Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 2435

Bereitstellung eines bestehenden Frameworks für die Analyse von Eye-Tracking Daten als Web Service(s)

Dominik Weber

Studiengang:	Informatik
Prüfer/in:	Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Betreuer/in:	Dipl.-Inf., Dipl.-Wirt.Ing.(FH) Karolina Vukojevic-Haupt, Dipl.-Inf. Tanja Blascheck
Beginn am:	12. August 2013
Beendet am:	12. Februar 2014
CR-Nummer:	H.3.5, H.4.1, D.2.13

Kurzfassung

Im Zentrum aktueller Forschungsfragen auf dem Gebiet der Datenvisualisierung stehen mit zunehmendem Maße Eye-Tracking-Experimente, die verschiedene Visualisierungstechniken auf ihre Benutzerfreundlichkeit und Aufgabenangemessenheit hin untersuchen oder miteinander vergleichen. In Eye-Tracking-Benutzerstudien werden die Blickpunkte von Probanden beim Lösen einer Aufgabe aufgezeichnet und anschließend ausgewertet. Neben der eigentlichen Durchführung der Benutzerstudie ist vor allem die Analyse der gesammelten Eye-Tracking-Daten ein zeitintensiver Prozess.

Das Ziel dieser Studienarbeit ist, ein Konzept zu entwickeln, das einen Benutzer mit Hilfe der Workflow-Technologie bei der Analyse einer Eye-Tracking-Benutzerstudie unterstützt. Dabei sollen die einzelnen Eye-Tracking-Analyseschritte auf Workflow-Aktivitäten abgebildet werden. Hierzu wird zunächst ein bestehendes Eye-Tracking-Analyseframework in Bezug auf Funktionalität und Schnittstellen untersucht. Anschließend wird beschrieben, wie das Analyseframework erweitert oder angepasst werden muss, damit es mit Hilfe der Web Service-Technologie in BPEL-Workflows eingebunden werden kann. Das Konzept wird in Form eines prototypisch implementierten Web Service vorgestellt und anhand von zwei beispielhaften Szenarien demonstriert.

Abstract

In the field of data visualization eye tracking experiments are increasingly used to examine and compare different visualization techniques in regard to their usability and task appropriateness. In eye tracking user studies the gaze data of participants is recorded and evaluated subsequently. Apart from conducting the user study itself, the analysis of the collected eye tracking data is a time-intensive process.

The aim of this work is to develop a concept to assist a user with the analysis of eye tracking user studies by using workflow technology. To do so the steps of an eye tracking analysis should be mapped to workflow activities. First, an existing framework for the analysis of eye tracking user studies is examined in terms of its functionality and interface. Afterwards, it is shown how the framework has to be extended or adapted, in order to integrate it into a BPEL-workflow by using web service technology. The concept is presented in form of a prototypically implemented web service and demonstrated in two exemplary scenarios.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Aufbau der Arbeit . . . 1

2 Grundlagen **3**
2.1 Eye-Tracking . . . 3
2.1.1 Geschichte des Eye-Trackings . . . 3
2.1.2 Eye-Tracking Techniken und Eye-Tracking Systeme . . . 4
2.1.3 Datentypen . . . 7
2.1.4 Metriken . . . 7
2.1.5 Statistiken . . . 8
2.1.6 Visualisierungen . . . 8
2.1.7 Eye-Tracking-Benutzerstudien . . . 11
2.2 Web Service Technologie . . . 12
2.2.1 Service-Oriented Architecture . . . 12
2.2.2 Enterprise Service Bus . . . 13
2.2.3 Web Service Technologie . . . 13
2.2.4 Web Services . . . 13
2.2.5 Extensible Markup Language . . . 15
2.2.6 SOAP . . . 16
2.2.7 Universal Description, Discovery and Integration . . . 18
2.2.8 Web Services Description Language . . . 18
2.2.9 Integration von Anwendungen . . . 19
2.3 Workflow Technologie . . . 21
2.3.1 Geschäftsprozesse und Workflows . . . 21
2.3.2 Business Workflows . . . 22
2.3.3 Scientific Workflows . . . 22
2.3.4 Workflow Management Systeme . . . 23
2.3.5 Business Process Execution Language . . . 24

3 Verwendete Software **27**
3.1 eTaddy . . . 27
3.1.1 Benutzeroberfläche . . . 27
3.1.2 Auswertung einer Studie . . . 29
3.2 SimTech Scientific Workflow Management System . . . 30
3.2.1 Benutzeroberfläche . . . 30
3.2.2 Architektur . . . 31

4 Verwandte Arbeiten **33**
4.1 Ausführung von Festkörpersimulationen auf Basis der Workflow Technologie . 33
4.2 Generisches Web Service Interface um Simulationsanwendungen in BPEL-Prozesse einzubinden 33
4.3 Framework für die Visualisierung von Datenqualität in Simulation-Workflows 34

5 Aufgabe und Lösungsansatz **37**
5.1 Hintergründe 37
5.2 Aufgabenstellung 37
5.3 Lösungsansatz 38

6 Lösungskonzept **39**
6.1 Automatisierung des Analyseprozesses 39
6.2 Erstellen eines Web Services 40
6.3 Anforderungen an den Web Service 40
6.3.1 Datenverwaltung 40
6.3.2 Einheitliche Schnittstelle 41
6.3.3 Modularität und Wiederverwendbarkeit 41
6.4 Gesamtsystem 41

7 Implementierung **43**
7.1 Analyse von eTaddy 43
7.1.1 Architektur 43
7.1.2 Datenmodell 44
7.1.3 Plug-in-System 44
7.2 Web Service 47
7.2.1 Wahl des Web Service-Frameworks 47
7.2.2 Erstellen des Web Services 47
7.2.3 Struktur einer ASP.NET Webanwendung 47
7.2.4 Verbindung mit eTaddy 49
7.2.5 Datenverwaltung 49
7.2.6 Serialisierung der Parameter und Rückgabewerte 50
7.3 Bereitstellung der eTaddy-Plug-ins als Web Service-Operationen 51
7.3.1 Einheitliche Schnittstelle 52
7.3.2 Parameter 53
7.3.3 Metriken 53
7.3.4 Statistiken 53
7.3.5 Visualisierungen 54
7.3.6 Weitere Operationen 55
7.4 Modularität und Wiederverwendbarkeit 56
7.5 Fehlerbehandlung 56
7.6 Deployment 57

8 Demonstration **61**
8.1 Beispielszenario 1 61

8.2 Beispielszenario 2 61
8.3 Diskussion 63

9 Zusammenfassung und Ausblick **65**
9.1 Zusammenfassung 65
9.2 Ausblick 65

Literaturverzeichnis **67**

Abbildungsverzeichnis

2.1 Eye-Tracking-Aufnahmen von dem Bild „An unexpected return“ 5
2.2 Tobii T60XL Eye-Tracker 6
2.3 Heatmap-Visualisierung 9
2.4 Scanpath-Visualisierung 10
2.5 SOA-Dreieck 12
2.6 Enterprise Service Bus 13
2.7 SOA-Dreieck mit WSDL, SOAP und UDDI 14
2.8 SOAP-Nachricht 17
2.9 SOAP-Nachrichtenpfad 18
2.10 WSDL-Dokument 20
2.11 Prozesse und Workflows 21
2.12 Business Workflows und Scientific Workflows 23

3.1 Dialog zur Auswahl der Datenbank beim Start von eTaddy 28
3.2 Hauptfenster von eTaddy 28
3.3 Benutzeroberfläche des SimTech SWfMS 30
3.4 Architektur des SimTech SWfMS 31

4.1 Java Data Quality Visualization Framework 35

6.1 Lösungskonzept 42

7.1 eTaddy Solution Explorer 44
7.2 eTaddy Datenmodell 45
7.3 eTaddy Plug-ins 46
7.4 Hinzufügen eines neuen Projekts 48
7.5 Web Service Projekt 50
7.6 Nassi-Shneiderman-Diagramm der „GetFixationDuration“ WebMethod 54
7.7 Visualisierung der Pupillengröße im Verlauf der Zeit 56
7.8 Vergleich der alten und neuen Heatmap-Implementierung 57
7.9 Übersicht der Web Service-Operationen 58
7.10 Ablauf der Bearbeitung einer Web Service-Operation 59

8.1 Beispielszenario 1 62
8.2 Beispielszenario 2 63

9.1 WO- und WAS-Raum 66

Tabellenverzeichnis

2.1 Beispielhafte Auswahl von Metriken 8

3.1 Liste der implementierten Plug-ins in eTaddy 29

Verzeichnis der Listings

7.1 Eine annotierte C#-Klasse 49
7.2 Eine C#-Hilfsklasse, die mit XML-Annotationen versehen wurde 51
7.3 Eine beispielhafte Web Service-Implementierung 52
7.4 Ergebnis der Serialisierung 52
7.5 Instanziierung eines LINQ-Objekts 53

1 Einleitung

Eye-Tracking bezeichnet den Prozess Augenposition und -bewegungen zu verfolgen und aufzuzeichnen. Bereits Ende des 19. Jahrhunderts gab es erste Forschungen im Bereich des Eye-Trackings, als Louis Émile Javal die Augenbewegungen beim Lesen eines Textes beobachtete [17]. Mit der Verbreitung von Personal Computern in den 80er Jahren wurde Eye-Tracking auf die Mensch-Computer-Interaktion angewendet. Übliche Aufgaben umfassten dabei das Suchen nach Einträgen in Programmmenüs. Heutzutage ist die Untersuchung von Benutzeroberflächen und Webseiten durch Eye-Tracking eine wichtige Technik, um die Benutzbarkeit zu verbessern.

Im Zentrum aktueller Forschungsfragen auf dem Gebiet der Datenvisualisierung stehen mit zunehmendem Maße Eye-Tracking-Experimente, die verschiedene Visualisierungstechniken auf ihre Benutzerfreundlichkeit und Aufgabenangemessenheit hin untersuchen oder miteinander vergleichen. In Eye-Tracking-Benutzerstudien werden die Blickpunkte von Probanden beim Lösen einer Aufgabe aufgezeichnet und anschließend ausgewertet. Neben der eigentlichen Durchführung der Benutzerstudie ist vor allem die Analyse der gesammelten Eye-Tracking-Daten ein zeitintensiver Prozess. Bereits nach wenigen Minuten fallen bei einer Eye-Tracking-Benutzerstudie viele Datensätze an. Das Ziel der Analyse ist, die Suchstrategien der Probanden beim Betrachten der Visualisierungen zu vergleichen. Die aufgezeichneten Blickpunkte werden daher anhand von Metriken, Statistiken und Visualisierungen ausgewertet.

In dieser Studienarbeit soll ein Konzept vorgestellt werden, das einen Benutzer mit der Workflow-Technologie bei der Analyse einer Eye-Tracking-Benutzerstudie unterstützt. Dabei sollen die einzelnen Eye-Tracking-Analyseschritte auf Workflow-Aktivitäten abgebildet werden. Hierzu soll zunächst ein bestehendes Eye-Tracking-Analyseframework in Bezug auf Funktionalität und Schnittstellen untersucht werden. Anschließend soll festgestellt werden, wie das Analyseframework erweitert oder angepasst werden muss, so dass es mit Hilfe der Web Service-Technologie in BPEL-Workflows eingebunden werden kann. Dabei sollen vor allem die Modularität und Wiederverwendbarkeit der Web Service Operationen betrachtet werden.

1.1 Aufbau der Arbeit

Diese Studienarbeit ist insgesamt in neun Kapitel gegliedert. In diesem Kapitel wurde das bearbeitete Thema eingeleitet und die Motivation der Arbeit vorgestellt. Anschließend werden in Kapitel 2 die Grundlagen der Themengebiete Eye-Tracking, Web Service Technologie und Workflow Technologie eingeführt. Für jedes der drei Themengebiete werden die Bergriffe, Konzepte und Techniken vorgestellt, die wichtig für das Verständnis dieser Arbeit sind. In Kapitel 3 wird die verwendete Software vorgestellt, welche die Grundlage für prototypische

Implementierung des Konzepts bildet. Kapitel 4 stellt kurz drei Diplomarbeiten vor, die der Aufgabenstellung dieser Arbeit ähneln und zeigt Ähnlichkeiten und Unterschiede auf. In Kapitel 5 werden die Hintergründe und die konkrete Aufgabenstellung beschrieben. Des Weiteren wird das Lösungskonzept vorgestellt. Darauf aufbauend wird in Kapitel 6 das Lösungskonzept ausführlich erläutert. In Kapitel 7 wird die prototypische Implementierung des Lösungskonzepts behandelt. Hier werden zunächst die notwendigen Vorbereitungen diskutiert und anschließend die konkrete Implementierung mit den aufgetretenen Herausforderungen beschrieben. Die Funktionalität des Prototyps wird in Kapitel 8 anhand von zwei Beispielszenarien gezeigt. Abschließend werden in Kapitel 9 die Konzepte und Ergebnisse der Studienarbeit zusammengefasst. Außerdem wird ein kurzer Ausblick gegeben, wie das Lösungskonzept weiterentwickelt werden kann.

2 Grundlagen

Dieses Kapitel führt in die Themen dieser Studienarbeit ein. Das Kapitel ist in die drei Unterkapitel Eye-Tracking (Abschnitt 2.1), Web Service Technologie (Abschnitt 2.2) und Workflow Technologie (Abschnitt 2.3) unterteilt. Für jedes der drei Themengebiete werden die Bergriffe, Konzepte und Techniken vorgestellt, die wichtig für das Verständnis dieser Arbeit sind.

2.1 Eye-Tracking

Eye-Tracking bezeichnet den Prozess Augenposition und -bewegungen zu verfolgen und aufzuzeichnen. Ein Eye-Tracker ist ein Gerät, das den Mensch bei der Aufzeichnung und der automatischen Analyse von Blickpunkten unterstützt.

Einsatzmöglichkeiten für Eye-Tracking sind unter anderem in der Marktforschung, die Evaluation von Benutzeroberflächen, die Mensch-Computer-Interaktion sowie psychologische und neurologische Forschung. Als Grundlage dafür dient die **Eye-Mind-Hypothese**, die besagt, dass sich das Sehen und die kognitive Verarbeitung des Gesehenen gegenseitig beeinflussen [18]. Der Vorgang des Sehens wird daher auch als intentional bezeichnet, da es eine aktive und bewusste Handlung des Sehenden darstellt.

In diesem Kapitel wird die Geschichte des Eye-Trackings vorgestellt. Anschließend wird ein kurzer Überblick über eine Auswahl von Eye-Tracking-Techniken und -Systeme gegeben. Danach werden wichtige Datentypen, Metriken, Statistiken und Visualisierungen eingeführt. Abschließend wird der Ablauf von Eye-Tracking-Benutzerstudien und deren Auswertung beschrieben.

2.1.1 Geschichte des Eye-Trackings

In diesem Abschnitt wird ein Überblick über die Geschichte des Eye-Trackings gegeben. Der Abschnitt basiert, soweit nicht anders angegeben, auf „Eye Tracking in Human–Computer Interaction and Usability Research: Ready to Deliver the Promises“ von Robert J. K. Jacob und Keith S. Karn [16].

Bereits Ende des 19. Jahrhunderts gab es erste Forschungen im Bereich des Eye-Trackings. Louis Émile Javal beobachtete im Jahr 1878 die Augenbewegungen beim Lesen eines Textes. Dabei stellte er fest, dass Probanden nach etwa zehn Zeichen eine Pause einlegen. Die frühen Eye-Tracking Methoden waren noch invasiv und erforderten direkten Kontakt mit der Hornhaut [17]. Dodge und Cline entwickelten 1901 die erste genaue nicht-invasive Technik, bei der das von der Hornhaut reflektierte Licht beobachtet wurde. Allerdings konnte nur die horizontale Augenposition aufgezeichnet werden und die Probanden durften den Kopf nicht bewegen [6].

Im Jahr 1930 stellten Miles Tinker und seine Kollegen weitere Untersuchungen über das Leseverhalten an. Sie variierten die Schriftart, Schriftgröße und das Seitenlayout, um deren Einfluss auf die Lesegeschwindigkeit zu messen und Muster in den Augenbewegungen festzustellen [33].

Die erste Anwendung von Eye-Tracking im Bereich Usability Engineering, also der systematischen Untersuchung der Benutzerinteraktion mit Produkten, um das Produktdesign zu verbessern, wurde 1947 von Fitts, Jones und Milton eingesetzt. Sie benutzten Filmkameras, um die Augenbewegungen von Piloten bei der Bedienung der Instrumente im Cockpit während einer Landung aufzuzeichnen [9].

