datenfluss auslösen. Dadurch wird das Element aus der ausgewählten Zeile und der zuvor definierten Spalte übertragen.
- *Navigation Button*: Ein *Navigation Button* löst eine Transition zu einer anderen Seite aus. Der Benutzer muss diese Transition manuell definieren, anschließend wird automatisch ein *Navigation Button* erzeugt. Der *Navigation Button* ist Bestandteil der Seite, nicht eines Frontends.
- *Boolscher Rückgabewert*: Der Benutzer kann eine Bedingung definieren, die an einen *boolschen Rückgabewert* eines Services geknüpft ist. Je nach Spezifikation kann ein Datenfluss, eine Transition oder ein Service-Aufruf ausgelöst werden. Die Bedingung wird in der Entwurfsansicht deklariert und besitzt in der Endanwendung keine Visualisierung.

A.4 KOMPOSITIONSPATTERN

In diesem Abschnitt sind mögliche Fälle spezifiziert, in denen das System bei der Modellierung von Relationen eingreifen kann. In Abhängigkeit von den in der Anforderungsanalyse identifizierten Dimensionen (Abschnitt 4.3.3.2) kann der Einsatz von den spezifizierten Interaktionselementen (Abschnitt A.3) in Form von Pattern beschrieben werden. Es wurden insgesamt 29 Pattern entworfen, in diesem Abschnitt sollen die Pattern (Tabelle A.8) auf die Fälle beschränkt werden, in denen nur zwei Frontends involviert sind (**Frontend-Abhängigkeit 1:1**).

Nr.	Frontend Beziehung	Frontend Verteilung	Ausgabetyt	Interaktionselement	
1	$I \rightarrow I$	Intra	-	Submit	
2	$I \rightarrow I$	Inter	-	Navigation Button	
3	$O \rightarrow I$	Intra	Single, Liste	Submit	
4	$O \rightarrow I$	Intra	Single, Liste	Boolscher wert	Rückgabe-
5	$O \rightarrow I$	Intra	(Komplexe) Tabelle	Forward	
6	$O \rightarrow I$	Intra	Tabelle	On Click	
7	$O \rightarrow I$	Inter	Single, Liste, (komplexe) Tabelle	Navigation Button	
8	$O \rightarrow I$	Inter	(Komplexe) Tabelle	Forward	
9	$O \rightarrow I$	Inter	Tabelle	On Click	
10	$O \rightarrow I$	Inter	Single, Liste	Boolscher wert	Rückgabe-
11	$I \rightarrow O$	Intra	-	Submit	
12	$I \rightarrow O$	Inter	-	Submit	

Tabelle A.8.: Klassifikation der Interaktionselemente abhängig von den definierten Dimensionen, dabei steht „I“ für Input und „O“ für Output

In dieser Tabelle werden mögliche Kombinationen aufgelistet. Jedoch bieten nicht alle Pattern eine natürliche Interaktion mit der Anwendung. Zum Beispiel sollte bei einem Intrapage-Datenfluss von Output zu Input kein *Forward-Button* verwendet werden, da dieser einen Seitenübergang suggeriert. *On Click* sollte dagegen nicht für einen Interpage-Datenfluss eingesetzt werden, da kein sichtbares Interaktionselement vorhanden ist und das Event nur indirekt durch Auswählen eines Elementes in der Tabelle ausgelöst wird. Somit könnte ungewollt eine Transition initiiert werden. Um ein möglichst typisches Verhalten der Anwendung zu offerieren, können die Betrachtungen erweitert werden: Wenn ein Eingabe-Bereich eines Frontends komplett durch einen Datenfluss befüllt werden kann, soll der Service-Aufruf bereits durch die Initiierung des Datenflusses erfolgen.

Die Transitionen, die durch einen Interpage-Datenfluss zustande kommen, sollten wenn möglich automatisch generiert werden. Es wurde festgelegt, dass aus Gründen der Benutzerfreundlichkeit bei einer Verzweigung Transitionen von Elementen der Verzweigung auf deren Ausgangspunkt erzeugt werden sollen. Bei einem Interpage-Datenfluss kann dementsprechend eine Transition notwendig sein, damit der Datenfluss zustande kommt, auf der anderen Seite soll gleichzeitig eine Transition zurück auf die Ausgangsseite generiert werden, falls es mehrere Transitionen von der Quell-Seite gibt.