Ein Jahr später, 1948, entwickelten Hartridge und Thompson den ersten Head-Mounted Eye-Tracker. Dadurch war es erstmals möglich Eye-Tracking-Benutzerstudien ohne ein stationäres System durchzuführen [11].

Weitere Forschung im Bereich des Leseverhaltens wurde 1968 von Edmund Huey in seinem Buch „The psychology and pedagogy of reading“ vorgestellt. Er stellte fest, dass ein Mensch Textzeilen in vielen kurzen Bewegungen von links nach rechts liest. Am Ende der Zeile werden die Augen in einer einzigen Bewegung vom Zeilenende zum Beginn der neuen Zeile bewegt [15].

Alfred L. Yarbus untersuchte im Jahr 1967 die Augenbewegungen von Probanden bei Betrachtung des Werkes „An unexpected return“ von Ilya Repin (siehe Abbildung 2.1). Die Probanden sollten mehrere Fragen beantworten und verschiedene Aufgaben durchführen. Yarbus stellte fest, dass es keine eindeutige Strategie gibt, um ein Bild zu betrachten, vielmehr sind die Augenbewegungen abhängig von der gestellten Aufgabe [35].

In den 70er Jahren wurden erste rudimentäre theoretische Modelle von Eye-Tracking Daten in Verbindung mit kognitiven Prozessen untersucht. Es wurde ein Zusammenhang zwischen Augenbewegungen und der Aufmerksamkeit, sowie von Fixationen (siehe Abschnitt 2.1.3) und bestimmten kognitiven Prozessen gesucht.

Im Jahr 1975 sagte Monty „Es ist nicht ungewöhnlich mehrere Tage für die Auswertung von Daten zu benötigen, die in wenigen Minuten gesammelt wurden“ [23]. Dies war einerseits bedingt durch die Technik, so musste eine Filmaufnahme erst entwickelt werden, um sie auszuwerten. Andererseits ist die Auswertung von Eye-Tracking Daten auch heute noch ein zeitintensiver Vorgang, da viele Schritte noch nicht automatisiert werden können und eine Bewertung von einem Menschen benötigen.

Mit der Verbreitung von Personal Computern in den 80er Jahren wurde Eye-Tracking auf die Mensch-Computer-Interaktion angewendet. Übliche Aufgaben umfassen dabei das Suchen nach Einträgen in Programmmenüs. Heutzutage ist die Untersuchung von Benutzeroberflächen und Webseiten durch Eye-Tracking eine wichtige Technik, um die Benutzbarkeit zu verbessern. Außerdem steht inzwischen genug Rechenleistung zur Verfügung, um eine Echtzeitanalyse mit Eye-Tracking Systemen durchzuführen. Dadurch ist es möglich einen Eye-Tracker als ein Eingabegerät zu verwenden.

2.1.2 Eye-Tracking Techniken und Eye-Tracking Systeme

Welche Eye-Tracking Technik und welches Eye-Tracking System bei einer Benutzerstudie eingesetzt wird, ist von der gestellten Aufgabe abhängig. Beim Betrachten von Visualisierungen, Webseiten oder Benutzeroberflächen kann ein stationäres System eingesetzt werden, das dem

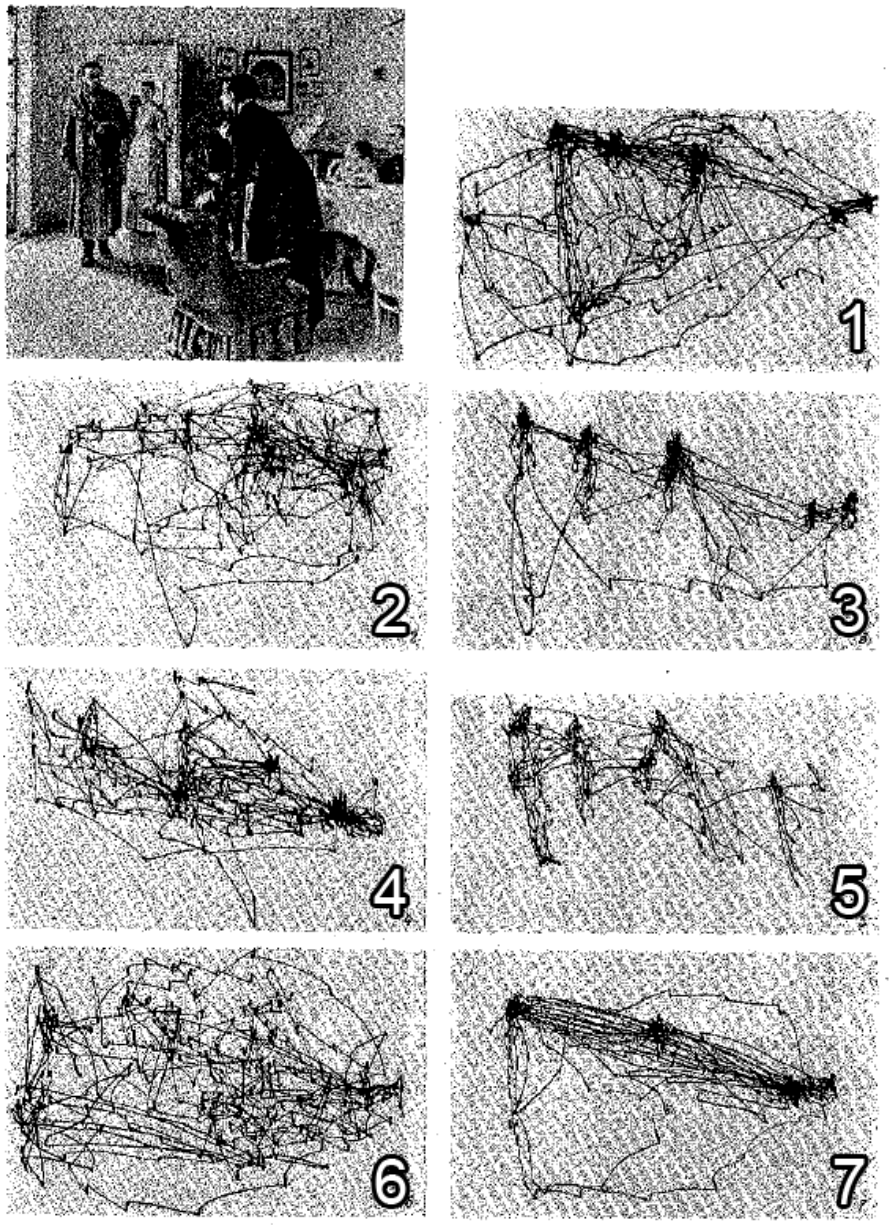

Abbildung 2.1: In dieser Abbildung sind sieben Aufnahmen des selben Probanden bei der Betrachtung des Bildes „An unexpected return“ zu sehen. Für jede Aufnahme hat der Proband eine andere Aufgabe bekommen, die jeweils drei Minuten angedauert haben: 1) Freie Betrachtung des Bildes. 2) Die materiellen Umstände schätzen. 3) Das Alter der Personen angeben. 4) Mutmaßen, was die Familie gemacht hat, bevor der „unerwartete Besucher“ aufgetaucht ist. 5) Die Kleidung der Personen merken. 6) Die Position der Objekte und Personen um Raum merken. 7) Abschätzen, wie lange der „unerwartete Besucher“ von der Familie fort war. [35]

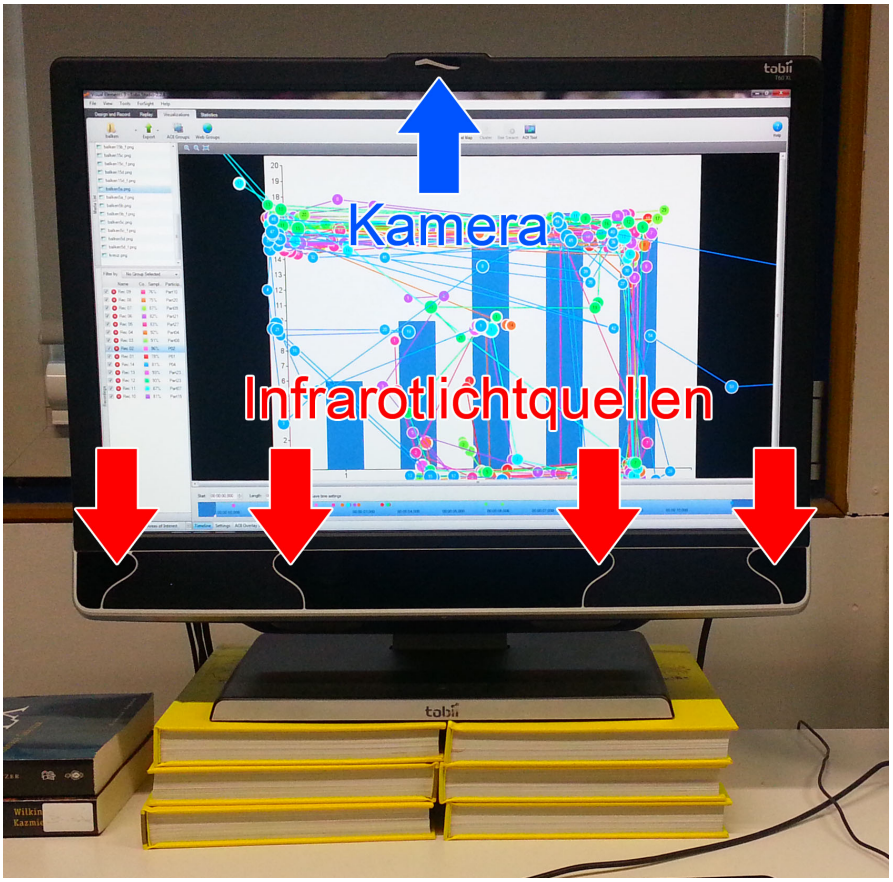

Abbildung 2.2: Bild eines Tobii T60XL Eye-Trackers. Pfeil oben: Position der Kamera. Pfeile unten: Positionen der vier Infrarotlichtquellen.

Probanden erlaubt, vor dem Eye-Tracker wie vor einem Computer zu sitzen. In Abbildung 2.2 ist ein Tobii T60XL Eye-Tracker[1] dargestellt, der in einen Computermonitor eingebaut ist. Bei diesem System wird die Technik der Videookulographie eingesetzt. Dabei wird vom Eye-Tracker Infrarotlicht ausgestrahlt, das von der Hornhaut des Probanden reflektiert und anschließend von einer Kamera aufgenommen wird. Zusätzlich wird die Ausrichtung der Pupillen berechnet. Ein bekanntes Verfahren hierfür ist der Starburst-Algorithmus [22]. Die dabei berechnete Position der Augen wird dem auf dem Bildschirm angezeigten Stimulus zugeordnet. Andrew Duchowski stellt in seinem Buch „Eye Tracking Methodology“ [8] neben der Videookulographie noch weitere Techniken vor. Böhme et al. haben eine Übersicht über aktuelle Techniken zusammengestellt [3].

Neben den stationären Systemen gibt es auch Head-Mounted Eye-Tracker, die es den Probanden erlauben sich frei im Raum zu bewegen. Dies ist beispielsweise nützlich, um das Kaufverhalten in einem Supermarkt ohne Einschränkungen zu untersuchen.

[1] http://www.tobii.com/Global/Analysis/Marketing/Brochures/ProductBrochures/Tobii_T60XL_Brochure_AcademicResearch.pdf Offizielle Produktbeschreibung des Tobii T60XL Eye-Trackers

2.1.3 Datentypen

In diesem Abschnitt werden die Datentypen einer Eye-Tracking-Benutzerstudie erklärt. Es werden die Begriffe Stimulus, Fixation, Area of Interest und Sakkade behandelt.

Stimulus

Bei einem Stimulus handelt es sich um das Objekt, das bei einer Eye-Tracking Studie betrachtet wird. Es kann sich dabei um ein statisches Bild, ein Video oder interaktive Objekte wie Webseiten oder Benutzeroberflächen von Programmen handeln.

Fixation

Bei Fixationen handelt es sich um Augenbewegungen, die sich auf einen Punkt konzentrieren [8] und der Blick auf einem Objekt von Interesse gehalten wird. Dabei steht das Augen nicht still. Vielmehr führt es mehr oder weniger zufällige Miniaturaugenbewegungen durch, die Tremor, Drift und Mikrosakkade genannt werden. Nach Holmqvist et al. [12] dauert eine Fixation etwa 200-300 Millisekunden an.

Sakkade

Eine Sakkade ist eine Serie von schnellen Augenbewegungen zwischen Fixationen. Sakkaden dauern zwischen 10 und 100 Millisekunden. Während dem Übergang nimmt das Gehirn keine Informationen auf [29]. Sakkaden sind die Manifestation des Wunsches die Aufmerksamkeit auf ein anderes Objekt zu fokussieren [8].

Area of Interest

Eine Area of Interest (AOI) definiert interessante Regionen auf dem Stimulus, über die Daten gesammelt werden sollen. So können Objekte auf dem Stimulus als AOIs markiert und Fixationen den AOIs zugeordnet werden. Der Besuch einer AOI, vom Eintritt des Blickes bis zum Verlassen, wird Dwell, Gaze oder Glance genannt. Interessant ist auch die Betrachtung der Übergänge von einer AOI zu einer anderen, da so beispielsweise verschiedene Suchstrategien beim Lösen einer Aufgabe verglichen werden können. Liegt eine Fixation in einer AOI, spricht man von einem AOI Treffer. Durch die Zuordnung von Fixationen in AOIs es möglich die aufgenommenen Daten zu transformieren und zu vereinfachen. Der Verlauf der besuchten AOIs kann als String, Übergangsmatrix oder Graph repräsentiert werden [12].

2.1.4 Metriken

Um den Erfolg eines Probanden beim Lösen einer Eye-Tracking Aufgabe bewerten zu können, müssen Metriken definiert werden. Diese basieren beispielsweise auf den Fixationen oder den Sakkaden. In der folgenden Tabelle werden die Metriken erklärt, die für das Verständnis dieser Arbeit notwendig sind. Diese und weitere Metriken wurden von Poole et al. [24] ausführlich definiert.

Metrik	Beschreibung	Einheit
Bearbeitungszeit	Die Zeit, die ein Proband benötigt, um eine Aufgabe zu bearbeiten. Vom Erscheinen der Aufgabe bis zur Lösung.	ms
Fixationsanzahl	Die Summe der Fixationen für einen Stimulus.	#
Fixationsdauer	Die Dauer der einzelnen Fixationen.	ms
Pupillengröße	Die Größenveränderung der Pupillen des Probanden beim Betrachten des Stimulus.	mm
Sakkadenanzahl	Die Summe der Sakkaden für einen Stimulus.	#
Sakkadendauer	Die Dauer der einzelnen Sakkaden.	ms

Tabelle 2.1: Eine beispielhafte Auswahl von Metriken. Diese und weitere Metriken wurden von Poole et al. ausführlich definiert [24].

2.1.5 Statistiken

Auf die Ergebnisse der Metriken können verschiedene Statistiken angewendet werden. Zum Beispiel kann für alle Probanden die durchschnittliche Anzahl von Fixationen für einen bestimmten Stimulus berechnet werden. Zu den typischen Statistiken gehört der Durchschnitt, die Berechnung des Minimums, des Medians und des Maximums, sowie der Standardabweichung. Auch weitere Statistiken, wie der t-Test, sind möglich.

2.1.6 Visualisierungen

Um die bei einer Eye-Tracking Aufnahme entstandenen Daten zu visualisieren, werden üblicherweise die im folgenden Abschnitt beschriebenen Techniken Heatmap und Scanpath verwendet. Als Grundlage für die Visualisierungen werden die Fixationen und Sakkaden, oder die AOIs und die AOI-Übergänge verwendet.

Heatmap

Eine Heatmap wird über den Stimulus gelegt und zeigt eine zeitaggregierte Verteilung der Fixationen eines oder mehrerer Probanden. Dafür wird der Stimulus in ein Raster eingeteilt. Die Fixationen fallen in das Raster und werden aufsummiert und normiert. Den Rasterzellen wird mit Hilfe einer Farbskala eine Farbe abhängig vom aufsummierten Wert zugewiesen. Oft wird die Farbpalette[2] so gewählt, dass niedrigen Werten eine kalte und hohen Werten eine warme Farbe zugeteilt werden.

Häufig werden Heatmaps mit einem Gauß-Filter geglättet. Dies kann vor oder nach dem Zuweisen von Farben geschehen und sorgt für glatte Übergänge zwischen den Farben.

Eine Heatmap kann dazu genutzt werden AOIs im Nachhinein zu definieren, indem Regionen mit einer hohen Fixationsdichte ausgewählt werden. In Abbildung 2.3 ist eine beispielhafte Heatmap zu sehen.

[2] `http://colorbrewer2.org/` bietet eine Vielzahl von Farbpaletten, die sich für Heatmaps eignen.

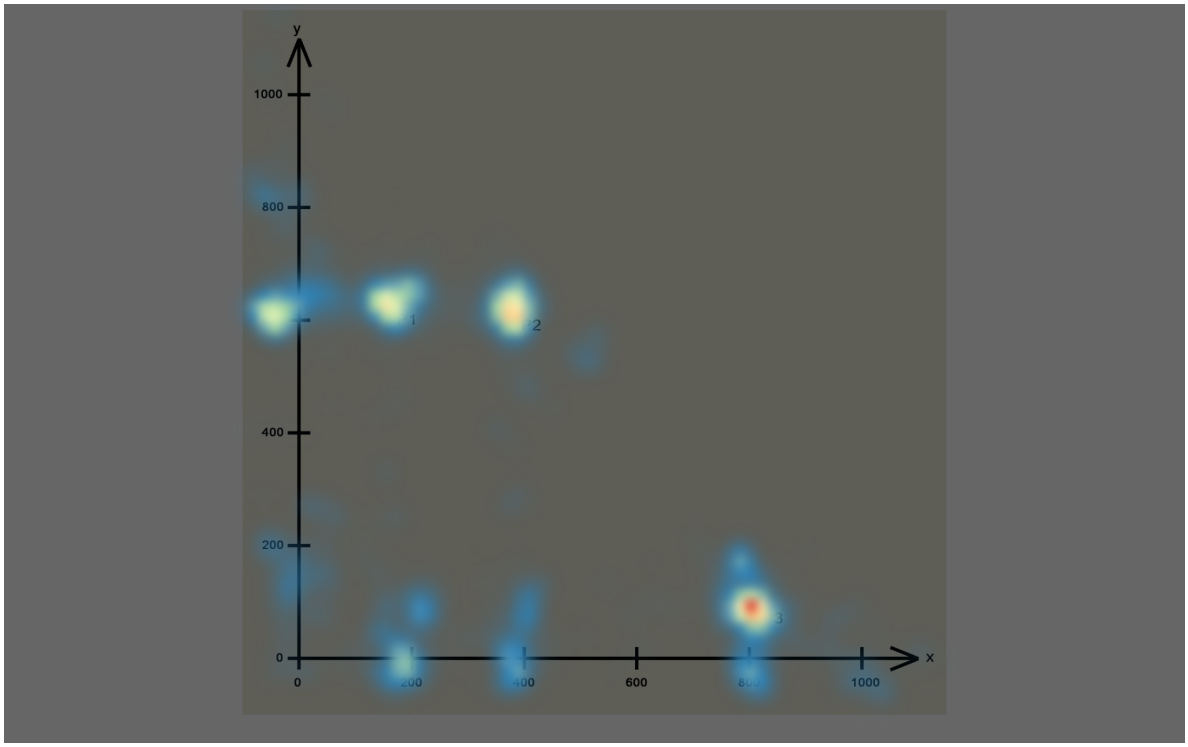

Abbildung 2.3: Eine Heatmap aus einer Eye-Tracking-Benutzerstudie, in der die Fixationen von mehreren Probanden eingeflossen sind. Der Stimulus ist ein Koordinatensystem mit drei Punkten P1, P2 und P3. Als Farbskala wurde ein Verlauf von Blau (wenige Fixationen), über Gelb und Rot (viele Fixationen) gewählt. Durch die Heatmap ist leicht zu sehen, dass die drei Punkte und die Achsenbeschriftungen häufig fixiert wurden. Die meisten Fixationen liegen bei Punkt P3, zu erkennen an der roten Färbung.

Scanpath

Ein Scanpath visualisiert die Abfolge von Fixationen und Sakkaden eines Probanden. Dadurch ist es möglich, sich einen Überblick über die Abfolge von Augenbewegungen zu verschaffen [31]. Ein Scanpath wird über den Stimulus gezeichnet. Fixationen werden als Kreise dargestellt und Sakkaden als Verbindungslinien zwischen den Kreisen. Die Fixationsdauer kann durch Anpassen des Kreisdurchmessers visualisiert werden. Mit Hilfe eines Scanpaths können AOIs definiert werden, in dem der Scanpath beispielsweise auf Regionen mit vielen Fixationen oder Übergängen untersucht wird. Auch ist das Erstellen eines AOI-Scanpath möglich, hierbei werden nicht Fixationen und Sakkaden visualisiert, sondern AOIs und die Übergänge zwischen AOIs.

Ein Nachteil von Scanpaths ist, dass der Überblick schnell verloren gehen kann. Durch eine Vielzahl von Fixationen und den daraus entstehenden Kantenkreuzungen wird der Scanpath schnell unübersichtlich. Wenn ein Überschuss von Elementen, deren Darstellung oder Anordnung zu einer Verschlechterung der Leistung beim Lösen einer Aufgabe führt,

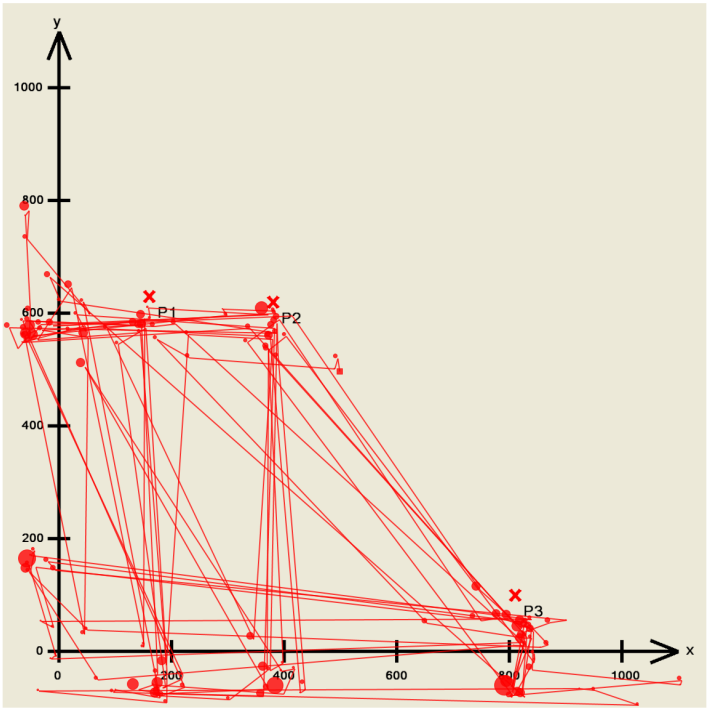

Abbildung 2.4: Ein Scanpath aus einer Eye-Tracking-Benutzerstudie, der die Fixationen und Sakkaden eines Probanden visualisiert. Der Stimulus ist ein Koordinatensystem mit drei Punkten P1, P2 und P3. Die Fixationen werden als Kreise und die Sakkaden als Verbindungslinien zwischen den Fixationen dargestellt. Die Fixationsdauer der einzelnen Fixationen ist im Durchmesser der Kreise kodiert.

spricht man von **Visual Clutter** [26]. Das Übereinanderzeichnen von Scanpaths mehrerer Probanden ist daher meist nicht zum empfehlen. Oft ist es noch möglich Muster zu erkennen, aber die genaue Abfolge der Augenbewegungen ist dann nicht mehr ersichtlich. Um dem Visual Clutter entgegenzuwirken, kann der Scanpath animiert werden, dabei wird nur noch ein Ausschnitt der Fixationen und Sakkaden dargestellt. Allerdings ist dann keine Aussage über die gesamte Abfolge mehr möglich.

Weitere Visualisierungen

Eine Verbesserung gegenüber von Scanpaths ist die Parallel Scan-Paths (PSP) Visualisierung, die auf AOIs basiert [25]. Dabei werden die Probleme der Kantenkreuzungen und Überlagerung von Fixationen vermieden, indem der zeitliche Verlauf auf die vertikale Achse und die betrachteten AOIs auf die horizontale Achse aufgetragen werden.

Ein Beispiel für eine videobasierte Visualisierung ist die Bee-Swarm-Visualisierung. Dabei werden die Fixationen von mehreren Probanden direkt auf dem Stimulus als Kreis oder Punkt abhängig vom Zeitstempel eingezeichnet. So ist es direkt ersichtlich, welchen Teil des

Stimulus die Probanden zu einem bestimmten Zeitpunkt angeschaut haben. Dies ist hilfreich, um Interessengebiete ausfindig zu machen.

2.1.7 Eye-Tracking-Benutzerstudien

Eye-Tracking hilft Suchstrategien zu verstehen und ermöglicht es beispielsweise verschiedene Visualisierungen oder Benutzeroberflächen zu vergleichen. Am Anfang einer Eye-Tracking-Benutzerstudie steht zunächst eine Hypothese, die durch die Studie verifiziert oder falsifiziert werden soll. Eine Hypothese ist eine „vermutete Antwort auf eine Frage, d.h. eine beliebige Aussage, die man provisorisch für bestimmte Zwecke als wahr annimmt, auch wenn man nicht weiß, ob sie wirklich wahr ist oder nicht“ [14]. Wie eine Hypothese formuliert werden kann, beschreibt Andrew Duchowski in „Eye Tracking Methodology“ [8].

Aus der Hypothese müssen die zu sammelnden Daten abgeleitet werden. Bei einer Eye-Tracking-Benutzerstudie gehören dazu zum einen messbare Daten, wie die Blickpunkte und den daraus berechneten Fixationen und Sakkaden. Zum anderen Daten, die in der Form von Fragebögen erfasst werden.

Bei der eigentlichen Durchführung der Studie unterschreibt der Proband zunächst eine Einverständniserklärung. Anschließend sollte ein Seh- und Farbtest durchgeführt werden. Leidet der Proband unter eine Sehschwäche oder Farbfehlsichtigkeit, muss der Datensatz von der Auswertung ausgenommen werden. Danach werden Daten wie das Alter, Geschlecht, Nationalität und Bildungsstand in Form von einem Fragebogen aufgenommen. Vor der Bearbeitung der Aufgaben muss zunächst der Eye-Tracker kalibriert werden, da die Sehachse bei jedem Menschen unterschiedlich ausgerichtet ist [7]. Eine mögliche Art der Kalibrierung ist die Betrachtung von fest vorgegebenen Punkten auf dem Bildschirm. Zwischen den Aufgaben oder nach der erfolgreichen Bearbeitung der Aufgabe können mit Hilfe von Fragebögen die behandelten Aufgaben bewertet werden. Dies kann beispielsweise mit dem Nasa TLX geschehen [10].

Für die Analyse ist es in einem Vorverarbeitungsschritt notwendig, die gesammelten Daten auf Fehler zu überprüfen. Anschließend können abhängig von der Hypothese und den daraus entstandenen Aufgaben die Daten anhand von Metriken, Statistiken und Visualisierungen analysiert werden.

2.2 Web Service Technologie

In diesem Abschnitt werden die Grundlagen der Web Service Technologie behandelt. Zunächst wird das Prinzip der Service Oriented Architecture vorgestellt. Anschließend wird die Web Service Technologie als eine Implementierung dieses Architekturstils eingeführt. Dazu gehören die Themen Extensible Markup Language, SOAP, Universal Description, Discovery and Integration und die Web Service Description Language. Abschließend werden verschiedene Möglichkeiten, um eine bereits vorhandene Anwendung zu integrieren vorgestellt. Der Abschnitt basiert, sofern nicht weiter angegeben, auf dem Buch „Web Services Platform Architecture“ [34].

2.2.1 Service-Oriented Architecture

Die Service-Oriented Architecture (SOA) ist ein abstrakter Architekturstil, der sich mit der losen Kopplung und dynamischen Bindung zwischen Services befasst. Ein Service ist eine in sich abgeschlossene Einheit von Funktionalität. Durch die Wiederverwendung und Kombination von Services kann die Flexibilität der Gesamtarchitektur erhöht werden. Services können anhand ihrer Metadaten beschrieben und dynamisch ausgewählt werden. Um einen Service einsetzen zu können, ist nur wenig oder kein Wissen über die Implementierung notwendig. Die Funktionen eines Services können über verschiedene Transportwege zur Verfügung gestellt werden.

Die zugrundeliegenden Prinzipien von einer SOA werden im SOA-Dreieck (siehe Abbildung 2.5) beschrieben. Um einen Service auffindbar (*Find*) zu machen, muss eine abstrakte Definition des Services mit allen notwendigen Informationen zum Aufruf bekannt sein. Diese wird vom Serviceanbieter (*Service Provider*) in einem Serviceverzeichnis (*Discovery Facility*) hinterlegt (*Publish*). Ein Servicenehmer (*Service Requestor*) stellt eine Anfrage an ein Serviceverzeichnis und erhält daraufhin eine Liste mit verfügbaren Services. Aus dieser Liste wird ein Service für den Aufruf ausgewählt. Dies kann beispielsweise durch eine zufällige Auswahl geschehen, allerdings ist auch die Hinzunahme von weiteren Kriterien möglich. Nach der Auswahl bindet (*Bind*) der Servicenehmer den Service an sich.

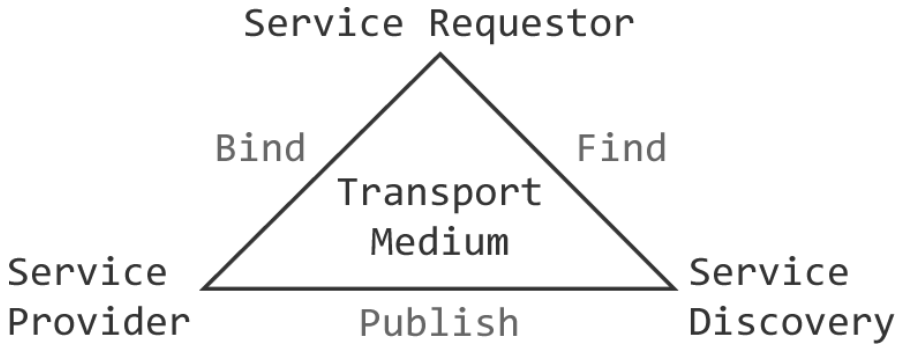

Abbildung 2.5: Das SOA-Dreieck zeigt das Zusammenspiel vom Service Requestor, Service Provider und der Service Discovery [34].

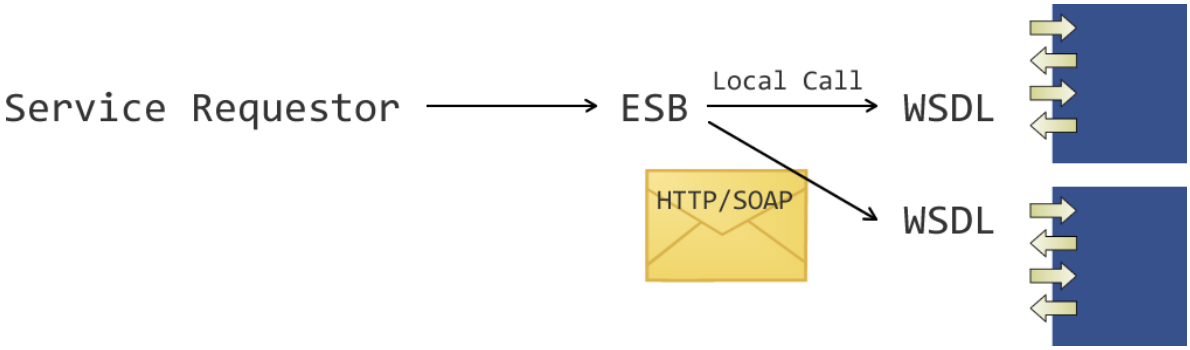

Abbildung 2.6: Der Enterprise Service Bus (ESB) versteckt Zugriffsmechanismen, wie das Transportprotokoll und das Encoding. Er kümmert sich um um die Entdeckung, die Auswahl und den Aufruf von Services anstelle des Servicenehmers.

2.2.2 Enterprise Service Bus

Um den Prozess der Serviceauswahl zu vereinfachen kann eine Middleware, der Enterprise Service Bus (ESB), verwendet werden. Dieser erhält die Anfrage vom Servicenehmer und kümmert sich an dessen Stelle um die Entdeckung, die Auswahl und den Aufruf von Services. Dabei werden Details wie die Zugriffsmechanismen (Transportprotokoll, Encoding, Endpoint), die Programmiersprache und der Architekturstil versteckt (siehe Abbildung 2.6). Außerdem stellt der ESB sicher, dass die nichtfunktionalen Eigenschaften des Aufrufs eingehalten werden.