A.5 SIMPLE MODE

Die beiden Modi Simple und Advanced Mode trennen die Funktionalität auf. Im Simple Mode bearbeitet der Benutzer vorrangig eine Seite, neben der Integration von Frontends und Erstellung von Datenflüssen kann er auch Transitionen modellieren. Die Generierung einer neuen Transition erfolgt über den „QuickPageSwitcher“. Dieser listet alle Seiten mit jeweils einem Interaktionselement für die Modellierung einer Transition auf. Dazu wird im „AbstractPageWidget“ der Zustand der Transition (keine/vorhanden) zu jeder anderen Seite in einem Dictionary gespeichert. Öffnet der Benutzer den Pageswitcher, fragt der „QuickPageSwitcher“ zunächst das jeweilige Dictionary der geöffneten Seite („AbstractPageWidget“) ab und visualisiert anschließend jede Seite in Thumbnailgröße mit einem Button, der den Zustand der Transition kennzeichnet.

ABBILDUNGSVERZEICHNIS

2.1. Abbildung der dreistufigen ServFace Methodologie ([Jug09])	6
2.2. Prozessansicht: Dialogstruktur der Anwendung (Screenshot)	7
2.3. Seitenansicht: geöffnete Seite, Möglichkeit zur Konfiguration (Screenshot) . .	8
2.4. Vereinfachtes Klassendiagramm	10
2.5. Diagramm zu der Frage „Hatten Sie eine Vorstellung davon, wie die finale Anwendung aussieht?“	15
3.1. Lokale Navigationsstrukturen: Down-to-Child, Across-to-Sibling, Up-to-Parent, Down-to-Grandchild [Adk05]	19
3.2. Pfadnavigation	21
3.3. Aufteilungsmöglichkeiten einer Seite: Vertikal, Horizontal, Kombination . .	24
3.4. Formular mit vertikal angeordneten Labels und Eingabefeldern [Wro]	26
3.5. Formular mit links ausgerichteten Labels in horizontaler Anordnung [Wro] .	27
3.6. Formular mit rechts ausgerichteten Labels in horizontaler Anordnung [Wro]	28
3.7. Trennung der Inhalte mit visuellen Hilfsmitteln	28
3.8. Trennung der Inhalte mit visuellen Hilfsmitteln	29
3.9. Priorität der Aktionen durch Farbe signalisiert	30
3.10. Adobe Dreamweaver: Darstellung der UI-Elemente innerhalb von Formular-Tags	33
3.11. Labviews: Programm aus Funktionsblöcken	39
3.12. Scratch: Programm aus Puzzleteilen	40
3.13. Hypercard: Verwendung der Metapher „Stapel“	41
3.14. Adobe Flash: Verwendung der Metapher „Film“	42
4.1. Aufgabenanalyse mit HTA	47
4.2. Derzeitige Visualisierung der Service Operationen als Widgets (Screenshot) .	49
4.3. Vermischung der präsentationsorientierten Darstellung und Konfigurationswerkzeugen	51
4.4. Prozessansicht als initiale Ansicht (Screenshot)	52
5.1. Simple Mode	62
5.2. Zwei-Seiten-Ansicht	63

5.3. Mockups für eine Webanwendung mit drei unterschiedlichen Navigationsstrukturen	66
5.4. Seitendarstellung für eine mobile Anwendung	67
5.5. Dialogstruktur als Matrix	71
5.6. Dialogstruktur als Flussgraph	72
5.7. Dialogstruktur als Haus-Raum Metapher, Variante 2	73
5.8. Dialogstruktur als Film	74
6.1. Modellierungsansicht zur Erstellung einer Webanwendung mit sequentieller und freier Navigation	82
6.2. Modellierungsansicht zur Erstellung einer mobilen Anwendung für Google Android	83
6.3. Simple Mode: Erstellung einer Transition über entsprechenden Button	84
6.4. Vereinfachtes Klassendiagramm der View	85
6.5. Erstellung einer Anwendung für Google Android mit dem sequentiellen Navigationsmodus	87

TABELLENVERZEICHNIS

2.1. Plattformspezifika	12
4.1. Plattformspezifika	50
4.2. Identifizierte Anforderungen	59
5.1. Bewertung der verschiedenen Aspekte der Darstellung der Dialogstruktur: +(erfüllt), ++ (sehr gut erfüllt), - (nicht erfüllt)	75
A.1. Bewertung des Layoutmanagers nach der Likert-Skala	94
A.2. Bewertung der Zwei-Seiten-Ansicht nach der Likert-Skala	95
A.3. Bewertung der Graphen-Ansicht nach der Likert-Skala	96
A.4. Bewertung der Matrixansicht nach der Likert-Skala	96
A.5. Bewertung des Simple Modes nach der Likert-Skala	97
A.6. Bewertung der Freien Frontends nach der Likert-Skala	100
A.7. Bewertung der Integrierten Frontends nach der Likert-Skala	100
A.8. Klassifikation der Interaktionselemente abhängig von den definierten Dimen- sionen, dabei steht „I“ für Input und „O“ für Output	102