2.2.3 Web Service Technologie

Die Web Service Technologie ist eine auf Standards beruhende XML-basierte *Umsetzung* des Service-orientierten Architekturstils (SOA) [34]. Dabei spielt der Enterprise Service Bus eine zentrale Rolle. Die Web Service Technologie stellt ein Modell für das Verwenden von Komponenten in einer lose gekoppelten Weise zur Verfügung. Allerdings wird keine Aussage darüber getroffen, wie die einzelnen Komponenten zu implementieren sind. Die Eigenschaften der Middleware werden versteckt .

2.2.4 Web Services

Ein Web Service ist eine identifizierbare Einheit von Funktionalität, die über das Web erreichbar sein kann (aber nicht muss) und eine einheitliche Schnittstelle zur Verfügung stellt. Er ist in sich geschlossen, auffindbar, kombinierbar und immer verfügbar. Mit Hilfe der Web Service Description Language (siehe Abschnitt 2.2.8) wird beschrieben, wie auf ein Web Service zugegriffen werden kann. Um einen passenden Web Service zu finden kann Universal Description Discovery and Integration (siehe Abschnitt 2.2.7) benützt werden. Der Aufruf eines Web Services erfolgt mit SOAP (siehe Abschnitt 2.2.6).

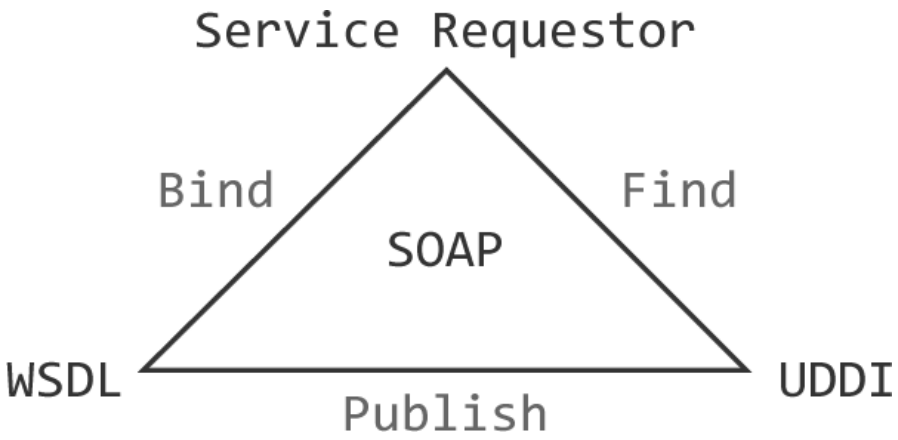

Abbildung 2.7: Das SOA-Dreieck mit WSDL, SOAP und UDDI.

Ein Web Service ermöglicht die Virtualisierung von Funktionalität in Form von Services. Diese sind dabei unabhängig von der Implementierung und der Plattform. Durch diese lose Kopplung ist es möglich die *Funktionalität* und nicht die *Programmierung* zu verwenden.
Das W3C definiert Web Services wie folgt[3]:

> „Ein Web Service ist ein Softwaresystem, das dafür ausgelegt wurde, um Interaktionen zwischen Maschinen über ein Netzwerk zu ermöglichen. Die Schnittstelle wird in einem maschinenlesbaren Format (WSDL) beschrieben. Andere Systeme interagieren mit dem Web Service in einer Art und Weise, die anhand von SOAP Nachrichten vorgeschrieben werden. In der Regel werden die SOAP Nachrichten mit HTTP in einer XML-Serialisierung und in Verbindung mit anderen Web-bezogenen Standards übertragen.“

Aus WSDL kann automatisch Client-side Code generiert werden (zum Beispiel für Java und .NET Frameworks). Ein Web Service kann Top-Down oder Bottom-Up erstellt werden.

Bottom-Up

Der Bottom-Up Ansatz zur Erstellung von Web Services wird eingesetzt, wenn ein bereits existierendes Programm als Web Service zur Verfügung gestellt werden soll. Das bestehende Programm wird mit Annotationen versehen. Anhand der Annotationen und der Eingabe- und Rückgabeparameter der Methoden können die WSDL Datentypen, Operationen und Nachrichten generiert werden. Dieser Ansatz wurde in der Implementierung dieser Arbeit eingesetzt.

Top-Down

Soll ein Web Service auf der Basis einer bestehenden WSDL Datei generiert werden, wird der Top-Down Ansatz eingesetzt. Durch die in der WSDL Datei enthaltenen Datentypen und

[3] http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice Zitat übersetzt aus dem Englischen

Operationen mit Ein- und Ausgabeparametern können Klassenskelette generiert werden. In diesen muss anschließend noch die Anwendungslogik implementiert werden.

2.2.5 Extensible Markup Language

Die Extensible Markup Language (XML) [36] ist eine Auszeichnungssprache für hierarchisch strukturierte Daten. Sie eignet sich für den plattform- und implementierungsunabhängigen Austausch von Daten und ist dabei für Menschen und Computer verständlich.

Ein XML-Dokument wird mit der XML Deklarierung eingeleitet. Diese legt die XML Version und die Zeichenkodierung fest (`<?xml version="1.0" encoding="UTF-8"?>`). Daraufhin folgt genau ein Wurzelelement, das weitere Elemente enthalten kann. Ein Element besteht aus einem Start-Tag (`<beispiel>`), gefolgt von einem optionalen Inhalt und schließlich dem End-Tag (`</beispiel>`). Bei dem Inhalt kann es sich um reinen Text (`<beispiel>Text</beispiel>`), verschachtelte Elemente (`<beispiel><a></a></beispiel>`) oder einer Kombination daraus handeln. Falls kein Inhalt vorhanden ist, kann das Element auch abgekürzt werden (`<beispiel/>`). Des Weiteren kann ein Element über Attribute verfügen. Diese setzen sich aus einem Namen und dem zugehörigen Wert zusammen (`<beispiel name="wert" />`). Dabei darf der Name des Attributs für ein Element nur einmal vorkommen. Kommentare können überall im XML-Dokument vorkommen. Sie werden mit `<!--` eingeleitet und enden mit `-->`.

Da Elemente mit Start- und End-Tags beginnen und enden, darf der Textinhalt eines Elementes bestimmte Zeichen nicht enthalten. Dazu gehören beispielsweise < und >. Diese müssen mit Maskierungszeichen ersetzt werden (`&lt;` für <). Problematisch ist vor allem das Einbinden von Binärdaten, wie Bilder oder Videos. Eine Möglichkeit ein Bild in XML einzubinden ist es, die Binärdaten in einem Vorverarbeitungsschritt mit Base64 zu kodieren (siehe folgender Abschnitt Base64-Kodierung).

Ein XML-Dokument wird als wohlgeformt bezeichnet, wenn alle unzulässigen Zeichen durch Maskierungzeichen ersetzt wurden. Zusätzlich darf das Dokument nur ein Wurzelelement enthalten. Alle Elemente müssen korrekt geschachtelt sein und die End-Tags dürfen nicht fehlen.

Um ein XML-Dokument zu validieren, gibt es mehrere Möglichkeiten. Zum einen kann die Struktur mit einer Document Type Definition (DTD)[4] definiert werden. DTD erlaubt es unter anderem die Namen von Elementen festzulegen und wie oft ein Element in einem anderen vorkommen darf. Hierbei ist es allerdings nicht möglich Datentypen zu definieren. Eine andere Möglichkeit ist XML Schema (siehe nächster Abschnitt). Wenn ein XML-Dokument einer DTD oder einem XML Schema entspricht, wird es als valide bezeichnet.

Um Kollisionen zwischen verschiedenen Elementen mit dem gleichen Namen verhindern zu können, wird das Konzept der Namespaces eingesetzt. Ein Namespace für ein Dokument wird mit dem Attribut `xmlns="..."` definiert. Man kann einem Namespace auch einen Namen zuweisen `xmlns:beispiel="..."`. Der Namespace wird vor dem Elementnamen im Start- und End-Tag, mit einem Doppelpunkt als Trennzeichen, notiert. So kann man die Elemente `<person:name/>` und `<projekt:name/>` unterscheiden. Die Kombination vom Name des Namespaces und dem lokalen Elementnamen wird als Qualified Name (QName) bezeichnet.

[4] http://www.w3.org/TR/xml11/#dt-attdecl

Es existieren viele Dokumentenformate auf Basis von XML, so zum Beispiel SOAP (siehe Abschnitt 2.2.6) oder WSDL (siehe Abschnitt 2.2.8).

XML-Schema

Die XML-Schema-Definition (XSD) [1] erlaubt es, ähnlich wie DTD, die Struktur von XML-Dokumenten zu definieren. XML-Schema benützt selbst eine XML-Syntax. Neben der Dokumentenstruktur werden einfache und komplexe Datentypen unterstützt. Aus einer Menge von primitiven Datentypen können rekursiv komplexere Strukturen erzeugt werden. Eine XML-Schema-Definition kann in andere Dokumente importiert werden. Sie ermöglicht die Validierung von XML-Dokumenten und die Wiederverwendung von Datentypen und Strukturen.

XPath

Die Anfragesprache XML Path Language[5] (XPath) wurde entworfen, um in einem XML-Dokument Elemente, Attribute und andere Informationsknoten adressieren zu können. XPath benützt dabei eine Syntax ähnlich der von Dateipfaden. Mit XPath kann eine Menge von Elementen in einem XML-Dokument ausgewählt und extrahiert werden oder eine bestimmte Stelle adressiert werden.

Base64-Kodierung

Base64[6] ist ein Verfahren zur Kodierung von 8-Bit Binärdaten. So können beispielsweise Bilder in XML-Dokumente eingefügt werden. Dabei wird ein Chunk von 6-Bit Binärdaten als ein `char` in 8-Bit dargestellt. Üblicherweise werden hierfür die druckbaren Zeichen 0-9, a-z und A-Z, sowie zwei weiteren Zeichen, die sich abhängig von der Base64-Variante unterscheiden, verwendet. Dadurch erhöht sich der Platzbedarf um etwa 33%. Eine Kodierung des Wortes „`beispiel`“ resultiert in der Zeichenkette „`YmVpc3BpZWw=`“.

2.2.6 SOAP

SOAP [20] ist eine Nachrichtenarchitektur, die es ermöglicht Informationen in einer dezentralisierten, verteilten Umgebung austauschen zu können. Ursprünglich stand SOAP für Simple Object Access Protocol, aber seit Version 1.2 ist SOAP kein Akronym mehr und steht für sich selbst. Dabei ist SOAP nur eine Spezifikation für den Austausch von strukturierten Informationen in Web Services und verwendet für den eigentlichen Transport der Daten die Protokolle der Anwendungsschicht, wie zum Beispiel HTTP oder SMTP. Diese Nachrichtenprotokolle können sich entlang der Knoten auf dem Nachrichtenpfad ändern und SOAP beschreibt, wie die Nachrichten zu verarbeiten sind. Der Nachrichtenpfad startet beim *Initial Sender* und kann sich über beliebig viele *Intermediaries* bis hin zum *Ultimate Receiver* erstrecken (siehe Abbildung 2.9).

[5] http://www.w3.org/TR/1999/REC-xpath-19991116/
[6] https://tools.ietf.org/html/rfc4648

```
POST /StudyService.asmx HTTP/1.1
Host: 127.0.0.1
SOAPAction: GetAllParticipants

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <GetAllParticipants xmlns="http://www.iaas.uni-stuttgart.de/studienarbeit/eyetracking" />
  </soap:Body>
</soap:Envelope>
```

Abbildung 2.8: Beispielhafte SOAP-Nachricht, die mittels HTTP gesendet wird. Die SOAP-Nachricht selbst befindet sich im HTTP-Body.

Dabei ist zu beachten, dass SOAP ein zustandsloses Einwegparadigma zum Nachrichtenaustausch definiert. Komplexere Interaktionsmuster (zum Beispiel Request/Response) müssen darauf aufbauend festgelegt werden. Dies kann durch Ausnützen des zugrundeliegenden Protokolls geschehen oder durch das Hinzufügen von speziellen Informationen zu den Nachrichten.

Die Syntax von SOAP basiert auf XML und erlaubt damit den Austausch von typisierten und strukturierten Informationen (siehe Abbildung 2.8). Das Wurzelelement in einem SOAP-Dokument ist `<envelope/>`. Dieses enthält ein optionales `<header/>` und ein `<body/>` Element. Der Inhalt des Dokuments wird entweder im Document-Style oder im RPC-Style Format dargestellt. Im Document-Style enthält das `<body/>` Element beliebigen XML Inhalt und die aufzurufende Operation muss anderweitig angegeben werden. Im RPC-Style enthält das `<body/>` Element nur ein Subelement mit dem Namen der Operation (`<requestname/>`). In der Antwort wird nach Konvention dann „Response“ an den Namen des Subelementes angehängt (`<requestnameResponse/>`).

Das Transportprotokoll kann sich entlang des Nachrichtenpfades ändern. Daraus folgt, dass ein Mechanismus zum Erzwingen der nichtfunktionalen Eigenschaften notwendig ist. Dies geschieht durch die Definition von Rollen, durch welche die Verarbeitungslogik umgesetzt wird. Im SOAP Header werden Regeln definiert, die für die Verarbeitung der Nachricht notwendig sind. Ein Intermediary liest alle Header aus, die für seine Rolle zutreffen und verarbeitet die Nachricht entsprechend. Der Intermediary entfernt daraufhin die Header. Bevor die Nachricht zum nächsten Intermediary weitergeleitet wird, können neue Header gesetzt werden. So können die Serialisierungsregeln jeweils für den nächsten Knoten angegeben werden und das Tansportprotokoll kann sich entlang des Nachrichtenpfades unter der Einhaltung der nichtfunktionalen Eigenschaften ändern.

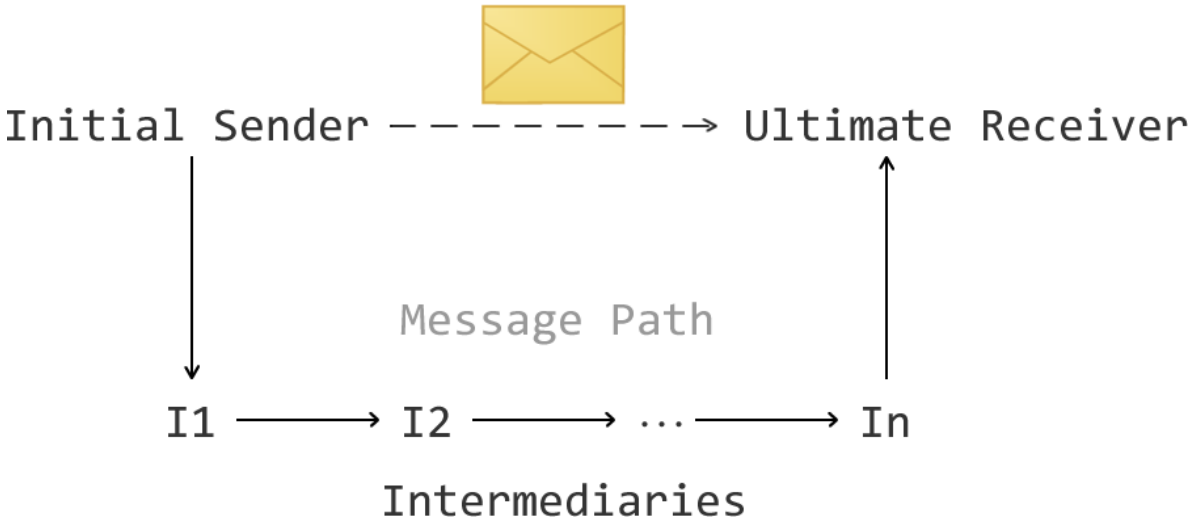

Abbildung 2.9: Der SOAP-Nachrichtenpfad. Eine SOAP-Nachricht wird nicht direkt vom Initial Sender zum Ultimate Receiver gesendet. Vielmehr wird sie abhängig von den SOAP-Headern über eine Reihe von Intermediaries geleitet, die die SOAP-Nachricht gegebenenfalls verändern. Das Transportprotokoll kann sich zwischen den Intermediaries ändern. Der Ultimate Receiver steht am Ende des Nachrichtenpfades und liest den Body der SOAP-Nachricht aus.

2.2.7 Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI) ermöglicht es, Informationen über Unternehmen und Web Services zu veröffentlichen und aufzufinden. Der Standard[7] wird von der „Organization for the Advancement of Structured Information Standards“[8] (OASIS) gefördert. Dabei unterscheidet man zwischen der Spezifikation (UDDI Service Registry Specification) und der Registratur als eigener Service (UDDI Registry Service). UDDI dient als Einstiegspunkt bei der Suche nach Services und erhöht die Wiederverwendbarkeit dieser erheblich. So können Serviceimplementierungen dynamisch zur Laufzeit ausgewählt werden. Serviceanbieter können die Beschreibung ihrer Implementierung in Form eines WSDL-Dokumentes einstellen und Servicenehmer wählen den Service aus, der am besten zu den eigenen Anforderungen passt.

Die Registratur ist logisch zentriert und physikalisch verteilt. Das bedeutet, dass mehrere UDDI Knoten existieren können, die sich gegenseitig aktuell halten. UDDI ist plattformunabhängig und basiert auf XML.

2.2.8 Web Services Description Language

Die Web Services *Definition* Language (Version 1.1) [4] oder Web Services *Description* Language (seit Version 2), kurz WSDL, ist eine auf XML basierende Beschreibungssprache für die Funk-

[7] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec
[8] https://www.oasis-open.org/

tionalität von Web Services. Eine WSDL-Datei von einem Web Service ist maschinenlesbar und in einen abstrakten und einen konkreten Teil aufgeteilt (siehe Abbildung 2.10).

Der **abstrakte Teil**, oder auch das „Service Interface“, beschreibt einen Web Service anhand der eingehenden und ausgehenden Nachrichten. Dabei wird nicht darauf eingegangen, welches Transportprotokoll verwendet wird oder welcher Endpoint konkret adressiert werden soll. Viel mehr wird die Struktur der Parameter und des Ergebnisses sowie das Muster des Nachrichtenaustausches definiert.

In WSDL 1.1 sind vier verschiedene Muster definiert. Bei dem „One-Way“ Muster wird eine Nachricht an den Service geschickt, auf die keine Antwort folgt. Im Gegensatz dazu wird beim „Request-Response“ Muster vom Service eine Antwort auf eine Nachricht produziert. Im umgekehrten Fall, dem „Solicit-Response“ Muster, schickt der Service eine Nachricht und erwartet daraufhin eine Antwort. Wenn der Service nur eine Benachrichtigung versendet, ohne dass er eine Antwort erwartet kommt, das „Notification“ Muster zum Einsatz.

Es ist möglich die abstrakte Definition in einer eigenen WSDL-Datei zu definieren und dann zu importieren. Mit der Hilfe von Namespaces werden Kollisionen bei Definitionen mit gleichem Namen vermieden.

Im Gegensatz dazu wird im **konkreten Teil** (auch „Service Implementation“ genannt) der abstrakte Teil referenziert und beschrieben, wie die Nachrichten serialisiert werden müssen, um mit einem bestimmten Service interagiert zu können. WSDL gibt dabei kein Standardformat vor. Vielmehr benützt es Protokollbindings, um die Nachrichten zu formatieren. Im konkreten Teil wird das Transportprotokoll, der Endpoint und das Encoding festgelegt. WSDL sieht es vor, dass mehrere konkrete Definitionen auf denselben abstrakten Teil referenzieren können und dieser somit wiederverwendet werden kann.

2.2.9 Integration von Anwendungen

Um bereits existierende Programme in die eigene Anwendung integrieren zu können, gibt es verschiedene Ansätze. Das grundsätzliche Ziel ist es, Informationen über Zustandsänderungen in der Anwendung „nach außen“ zu bringen und umgekehrt die Anwendung „von außen“ zu beeinflussen. Die Vorgehensweise ist davon abhängig, ob der Quelltext der existierenden Anwendung vorliegt oder nicht. Die Schnittstelle zwischen zwei Programmen wird Adapter genannt und dieser kann an verschiedenen Stellen angesetzt werden.

Bei einem „User Interface Adapter“ wird die Benutzeroberfläche auf Änderungen beobachtet. Um mit der Anwendung interagieren zu können, müssen Benutzereingaben simuliert werden. Falls die Anwendung ein öffentliches Application Programming Interface (API) bereitstellt, können Änderungen über diese Schnittstelle vorgenommen werden. Je nach API können so Lese- und Schreiboperationen durchgeführt werden. Diese Art der Integration wird „Business Logic Adapter“ genannt. Ein „Database Adapter“ achtet auf Änderungen in der Datenbank. Dies kann mit Triggern realisiert werden. Mit INSERT-, UPDATE- und DELETE-Anfragen können die Daten von außen bearbeitet werden. Dabei muss beachtet werden, dass die Konsistenz der Datensätze erhalten bleibt.

Ein Adapter, der Zustandsänderungen in der externen Anwendung beobachtet, nennt man Outbound-Adapter. Der umgekehrte Fall, um die Anwendung zu beeinflussen, wird Inbound-Adapter genannt.

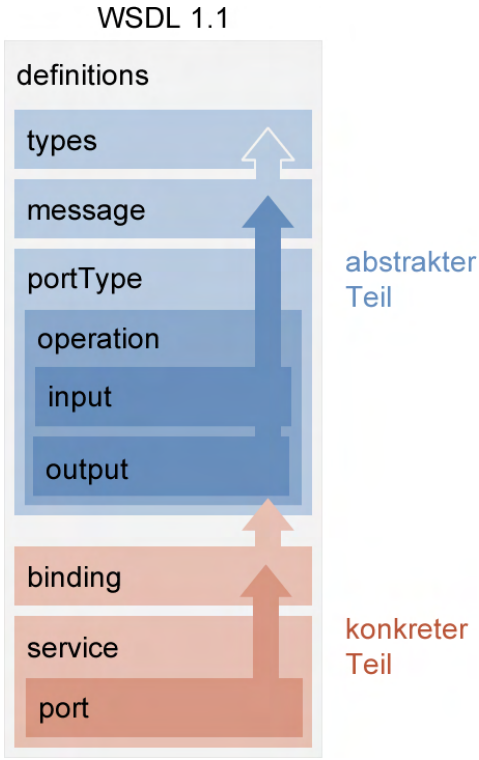

Abbildung 2.10: Aufbau eines WSDL-Dokuments. Der konkrete Teil referenziert auf den abstrakten Teil. Von einer abstrakten Definition können mehrere konkrete Instanzen erzeugt werden. Somit ist es möglich den abstrakten Teil wiederzuverwenden.

Um Nachrichten zwischen den verschiedenen Systeme austauschen, können Mediatoren eingesetzt werden. Ein Mediator übersetzt die Nachrichten im Format der einen Anwendung in das Format der anderen Anwendung.

2.3 Workflow Technologie

Dieser Abschnitt beschäftigt sich mit der Workflow Technologie. Zunächst werden die Begriffe „Geschäftsprozess“ und „Workflow“ definiert. Anschließend wird das Teilgebiet Business Workflows genauer vorgestellt. Daraufhin werden die Unterschiede der Scientific Workflows gegenüber den Business Workflows aufgezeigt. Abschließend werden Workflow Management Systeme und die Workflow-Sprache BPEL eingeführt. Die Abschnitte basieren, soweit nicht anders angegeben, auf dem Buch „Production workflow“ [21].

2.3.1 Geschäftsprozesse und Workflows

Um ein Produkt herzustellen oder einen Service durchzuführen sind oft die gleichen Arbeitsschritte notwendig, die immer wieder ausgeführt werden müssen. Typische Beispiele sind das Assemblieren von Teilen am Montageband oder die Eröffnung eines neuen Kontos. Die zu bearbeiteten Aufgaben folgen dabei stets den gleichen Mustern. Sie sind geordnet und bilden eine Prozesskette mit Vorbedingungen, Nachfolgeaufgaben und alternativen Pfaden. Diese Prozesse werden als Geschäftsprozesse bezeichnet.

Bei Geschäftsprozessen wird zwischen dem abstrakten Prozessmodell und der tatsächlichen Ausführung der Prozessmodelle unterschieden. Das abstrakte Prozessmodell beschreibt, in welcher Reihenfolge die einzelnen Schritte ausgeführt werden müssen, damit ein Arbeitsablauf abgearbeitet werden kann. Die eigentliche Ausführung, der Prozess selbst, ist eine Instanz dieses Prozessmodells. Wird das Prinzip der Prozesse und Prozessmodelle auf den Computer angewendet, werde sie entsprechend Workflow und Workflowmodell genannt (siehe Abbildung 2.11). Die Modellierung von Workflows basiert dabei auf Graphen, insbesondere dem Prinzip der Petri-Netzen. Eine typische Visualisierung für Workflows sind Flussdiagramme. Nicht alle Arbeitsschritte müssen bei einem Workflow am Computer ausgeführt werden. Deshalb wird zwischen manuellen Aufgaben und Datenverarbeitungsaufgaben unterschieden.

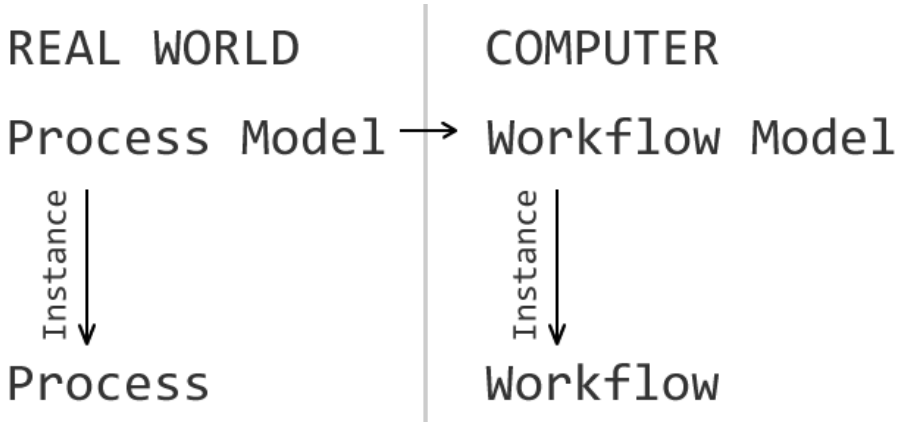

Abbildung 2.11: Prozessmodelle werden auf Workflowmodelle abgebildet. Die Instanz eines Prozessmodells wird Prozess genannt. Entsprechend ist die Instanz eines Workflowmodells ein Workflow [21].

2.3.2 Business Workflows

Workflows im Unternehmenskontext werden als Business Workflows bezeichnet. Sie ermöglichen, dass Anwendungen verschiedener Unternehmen miteinander geordnet kommunizieren können, indem sie eine Logik über einer Menge von Service Interaktionen mit Hilfe der Web Service Technologien definieren.

Ein Workflow besteht aus einer Menge von Aktivitäten, die nacheinander oder gleichzeitig abgearbeitet werden. Durch Kontrollstrukturen kann der Workflow beeinflusst werden. Aktivitäten können Unteraktivitäten beinhalten, die transparent bearbeitet werden. Umgekehrt kann eine Aktivität auch ein Unteraktivitäten einer anderer Aktivität sein. Aktivitäten können andere Geschäftsprozesse auslösen, die dann unabhängig von der ursprünglichen Aktivität laufen. Auch kann die Aktivität warten, bis der angestoßene Geschäftsprozess ein Ergebnis liefert.

Ein Workflow wird in drei Dimensionen aufgeteilt:

- **Wer?** Die am Workflow beteiligten Organisation und die Rollen, die sie einnehmen.
- **Wie?** Die Wie-Dimension beschreibt die Prozesslogik und welche Aktivitäten in welcher Reihenfolge ausgeführt werden.
- **Mit was?** Die Ressourcen, die Teil des Workflows sind.

2.3.3 Scientific Workflows

Scientific Workflows sind auf wissenschaftliche Arbeiten zugeschnitten und ein wichtiger Bestandteil von eScience. Die Vision von eScience ist, dass geografisch verteilte Wissenschaftler an großangelegten wissenschaftlichen Experimenten zusammenarbeiten können. Die Herausforderung besteht darin domänenspezifische Datentypen, Programme und verteilte Ressourcen zu verwalten. Für dieses Zusammenspiel von oftmals höchst heterogenen Anwendungen ist die Workflow-Technologie geeignet. Allerdings haben Scientific Workflows gegenüber Business Workflows besondere Anforderungen. Oft entstehen bei lang laufenden Simulationen große und komplexe Datenmengen. Den Wissenschaftlern muss es möglich sein, in einer einfachen Umgebung eigene Workflows zu erstellen, auszuführen und in Echtzeit verfolgen zu können. Die Herkunft der Ergebnisse der Workflow-Ausführung muss in einzelnen Schritten zurückverfolgt werden können. Die erstellen Workflows müssen wiederverwendbar sein und einfach zwischen Wissenschaftlern ausgetauscht werden können. Genauso sollen die Ergebnisse gemeinsam analysiert werden können [19].

In Abbildung 2.12 sind die verschiedenen Phasen von Business Workflows und Scientific Workflows gegenübergestellt. Bei Business Workflows werden die Phasen Modellierung (Modeling), Deployment, Ausführung (Execution), Beobachtung (Monitoring) und Analyse (Analysis) jeweils von Fachmännern ausgeführt, die sich nur um die jeweilige Phase kümmern. Der Workflow wird modelliert, um ein Betriebsziel zu erfüllen und anschließend an IT-Spezialisten übergeben, die ihn ausführbar machen. Nach dem Deployment wird der Workflow von einem Kunden oder einem Angestellten ausgelöst. Üblicherweise werden mehrere Instanzen des Workflows angelegt, die beobachtet werden müssen. Die Workflowausführungen werden analysiert und der Workflow wird daraufhin eventuell angepasst. Bei Scientific

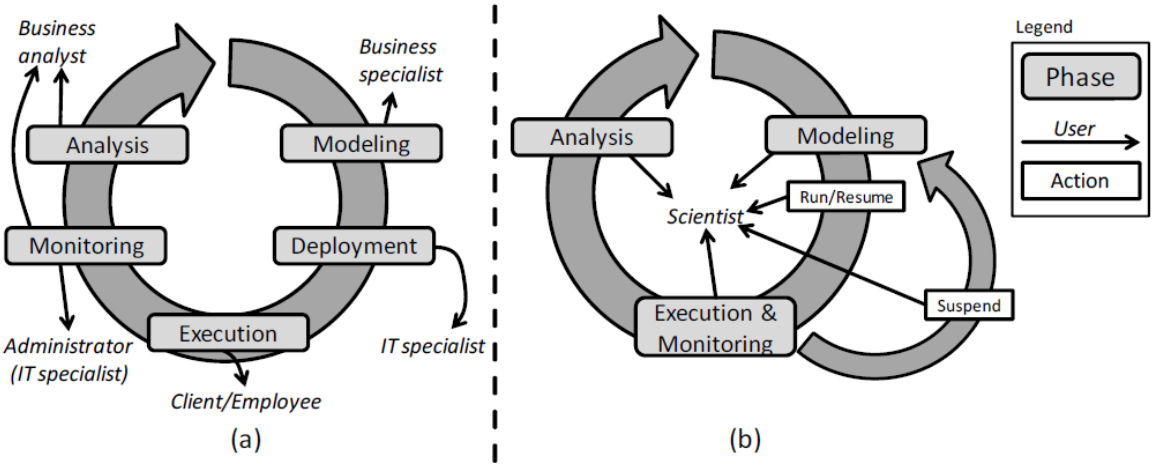

Abbildung 2.12: Ein Business Workflow (a) wird in verschiedenen Phasen erstellt, die jeweils von anderen Benutzergruppen bearbeitet werden. Bei einem Scientific Workflow (b) übernimmt ein Wissenschaftler die verschiedenen Aufgaben. Der Scientific Workflow wird iterativ verbessert und hat daher einen zusätzlichen Zyklus zwischen der Modellierung und Ausführung [19].

Workflows kümmern sich die Wissenschaftler um die verschiedenen Phasen. Sie übernehmen die Rollen der Modellierer, Administratoren, Benutzer und Analysten. Der Schwerpunkt der Arbeit ist häufig nur eine Workflow-Instanz, bei der eine Simulation aufgesetzt und direkt ausgeführt wird. Der Workflow wird durch das Prinzip von Versuch und Irrtum erstellt, nach der Ausführung werden die Ergebnisse analysiert und das Modell angepasst oder der Workflow mit anderen Parametern ausgeführt. Daher gibt es einen zusätzlichen Zyklus zwischen Modellierung und Ausführung, bei dem die aktive Instanz unterbrochen (Suspend) und das Modell verändert wird. [19].