LITERATURVERZEICHNIS

- [Abo] ABOUTWEBDESIGN.DE: *eBook: Die Webdesign-Referenz*. AboutWebDesign.de, Online Magazin
- [Adk05] ADKISSON, H.P.: *Web Design Practices: Navigation*. <http://www.webdesignpractices.com/index.html>. 2005
- [BETT94] BATTISTA, G. D. ; EADES, P. ; TAMASSIA, R. ; TOLLIS, I.G.: Algorithms for drawing graphs: an annotated bibliography. In: *Computational Geometry-Theory and Application*, 4, 1994, Nr. 5, S. 235–282
- [BETT98] BATTISTA, G. D. ; EADES, P. ; TAMASSIA, R. ; TOLLIS, I.G.: *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998
- [BHL91] BLAKOWSKI, G. ; HÜBEL, J. ; LANGREHR, U.: Tools for Specifying and Executing Synchronized Multimedia Presentations. In: *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1991
- [Bol] BOLES, D.: *Autorensysteme*. <http://www-is.informatik.uni-oldenburg.de/~dibo/teaching/mm/buch/node63.html>
- [BPGN04] BIER, E. ; POPAT, K. ; GOOD, L. ; NEWBERGER, A.: Zoomable user interface for in-depth reading. 2004, S. 424
- [DFN⁺10] DANNECKER, L. ; FELDMANN, M. ; NESTLER, T. ; HÜBSCH, G. ; JUGEL, U. ; MUTHMANN, K.: *Rapid Development of Composite Applications Using Annotated Web Services*. In *Proceedings of the 6th Model-Driven Web Engineering Workshop (at ICWE)*, 2010
- [Fis05] FISCHER, S. E.: *Der Information Worker*. 2005
- [FNJ⁺09a] FELDMANN, M. ; NESTLER, T. ; JUGEL, U. ; MUTHMANN, K. ; HÜBSCH, G. ; SCHILL, A.: Overview of an End User enabled Model-driven Development Approach

for Interactive Applications based on Annotated Services. In: *In Proc. of the 4th Workshop on Emerging Web Services Technology (at ECOWS)*, 2009

- [FNJ⁺09b] FELDMANN, M. ; NESTLER, T. ; JUGEL, U. ; MUTHMANN, K. ; HÜBSCH, G. ; SCHILL, A.: Overview of an End User enabled Model-driven Development Approach for Interactive Applications based on Annotated Services. In: *Proceedings of the 4th Workshop on Emerging Web Services Technology (at ECWOS)*, ACM, 2009
- [Gar02] GARRETT, J. J.: *A visual vocabulary for describing information architecture and interaction design*. <http://www.jjg.net/ia/visvocab/>. März/2002
- [Gar03] GARRETT, J. J.: *The Elements of User Experience: User-Centered Design for the Web*. American Institute of Graphic Arts, 2003
- [HM00] HERMAN, I. ; MARSHALL, M.S.: Graph visualization and navigation in information visualization: A survey. In: *IEEE Transactions on Visualization and Computer Graphics*, 2000
- [Inc] INCORPORATED, Adobe S.: *Der Eigenschafteninspektor*. http://livedocs.adobe.com/de_DE/Dreamweaver/9.0/help.html?content=WScbb6b82af5544594822510a94ae8d65-7f9b.html
- [Ini09] INITIATIVE, W3C Mobile W.: *W3C Mobile Web Initiative: The Web on the Move*. <http://www.w3.org/Mobile/>. 2009
- [Jac] JACOBSEN, J.: *Sitemaps für die Site*. http://www.contentmanager.de/magazin/artikel_311_sitemaps_fuer_die_site.html
- [Jac03] JACOBSEN, J.: *Sitemaps in der Konzeption*. http://www.contentmanager.de/magazin/artikel_293_sitemaps_in_der_konzeption.html. 2003
- [Jac05] JACOBSEN, J.: *Website-Konzeption*. Addison-Wesley, 2005
- [JPS⁺09] JANEIRO, J. ; PREUSSNER, A. ; SPRINGER, T. ; SCHILL, A. ; WAUER, M.: Improving the Development of Service-based Applications through Service Annotations. In: *In Proc. of the IADIS International Conference WWW/Internet 2009, Rom, Italy*, 2009
- [Jug09] JUGEL, U.: ServFace Methodology Overview. In: *ServFace Deliverable 1.2*, 2009
- [KBS05] KRAIKER, S. ; BUTZ, A. P. ; SCHMIDT, A. D.: *Hierarchical Task Analysis*. http://www.medien.ifl.lmu.de/fileadmin/mimuc/mmi_ws0405/uebung/essays/sebastian.kraiker/sebastian.kraiker.html. 2005
- [KMA] KO, A.J. ; MYERS, B.A. ; AUNG, H.H.: Six Learning Barriers in End-User Programming Systems.