2.3.4 Workflow Management Systeme

Bei einem Workflow Management System (WfMS) handelt es sich um Software, welche die Definition, das Management und die Ausführung von Workflow-Modellen erlaubt. Die Kernkomponente ist die Workflow-Engine. Diese kümmert sich um die Instanziierung von Workflows und überwacht deren Fortschritt. In einer Datenbank kann der Fortschritt eines Workflows persistent gespeichert werden.

Neben der Ausführung ist die Modellierung der Workflows eine weitere wichtige Aufgabe eines WfMS. Dazu gehört die Spezifikation von Services, die vom Workflow benützt werden.

Die üblichen Workflow Management Systeme sind auf Business Workflows ausgelegt. Im Fall von Scientific Workflows ist es sinnvoll ein Scientific Workflow Management System (SWfMS) zu verwenden, das auf die Anforderungen der Scientific Workflows und der Wissenschaftler zugeschnitten ist. Scientific Workflows sind datengesteuert und verarbeiten oft große Datenmengen. Wichtig ist die Benutzerfreundlichkeit, da es sich nicht bei allen Wissenschaftlern um Informatiker handelt. Deshalb sollte die Modellierung ein möglichst

einfacher Prozess sein. Die Services müssen von den Wissenschaftlern selbst festgelegt werden können. Außerdem muss es möglich sein, den Workflow jederzeit anzupassen. Weiterhin ist es erforderlich, dass die Prozessausführung beobachtet werden kann und dass sich die Ergebnisse zurückverfolgen lassen. Da Simulationen oft langläufig sind, muss das System robust sein. Auch soll es den Benutzer bei häufigen Aufgaben unterstützen und das Deployment möglichst einfach gestalten, indem das System dem Benutzer die Arbeit abnimmt [19].

2.3.5 Business Process Execution Language

Die Business Process Execution Language (BPEL) [5][34] ist eine erweiterbare Workflow-Sprache mit der Geschäftsprozesse beschrieben werden können. BPEL ist auch unter den Namen „Web Services BPEL“ (WS-BPEL) und „BPEL for Web Services“ (BPEL4WS) bekannt. Mit BPEL ist es möglich, Aktionen in Geschäftsprozessen zu spezifizieren, die von Web Services ausgeführt werden. BPEL-Prozesse importieren und exportieren Informationen ausschließlich über Web Service-Schnittstellen. BPEL-Prozesse können selbst als Web Services verwendet werden, da sie ein rekursives Kompositionsmodell besitzen.

Eine Eigenschaft von BPEL ist die flexible Integration. Ein BPEL-Prozess ist von bestimmten Serviceinstanzen entkoppelt. Diese werden erst zur Laufzeit gebunden.

BPEL enthält Sprachkonstrukte, die aus der strukturierten Programmierung bekannt sind. So sind if-elseif-else-Blöcke, while-Schleifen und sequentielle und parallele Ausführungen möglich. Durch alternative Pfade wird der *Kontrollfluss* gesteuert. Der *Datenfluss* hingegen bestimmt, welche Daten ausgetauscht werden müssen. Die folgenden Abschnitte basieren auf dem Buch „Web Services Platform Architecture“ [34].

Scopes

Mit Hilfe des Scoping-Systems wird die Logik des Workflows gekapselt. Ein Scope enthält lokale Variablen und ermöglicht den Austausch von Daten zwischen den Aktivitäten eines Scopes. Fehler können innerhalb eines Scopes behoben werden, indem sie kompensiert werden. Auch Events sind an den Scope gebunden.

Variablen

Variablen in BPEL leben innerhalb des umgebenden Scopes. Die Werte dieser Variablen sind entweder Nachrichten zwischen dem Prozess und den Partnern oder Zwischendaten, die privat für den Prozess sind. Der Datentyp der Variablen wird durch WSDL-Nachrichtentypen oder XML-Schema definiert. Der Inhalt der Variablen kann mit XPath abgefragt und manipuliert werden.

Aktivitäten

Es gibt einfache und strukturierte Aktivitäten. Einfache Aktivitäten sind beispielsweise eingehende oder ausgehende Web Service-Interaktionen oder spezielle Aktivitäten für die Datenmanipulation. Strukturierte Aktivitäten enthalten andere Aktivitäten und definieren die Kontrollflussabhängigkeit zwischen ihnen.

Zu den einfachen Aktivitäten gehören die Empty, Wait, Terminate, Throw und Compensate Aktivitäten. Die strukturierten Aktivitäten setzen sich aus Sequence (sequentielle Ausführung von Aktivitäten), While für Wiederholungen und Flow (parallele Ausführung von Aktivitäten) zusammen.

In BPEL werden Daten geschrieben und gelesen, indem Aktivitäten Nachrichten zwischen dem Prozess und seinen Partnern austauschen. Mit speziellen Zuweise-Aktivitäten können Daten manipuliert werden. Bedingungen fragen die Daten ab und führen Vergleiche auf boolesche Wert, Zeitintervalle und Variablen durch.

Zu den Aktivitäten, die mit Web Services interagieren gehören Receive (Eingabe), Reply (Request/Response), Pick, Invoke und Event Handler.

Partner Links

Partner Links beschreiben die Interaktionen zwischen dem Prozess und seinen Partnern. Dem Prozess und den Partnern werden Rollen zugewiesen, die auf WSDL-PortTypes abgebildet werden. Die Bindings und Ports können zur Laufzeit festgelegt werden. Jede Kommunikationsaktivität des Prozesses spezifiziert die Rollen der Partner Links und die zugehörigen Variablen für den Nachrichtenaustausch.

Abstrakte und ausführbare Prozesse

In BPEL können ausführbare und abstrakte Prozesse definiert werden. Abstrakte Prozesse dienen als Vorlage für andere Prozesse und sind nicht dazu gedacht ausgeführt zu werden. Die Modellierung der ausführbaren und abstrakten Prozesse erfolgt auf die gleiche Art und Weise.

Instanziierung

Nachdem ein BPEL-Prozess definiert und deployt wurde, kann er instanziiert werden. Immer wenn ein BPEL-Prozess gestartet wird, wird eine neue Instanz von ihm angelegt. Es ist möglich, dass mehrere Instanzen des selben BPEL-Prozesses existieren. Eine Instanz terminiert, wenn die letzte Aktivität ausgeführt wurde oder wenn ein Fehler auftritt, der nicht mehr kompensiert werden kann. Da die Prozessinstanzen über einen großen Zeitraum hinweg laufen können, müssen Fehler erkannt und behandelt werden. Dafür werden Compensation Handler definiert, die bereits ausgeführte Aktivitäten innerhalb eines Scopes rückgängig machen. Die Verwaltung der Instanzen wird vom Workflow Management System übernommen.

3 Verwendete Software

In diesem Kapitel wird die verwendete Software vorgestellt, auf der die Implementierung basiert. Als Grundlage für die Eye-Tracking-Funktionalität dient das Programm eTaddy. Um die modellierten Workflows auszuführen wird das SimTech SWfMS eingesetzt.

3.1 eTaddy

Das Programm eTaddy (eyeTracking Analysis, conDuction, and Designtool for userstudYs) wurde von Tanja Blascheck im Rahmen ihrer Diplomarbeit erstellt. Die Arbeit hat den Titel „Eyetracking basiertes Analysekonzept zur Evaluation von Visualisierungen“ [2] und stellt ein Konzept für ein Framework vor, das den Benutzer bei der Erstellung, Durchführung und Auswertung von Eye-Tracking Benutzerstudien unterstützen soll. Die bei einer Eye-Tracking Benutzerstudie anfallenden Eye-Tracking- und Probandendaten werden in einer Datenbank gespeichert. Für die Auswertung werden verschiedene Metriken, Statistiken und Visualisierungen als Plug-ins zur Verfügung gestellt. Eine Liste bereits implementierter Plug-ins zeigt Tabelle 3.1. Die Ergebnisse der Berechnungen werden auch in der Datenbank gespeichert, um sie zu einem späteren Zeitpunkt wieder abrufen zu können. In den folgenden Abschnitten werden die Komponenten von eTaddy vorgestellt, die für diese Arbeit wichtig sind. Die Abschnitte basieren auf der Diplomarbeit von Tanja Blascheck [2] und eigenen Erfahrungen mit der Arbeit an eTaddy.

3.1.1 Benutzeroberfläche

Die Benutzeroberfläche von eTaddy wurde nach dem Prinzip des Visual Information-Seeking Mantras von Ben Shneiderman entworfen. Das Mantra besagt „Overview first, zoom and filter, then details-on-demand“ [30]. Nach dem Start und der Auswahl der Datenbank (siehe Abbildung 3.1) wird in eTaddy zunächst eine Übersicht über die Stimuli einer Eye-Tracking Benutzerstudie gegeben (Overview). Der Benutzer kann anschließend einzelne Probanden auswählen (Filter) und verschiedene Metriken, Statistiken und Visualisierungen anwenden (Details-on-demand), die als Plug-ins realisiert sind. Die Plug-ins erwarten unterschiedlich viele Probanden als Parameter und werden in der Benutzeroberfläche ausgegraut, wenn eine ungültige Anzahl an Probanden markiert wurde. Die Ergebnisse von Berechnungen werden in einer Timeline dargestellt. Dabei stehen neue Ergebnisse oben in einer Liste und der Benutzer kann ältere Berechnungen einsehen, indem er die Scrollbar nach unten verschiebt. In Abbildung 3.2 ist das Hauptfenster der Anwendung, mit einer Erklärung der einzelnen Komponenten, zu sehen.

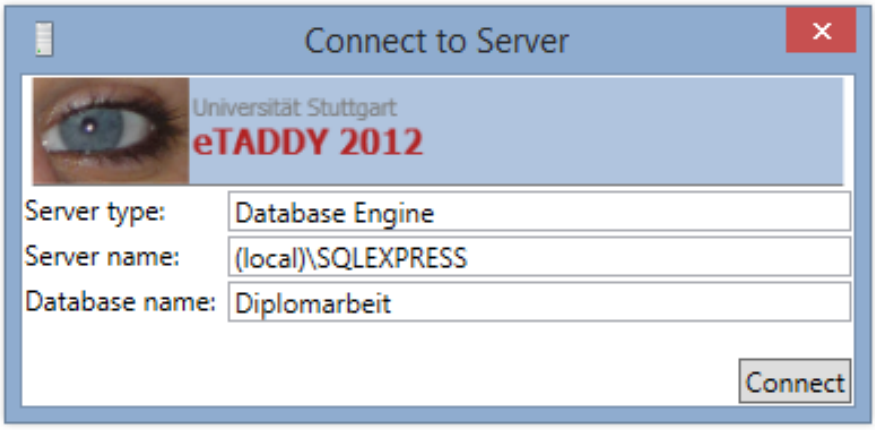

Abbildung 3.1: Der Dialog zur Auswahl der Datenbank beim Start von eTaddy. Erforderliche Eingaben sind der Typ der Datenbank, der Servername und der Name der Datenbank.

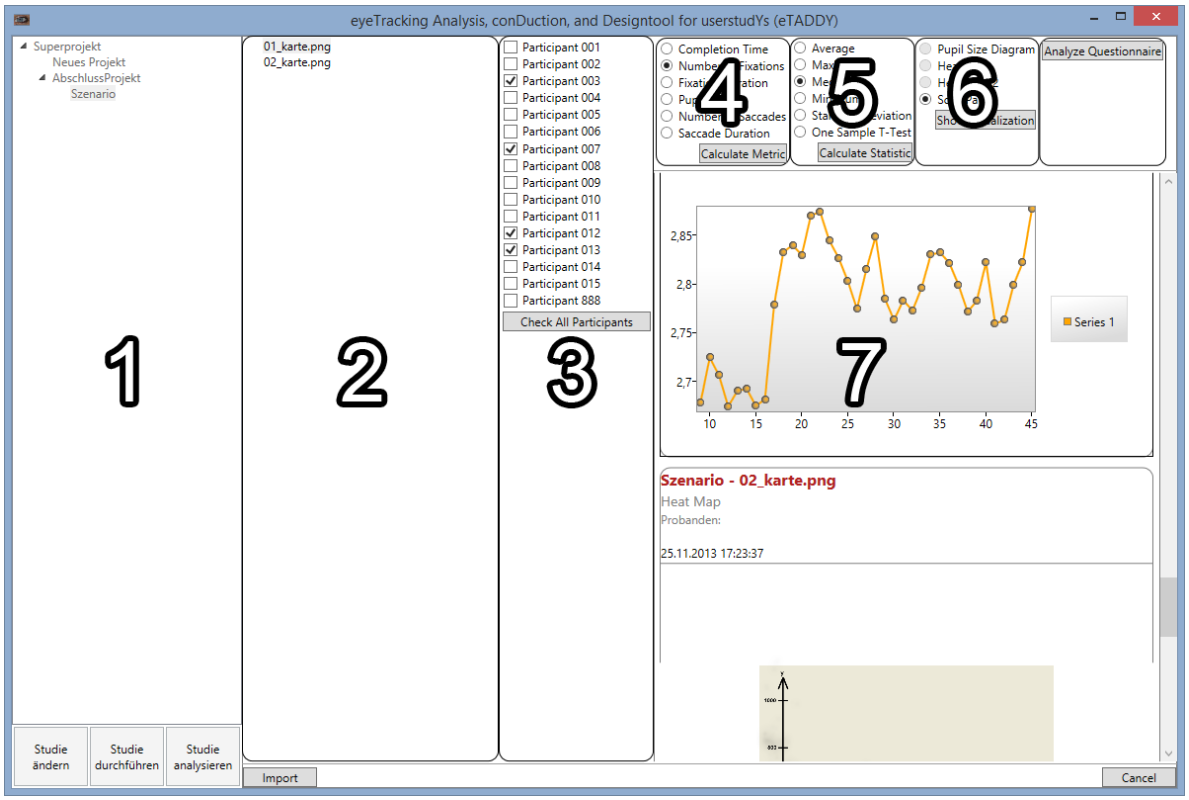

Abbildung 3.2: Das Hauptfenster von eTaddy. Zu sehen ist 1) Auswahl der Studie, 2) Auswahl der Stimuli, 3) Auswahl der Probanden, 4) Metriken, 5) Statistiken, 6) Visualisierungen und 7) die Timeline mit den bereits ausgeführten Berechnungen.

Name	Typ	Eingabe	Ausgabe
Bearbeitungszeit	Metrik	>0 Probanden 1 Stimulus	Probandenname Fertigstellungszeit (s)
Fixationsanzahl	Metrik	>0 Probanden 1 Stimulus	Probandenname Fixationsanzahl
Fixationsdauer	Metrik	1 Proband 1 Stimulus	Fixationsindex Fixationsdauer (ms)
Pupillengröße	Metrik	1 Proband 1 Stimulus	Fixationsindex Pupillengröße (mm)
Sakkadenanzahl	Metrik	>0 Probanden 1 Stimulus	Probandenname Sakkadenanzahl
Sakkadendauer	Metrik	1 Proband 1 Stimulus	Probandenname Sakkadendauer (ms)
Durchschnitt	Statistik	Abh. v. Metrik	Abh. v. Metrik
Maximum	Statistik	Abh. v. Metrik	Abh. v. Metrik
Median	Statistik	Abh. v. Metrik	Abh. v. Metrik
Minimum	Statistik	Abh. v. Metrik	Abh. v. Metrik
Standardabweichung	Statistik	Abh. v. Metrik	Abh. v. Metrik
T-Test	Statistik	Abh. v. Metrik	Abh. v. Metrik
Heatmap	Visualisierung	0 Probanden[a] 1 Stimulus	Heatmap-Visualisierung
Scanpath	Visualisierung	>0 Probanden 1 Stimulus	Scanpath-Visualisierung
Diagramm der Pupillengröße	Visualisierung	1 Proband 1 Stimulus	Visualisierung der Pupillengröße im Verlauf der Zeit

Tabelle 3.1: Liste der implementierten Plug-ins in eTaddy [2].

[a] Das Heatmap Plug-in liest immer alle Probanden aus.

3.1.2 Auswertung einer Studie

Um eine Eye-Tracking Benutzerstudie auszuwerten, muss der Benutzer eine Verbindung zur Datenbank herstellen. Im nächsten Schritt wird ein Projekt, also eine Benutzerstudie, mit einem Klick auf „Studie analysieren“ geöffnet. Das Programm lädt daraufhin die Daten der Studie und die bereits durchgeführten Berechnungen aus der Datenbank. Im nächsten Schritt wird ein Stimulus ausgewählt. Anschließend werden die Probanden angezeigt, die den Stimulus bearbeitet haben. Auf den gewählten Stimulus und eine Auswahl an Probanden können nun die Metriken angewendet werden. Um eine Statistik zu berechnen muss immer auch noch eine Metrik ausgewählt sein, da sich die Statistiken auf die Metriken beziehen. Alle Berechnungen werden schließlich in der Timeline angezeigt, mit der es möglich ist, den kompletten Verlauf der Berechnungen zu verfolgen.

3.2 SimTech Scientific Workflow Management System

Das SimTech Scientific Workflow Management System (SimTech SWfMS) ist im Rahmen des DFG Cluster of Excellence Simulation Technology[1] (SimTech) entstanden. Es baut auf den Eclipse BPEL Designer[2] auf. Dementsprechend werden die Workflowmodelle in BPEL erstellt [19].

3.2.1 Benutzeroberfläche

Abbildung 3.3 zeigt die Benutzeroberfläche des SimTech SWfMS. Die Benutzeroberfläche besitzt mehrere Erweiterungen gegebenüber dem Eclipse BPEL Designer. Die „SimTech Modeling Perspective“ (1) erlaubt das Modellieren von Workflows. Im „SimTech Service Catalog“ (2) sind verfügbare Services aufgelistet, die in der Modeling Perspective benützt werden können. Zwischenergebnisse während der Ausführung werden in der „SimTech Runtime Perspective“ (3) angezeigt. Sie unterstützt den Benutzer während der Ausführung des Workflows. Nach erfolgreicher Ausführung können die Ergebnisse in der „SimTech Analysis Perspective“ analysiert werden [19].

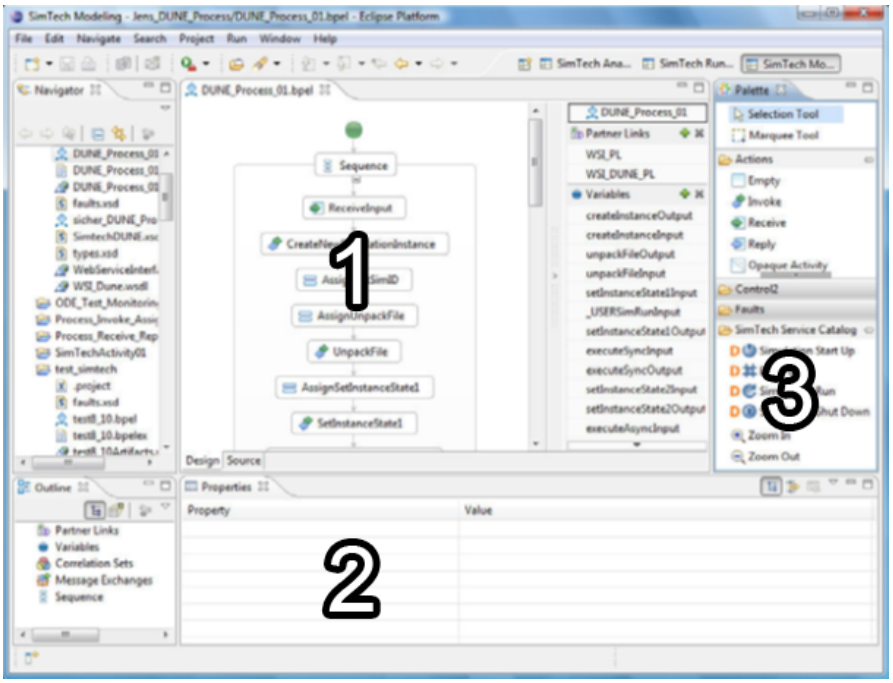

Abbildung 3.3: Die grafische Benutzeroberfläche des SimTech SWfMS. 1) Modellierung 2) Service-Katalog 3) Zwischenergebnisse [19]

[1] http://www.simtech.uni-stuttgart.de/
[2] http://www.eclipse.org/bpel/

3.2.2 Architektur

Die Ausführungsengine basiert auf Apache Orchestration Director Engine[3] (ODE). Die Funktionalität vom Enterprise Service Bus wird von Apache Axis2[4] übernommen. Axis2 kümmert sich um den Aufruf der Services, die die Workflow Aktivitäten implementieren. Um den Zustand der Ausführung festzuhalten wird eine Datenbank verwendet. Das SimTech SWfMS unterstützt verschiedene Datenbankverwaltungssysteme [19].

In Abbildung 3.4 ist die Architektur des SimTech SWfMS abgebildet. Die Grafik zeigt den Zusammenhang der Komponenten. Der Benutzer interagiert mit der grafischen Benutzeroberfläche (GUI). Er kann Services im Service-Katalog (Service Catalog) auswählen, Workflows modellieren (Workflow Modeller), den Fortschritt von aktiven Workflow-Instanzen beobachten (Monitor) und die Ergebnisse betrachten (Result Display). Im Zentrum der Laufzeitkomponenten (Runtime Components) ist die Ausführungsengine (Execution Engine). Der Service Bus kümmert sich um die Serviceauswahl (Service Discovery) und das Management der Ressourcen [19].

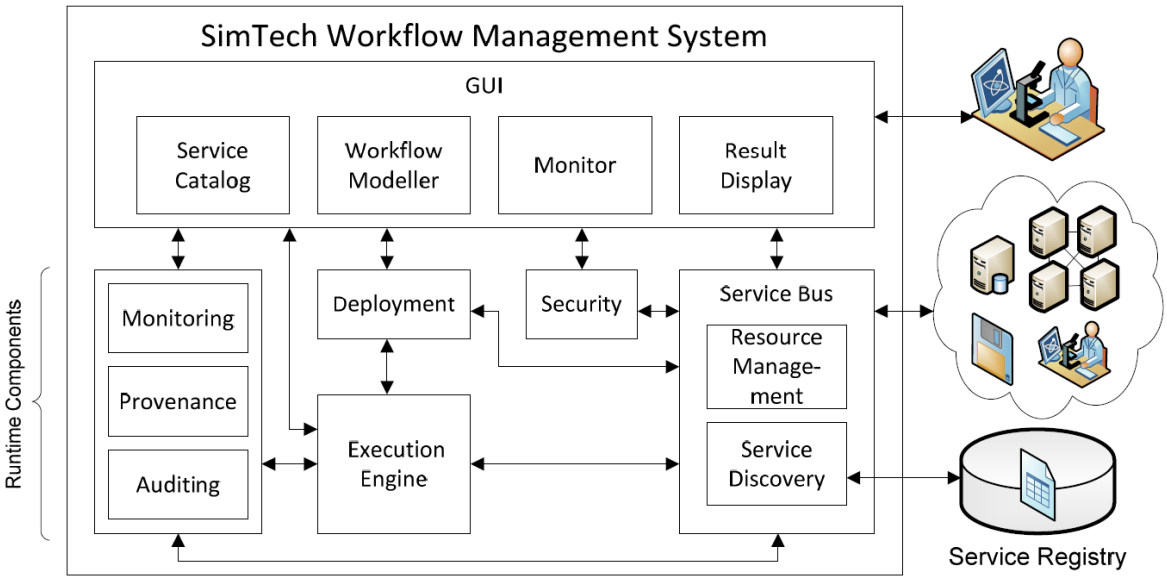

Abbildung 3.4: Architektur des SimTech SWfMS [19].

[3] http://ode.apache.org/
[4] http://ws.apache.org/axis2/

4 Verwandte Arbeiten

In diesem Kapitel wird eine Auswahl von verwandten Arbeiten vorgestellt. Bei den Arbeiten handelt es sich um Diplomarbeiten, die am Institut für Architektur von Anwendungssystemen (IAAS) an der Universität Stuttgart durchgeführt wurden. Die Arbeiten wurden wegen ihrer Ähnlichkeit gegenüber der Aufgabe dieser Studienarbeit ausgewählt.

4.1 Ausführung von Festkörpersimulationen auf Basis der Workflow Technologie

In der Diplomarbeit von Sven Hotta mit dem Titel „Ausführung von Festkörpersimulationen auf Basis der Workflow Technologie“ [13] wurde eine in Fortran geschriebene Festkörpersimulationsanwendung in einen Simulationsworkflow eingebunden. Die Arbeit basiert auf einer Kooperation zwischen dem Exzellenzcluster SimTech[1] und dem Institut für Materialprüfung, Werkstoffkunde und Festigkeitslehre[2]. Dabei wurden die manuellen Arbeitsschritte einer Simulation auf einen Simulationsworkflow abgebildet. Hierfür war es notwendig, die Funktionalität der in Fortran geschriebenen Festkörpersimulationsanwendung als Web Services zur Verfügung zu stellen. Da für Fortran kein Web Service Framework existiert, werden in der Arbeit verschiedene Möglichkeiten vorgestellt, um eine in Java geschriebene Web Service Implementierung mit dem nativen Fortran Code zu verbinden. Wichtige Anforderungen für die Umsetzung waren ein zentraler Datenspeicher und die Möglichkeit von einem aktiven Workflow Snapshots zu erstellen.

Auch in dieser Arbeit wird die Funktionalität einer bestehenden Anwendung als Web Serivce zur Verfügung gestellt, indem die manuellen Arbeitsschritte auf Web Service-Operationen abgebildet werden. Des Weiteren wird auch eine zentrale Eye-Tracking-Datenbank verwendet.

4.2 Generisches Web Service Interface um Simulationsanwendungen in BPEL-Prozesse einzubinden

Die Diplomarbeit „Generisches Web Service Interface um Simulationsanwendungen in BPEL-Prozesse einzubinden“ von Jens Rutschmann [28] befasst sich mit dem Problem, dass viele vorhandene Simulationsprogramme schlecht zu gekoppelten Simulationen zusammengefügt werden können, da sie sich von der Technologie oder Architektur zu sehr unterscheiden. Um diesem Problem entgegenzuwirken, wird das Konzept einer einheitlichen Schnittstelle vorgestellt, über welche die Simulationsanwendungen angesprochen und Daten ausgetauscht

[1] http://www.simtech.uni-stuttgart.de/
[2] http://www.imwf.uni-stuttgart.de/

werden können. Das Konzept sieht es vor die Simulationsanwendungen anschließend über die Schnittstelle mit BPEL zu verbinden. Auch soll die Schnittstelle in Zukunft eine grafische Oberfläche für die Modellierung von komplexen Simulationen unterstützten. Abschließend wurde eine Laufzeitumgebung erstellt, in der die Simulationsanwendungen, die erstellte Schnittstelle und die Workflow-Umgebung ausgeführt werden können.

In dieser Studienarbeit wird nur eine bestehende Anwendung betrachtet, deren Funktionalität über Web Service-Operationen zur Verfügung gestellt wird. Allerdings wird auch hier versucht von der zugrundeliegenden Implementierung der Operationen durch eine einheitliche Schnittstelle zu abstrahieren. Somit es es möglich, die Implementierung der Operationen flexibel auszutauschen.

4.3 Framework für die Visualisierung von Datenqualität in Simulation-Workflows

Marcel Russ stellt in seiner Diplomarbeit mit dem Titel „Framework für die Visualisierung von Datenqualität in Simulation-Workflows“ [27] ein Konzept für die Visualisierung der Datenqualität in Simulations-Workflows vor. Das Ziel des Java Data Quality Visualization Frameworks (JDQVisF) ist, laufende Simulationen zu überwachen und gegebenenfalls anhand der visualisierten Datenqualitätswerte in diese einzugreifen. Das Konzept sieht ein Rechtesystem für die an den Simulationen beteiligten Wissenschaftlern vor, um die Visualisierungen zu ändern und, um Steuerungsbefehle an den Simulations-Workflow zu senden. Des Weiteren sind verschiedene Visualisierungen für unterschiedliche Anzeigegeräte vorgesehen. In Abbildung 4.1 ist das JDQVisF als zentrale Komponente zwischen verschiedenen Simulationen und Anzeigegeräten zu sehen.

Das Prinzip die Visualisierungen für unterschiedliche Anzeigegeräte anzupassen eignet sich auch für die Visualisierung von Eye-Tracking-Daten.

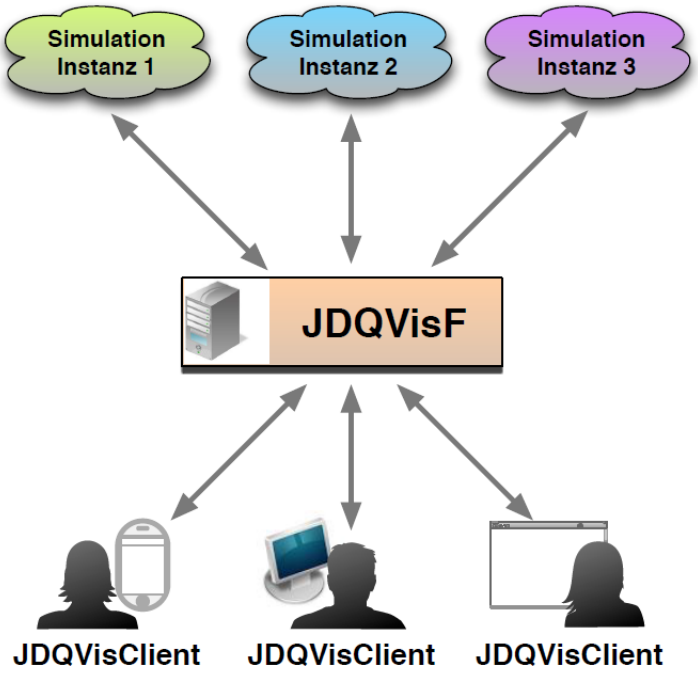

Abbildung 4.1: Das Java Data Quality Visualization Framework (JDQVisF) ist die zentrale Komponente zwischen verschiedenen Simulationen und Anzeigegeräten. Es erlaubt die Überwachung der Datenqualität anhand von Visualisierungen, die für die Anzeigegeräte zugeschnitten sind [27].

5 Aufgabe und Lösungsansatz

In diesem Kapitel werden die Hintergründe, die Aufgabenstellung und der Lösungsansatz dieser Studienarbeit vorgestellt. Entwickelt wird ein Konzept, um den Benutzer bei der Analyse von Eye-Tracking-Benutzerstudien mit Hilfe der Web Service und Workflow Technologie zu untersützen.

5.1 Hintergründe

Das Institut für Visualisierung und interaktive Systeme[1] (VIS) beschäftigt sich mit der Darstellung, Verarbeitung und Analyse großer Datenmengen. Im Zentrum aktueller Forschungsfragen auf dem Gebiet der Visualisierung stehen mit zunehmendem Maße Eye-Tracking-Experimente, die verschiedene Visualisierungstechniken auf ihre Benutzerfreundlichkeit und Aufgabenangemessenheit hin untersuchen oder miteinander vergleichen. In dieser Arbeit wird die Automatisierung der Analyse von Eye-Tracking-Benutzerstudien mit Hilfe der Workflow-Technologie behandelt.

5.2 Aufgabenstellung

Das Ziel dieser Studienarbeit ist die Entwicklung eines Konzepts für die Bereitstellung eines bestehenden Frameworks für die Analyse von Eye-Tracking Daten als Web Service. Zunächst soll die bestehende Anwendung in Bezug auf ihre Funktionalität und ihre Schnittstellen analysiert werden. Anschließend soll untersucht werden, wie diese Anwendung erweitert oder angepasst werden muss, so dass sie als Web Service bereit gestellt werden kann. Dabei soll vor allem die Modularität und Wiederverwendbarkeit der Web Service-Operationen beachtet werden. Des weiteren soll ein Workflow modelliert werden, der exemplarisch zeigen soll, wie die Analyse von Eye-Tracking Experimenten mit Hilfe von Workflow- und Web Service Technologien unterstützt werden kann. Die Studienarbeit umfasst dabei folgende Aufgaben:

- Implementierung eines Prototyps (Kapitel 7)
- Demonstration des Prototyps (Kapitel 8)

[1] http://www.vis.uni-stuttgart.de

5.3 Lösungsansatz

Im Zentrum dieser Arbeit steht die Entwicklung eines Web Services, der Operationen für die Analyse von Eye-Tracking Daten bereitstellt. Als Grundlage für die Operationen dienen die Funktionen des Eye-Tracking Analysetools eTaddy [2], das von Tanja Blascheck im Rahmen ihrer Diplomarbeit entwickelt wurde. Die Funktionen von eTaddy wurden in Form eines Plug-in-Systems implementiert und sind damit beliebig erweiterbar. Durch die Implementierung dieser Plug-in Schnittstelle in einem Web Service, können die Funktionen als Web Service-Operationen zur Verfügung gestellt werden. Die Plug-ins sind leicht austauschbar und können durch bessere Implementierungen ersetzt werden. Neue Plug-ins erweitern sowohl eTaddy als auch den Web Service um neue Funktionalität. Somit ist die Modularität und Wiederverwendbarkeit der Web Service-Operationen gegeben.