- [Ltd] LTD, Motive: *The Motive Web Design Glossary*. <http://www.motive.co.nz/glossary/navigation.php>
- [Ltd10] LTD, Volantis S.: *Volantis Mobility Server*. <http://www.volantis.com/>. 2010
- [MK06] MUSCIANO, C. ; KENNEDY, B.: *HTML. The Definitive Guide*. O' Reilly, 2006
- [Moo07] MOODY, D.: What makes a good diagram? improving the cognitive effectiveness of diagrams in is development. In: *Advances in Information Systems Development*, 2007, S. 481–492
- [Mro07] MROSEK, S.: *Hosting-Tipps.net*. http://www.hosting-tipps.net/glossar/glossar_w.htm. 2007
- [MS] MANAGEMENT, Information ; SOURCEMEDIA, Inc.: *Not Everyone Who Drives a Car Fixes It Themselves*. <http://www.information-management.com/news/1041222-1.html>
- [NDP09] NESTLER, T. ; DANNECKER, L. ; PURSCHE, A.: User-centric Composition of Service Front-ends at the Presentation Layer. In: *Proceedings of the 1st Workshop on User-generated Services (at ICSOC)*, 2009
- [NFH⁺10] NESTLER, T. ; FELDMANN, M. ; HÜBSCH, G. ; PREUSSNER, A. ; JUGEL, U.: *The ServFace Builder - A WYSIWYG approach for building Service-based Applications*. In *Proceedings of the Tenth International Conference on Web Engineering (ICWE)*, 2010
- [NFPS09] NESTLER, T. ; FELDMANN, M. ; PREUSSNER, A. ; SCHILL, A.: *Service Composition at the Presentation Layer using Web Service Annotations*. In *Proceedings of the First International Workshop on Lightweight Integration on the Web (at ICWE)*, 2009
- [NNA09] NAMOUNE, A. ; NESTLER, T. ; ANGELI, A. D.: End User Development of Service-based Applications. In: *In Proc. of the Second Workshop on HCI and Services (at HCI 2009)*, 2009
- [NNA10] NAMOUN, A. ; NESTLER, T. ; ANGELI, A. D.: *Conceptual and Usability Issues in the Composable Web of Software Services*. In *Proceedings of the Second International Workshop on Lightweight Integration on the Web (at ICWE)*, 2010
- [Pat10] PATH, Elastic: *Mobile Commerce Usability: Home Pages and Navigation*. <http://www.getelastic.com/mobile-home-page-navigation/>. 2010
- [PBw] PBWORKS: *MashupDefinition*. <http://mashup-tools.pbworks.com/MashupDefinition>
- [Sch] SCHIFFER, S.: *Visuelle Programmierung - Potential und Grenzen*. <http://www.swe.uni-linz.ac.at/people/schiffer/se-96-19/se-96-19.htm>

- [SM] SUN MICROSYSTEMS, Inc.: *Sun Web Application Guidelines - Version 4.0.* http://developers.sun.com/docs/web-app-guidelines/uispec4_0/05-navigation.html#5.1.1
- [sym10] SYMWEB, Internetagentur: *symweb - Erfolgsorientierte Internet-Lösungen: Internetlexikon Navigation - Website.* http://www.symweb.de/glossar/navigation-website_447.htm. 2010
- [Thi07] THIMBLEBY, H.: *Press On — Principles of Interaction Programming.* MIT Press, 2007. – ISBN 978-0-262-20170-4
- [Tuf] TUFTE, E. R.: *The Work of Edward Tufte and Graphics Press.* <http://www.edwardtufte.com/tufte/>
- [Tun99] TUNKELANG, D.: *A Numerical Optimization Approach to General Graph Drawing,* Citeseer, Diss., 1999
- [Vää92] VÄÄNÄNEN, K.: *Interfaces to Hypermedia: Communicating the Structure and Interaction Possibilities to the Users.* In: *Proceedings of 2nd Eurographics Workshop on Multimedia, Darmstadt, Mai 1992*
- [Web] WEBDEVELOPERSNOTES.COM, M. S.: *Web Developers Notes: Web Site Navigation.* http://www.webdevelopersnotes.com/tips/webdesign/web_site_navigation.php3
- [Wro] WROBLEWSKI, L.: *Ideation + Design.* <http://www.lukew.com/>