Ein weiterer wichtiger Punkt ist die Verwaltung der Daten, die bei einer Eye-Tracking Benutzerstudie anfallen. Da es sich dabei um große Datenmengen handeln kann, sollte vermieden werden, die gesamten Daten als Nachrichten zu versenden. Sinnvoller ist es, die Daten in einer Datenbank zu halten und in den Nachrichten nur auf die Daten zu referenzieren. eTaddy setzt auf ein Datenmodell, das auch von anderen Eye-Tracking Anwendungen im Institut für Visualisierung und Interaktive Systeme (VIS) eingesetzt wird. Daher ist es sinnvoll, dieses Datenmodell zu verwenden.

Die Automatisierung der Analyse von Eye-Tracking-Daten kann mit Hilfe der Workflow-Technologie realisiert werden. Die Modellierung und Ausführung des Workflows erfolgt mit dem SimTech Scientific Workflow Management System (SimTech SWfMS). Dieses System wurde am Institut für Architektur von Anwendungssystemen[2] (IAAS) im Rahmen des SimTech Exzellenzclusters entwickelt.

[2] http://www.iaas.uni-stuttgart.de/

6 Lösungskonzept

In diesem Kapitel wird das Lösungskonzept vorgestellt, dessen prototypische Umsetzung in Kapitel 7 gezeigt wird. Zunächst wird konzeptionell erklärt, wie der Benutzer bei der Analyse von Eye-Tracking-Benutzerstudien unterstützt werden kann. Anschließend wird der Aufbau des Web Service vorgestellt. Im folgenden Abschnitt werden die Anforderungen an den Web Service behandelt. Abschließend wird das Gesamtsystem veranschaulicht.

6.1 Automatisierung des Analyseprozesses

Eye-Tracking wird unter anderem in der Marktforschung, der Evaluation von Benutzeroberflächen, der Mensch-Computer-Interaktion sowie der psychologischen und neurologischen Forschung eingesetzt. Mit der Hilfe von Eye-Tracking-Benutzerstudien können verschiedene Visualisierungen oder Benutzeroberflächen verglichen werden, indem die Suchstrategien der Probanden betrachtet werden. Neben der eigentlichen Durchführung der Benutzerstudie ist vor allem die anschließende Analyse der gesammelten Daten ein zeitintensiver Prozess. Bereits nach wenigen Minuten fallen bei einer Eye-Tracking-Benutzerstudie viele Datensätze an. Diese müssen mit der Hilfe von Metriken, statistischen Berechnungen und Visualisierungen ausgewertet werden.

Mit eTaddy existiert bereits ein Framework für die Analyse von Eye-Tracking-Benutzerstudien. Das im Rahmen einer Diplomarbeit [2] entstandene Programm orientiert sich am Visual Information Seeking Mantra [30] und stellt dem Benutzer verschiedene Funktionalitäten zur Analyse von Eye-Tracking-Benutzerstudien zur Verfügung. Die grafische Benutzeroberfläche von eTaddy ist auf die Bedienung mit Maus und Tastatur ausgelegt. Der Benutzer wählt Studien, Stimuli und Probanden für die Berechnungen von Metriken, Statistiken und Visualisierungen aus. Bereits durchgeführte Berechnungen werden in einer Timeline angezeigt. Immer wieder auftretende Analyseschritte können allerdings nicht automatisiert werden. Allgemein ist die Analyse ein iterativer Prozess, bei dem die Ergebnisse der Metriken, Statistiken und Visualisierungen betrachtet werden und abhängig von diesen weitere Analyseschritte durchgeführt werden. eTaddy sieht es nicht vor, dass diese Analyseprozesse aufgezeichnet und kollaborativ mit anderen Benutzern verbessert werden. Ein Analyseprozess kann als Workflow modelliert werden, indem die einzelnen Analyseschritte auf Workflow-Aktivitäten abgebildet werden. Das Workflowmodell kann anschließend in dem SimTech Scientific Workflow Management System (SWfMS) instanziiert und ausgeführt werden.

6.2 Erstellen eines Web Services

Damit der Analyseprozess von einer Eye-Tracking-Benutzerstudie auf einen Workflow abgebildet werden kann, müssen die einzelnen Analyseschritte als Web Service-Operationen zur Verfügung gestellt werden. Die Funktionen zur Analyse in eTaddy sind als Plug-ins realisiert. Um diese Plug-ins können die Web Service-Operationen als Wrapper gebaut werden. Wird der Analyseprozess als Workflow modelliert, kann die Workflow-Aktivitäten SOAP-Nachrichten an die Web Service-Operationen schicken, die dem Analyseschritt entsprechen. In der SOAP-Nachricht können, neben der Operation selbst, die für die Operation notwendigen Parameter eingebunden werden. Der Web Service erhält die SOAP-Nachricht und ruft intern das zur Operation passende Plug-in auf und übergibt dabei die Parameter. Das Ergebnis der Berechnung des Plug-ins kann anschließend in eine XML-Struktur überführt werden und in den Body einer SOAP-Response-Nachricht eingefügt werden. Der Workflow kann auf die Antwort reagieren und gegebenenfalls einen anderen Ausführungspfad verfolgen. Die Mehrfachausführung, um beispielsweise eine Visualisierung für alle Probanden zu generieren, ist durch den Einsatz von Schleifen trivial. Die entsprechenden Operationen müssen nur in einer Schleife für jeden Probanden aufgerufen werden.

eTaddy sieht es vor, dass bei der Analyse einer Benutzerstudie zunächst der zu betrachtende Stimulus aus einer Liste mit allen Stimuli ausgewählt wird. Im nächsten Schritt werden die Probanden angezeigt, die den Stimulus bearbeitet haben (siehe Abbildung 3.2, Punkt 2 und 3). Diese Liste der Probanden dient als Grundlage für die Berechnung der Metriken, Statistiken und Visualisierungen. Auch der Web Service muss daher Operationen zur Verfügung stellen, welche die Stimuli und Probanden der Studie zurückgeben und somit als Einstiegspunkt für den Workflow dienen. In eTaddy wird für die Auswahl nur eine Liste von anonymisierten Probandennamen angezeigt. Die entsprechende Web Service-Operation kann allerdings weitere Informationen der Probanden bereitstellen, die beispielsweise über einen Fragebogen gesammelt wurden. Durch das Bereitstellen des Alters können die Probanden vom Workflow zum Beispiel einfach in verschiedene Altersgruppen eingeteilt werden.

6.3 Anforderungen an den Web Service

In den folgenden Abschnitten werden die Anforderungen an den Web Service bezüglich der Datenverwaltung, der Schnittstellen, der Modularität und der Wiederverwendbarkeit vorgestellt.

6.3.1 Datenverwaltung

Bei einer Eye-Tracking-Benutzerstudie können große Datenmengen anfallen. Für die Datenverwaltung verwendet eTaddy ein Datenmodell, das am Institut für Visualisierung und Interaktive Systeme (VIS) für das Speichern von Eye-Tracking-Benutzerstudien erstellt wurde. Auf die dort abgespeicherten Benutzerstudien können auch andere Programme vom VIS zugreifen. Daher ist es sinnvoll dieses Datenmodell auch im Web Service zu verwenden. Um die Größe der versendeten SOAP-Nachrichten klein zu halten, können in den eingehenden Nachrichten anstelle der Rohdaten nur Referenzen auf die Einträge der Datenbank

verwendet werden. Die Web Service-Operationen müssen dann, bevor ein Plug-in aufgerufen werden kann, die Referenzen auflösen und die entsprechenden Datensätze aus der Datenbank auslesen. In den SOAP-Response-Nachrichten ist es allerdings sinnvoll die Ergebnisse der Berechnungen einzubinden, damit diese vom Workflow ausgelesen werden können und der Kontrollfluss darauf reagieren kann.

6.3.2 Einheitliche Schnittstelle

Die Plug-ins in eTaddy erwarten eine unterschiedliche Anzahl an Probanden (siehe Tabelle 3.1). Eine direkte Abbildung der erwarteten Anzahl der Probanden auf die Web Service-Operationen würde diese nur unnötig komplex machen. Daher kann von den Plug-ins abstrahiert werden, indem die Web Service-Operationen eine beliebige Anzahl an Probanden erlauben. Plug-ins, die nur einen Probanden erwarten, können transparent mehrfach ausgeführt werden. Damit können alle Web Service-Operationen, welche die Funktionalität der Plug-ins umsetzen, angesprochen werden, indem der Name der Operation, eine Referenz auf einen Stimulus und eine Liste von Referenzen auf Probanden übergeben werden. Durch diese einheitliche Schnittstelle wird die Modellierung des Analyseprozesses als Workflow vereinfacht.

6.3.3 Modularität und Wiederverwendbarkeit

eTaddy ist darauf ausgelegt, dass neue Funktionalität in Form von Plug-ins hinzugefügt wird. Durch die Plug-ins wird die Implementierung der Metriken, Statistiken und Visualisierungen gekapselt und die Implementierung kann beliebig erweitert, entfernt oder ausgetauscht werden. Der Web Service übernimmt die Eigenschaft der Modularität, indem ein Plug-in als eine Web Service-Operation zur Verfügung gestellt wird. Durch die Entwicklung von einem neuen Plug-in wird sowohl eTaddy, als auch der Web Service um neue Funktionalität erweitert. Die Web Service-Operationen abstrahieren durch die im vorherigen Abschnitt beschriebene einheitliche Schnittstelle noch weiter und sind daher nicht von den Plug-ins abhängig. So kann die Implementierung der Web Service-Operationen komplett ausgetauscht werden.

6.4 Gesamtsystem

In Abbildung 6.1 ist das Lösungskonzept abgebildet. Ein Web Service stellt die Funktionalität von eTaddy als Web Service-Operationen zur Verfügung. Die Funktionalität in eTaddy ist in Form eines Plug-in-Systems realisiert, das von eTaddy und dem Web Service implementiert wird. Außerdem greift der Web Service auf die selbe Datenbank wie eTaddy zu, um Eye-Tracking-Daten zur Berechnung auszulesen. Durch Abbilden der einzelnen Analyseschritte als Web Service-Operationen kann der Analyseprozess einer Eye-Tracking-Benutzerstudie als Workflow modelliert werden.

Die Workflowmodelle können geteilt werden, so dass sie von mehreren Personen benützt und verbessert werden können. Wiederkehrende Arbeitsschritte bei der Analyse können gespeichert und auf andere Eye-Tracking-Benutzerstudien angewandt werden.

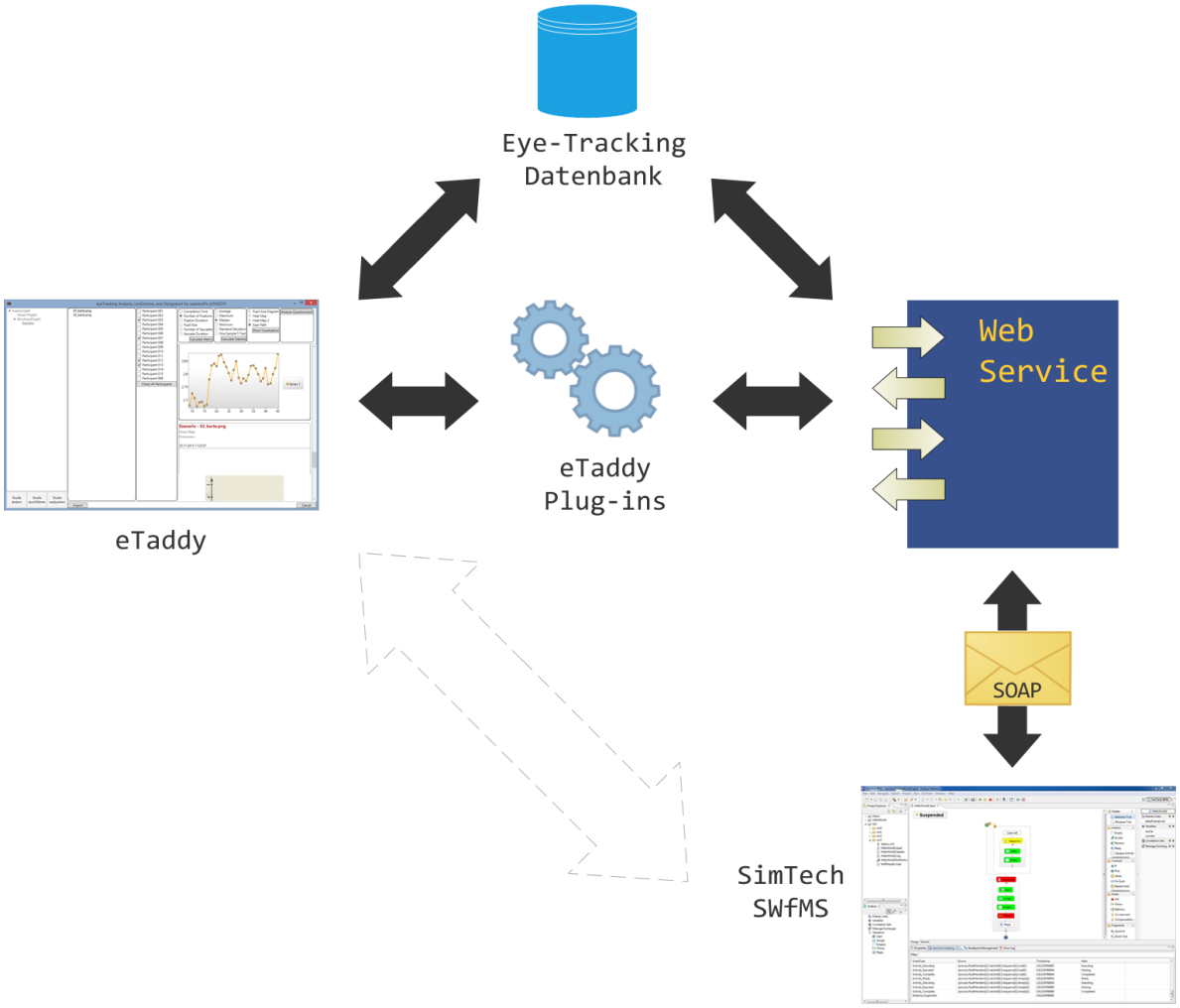

Abbildung 6.1: Um die Analyse von Eye-Tracking-Benutzerstudien zu automatisieren, werden die Funktionen von eTaddy als Web Service-Operationen zur Verfügung gestellt. Die Funktionalität von eTaddy ist in Form von Plug-ins realisiert, die der Web Service einbindet. eTaddy und der Web Service greifen auf die selbe Datenbank zu.

7 Implementierung

In diesem Kapitel wird der prototypisch implementierte Web Service vorgestellt. Zunächst wird ein Überblick über die Architektur von eTaddy gegeben. Anschließend wird die Wahl des Web Service-Frameworks und dessen Implementierung erklärt. In den folgenden Abschnitten wird auf die Datenverwaltung und die Serialisierung der Parameter und Rückgabewerte eingegangen. Daraufhin wird die Umsetzung der Plug-ins von eTaddy als Web Service-Operationen erklärt. Weiterhin wird die Modularität und Wiederverwendbarkeit des Web Services anhand eines neuen Plug-ins gezeigt. Außerdem wird auf die Fehlerbehandlung innerhalb der Web Service-Operationen eingegangen. Im letzten Abschnitt wird beschrieben, wie der prototypisch implementierte Web Service auf einer virtuellen Maschine aufgesetzt wurde. Abschließend wird die Implementierung nochmal zusammengefasst.

7.1 Analyse von eTaddy

In Abschnitt 3.1 wurde bereits die Funktionalität und die Benutzeroberfläche von eTaddy vorgestellt. In den folgenden Abschnitten wird auf die Implementierungsdetails eingegangen. Die Analyse der Architektur von eTaddy wurde vor der eigentlichen Implementierung des Web Services durchgeführt.

7.1.1 Architektur

eTaddy ist in der Programmiersprache C# geschrieben und baut auf die Windows Presentation Foundation[1] (WPF) auf. Die Benutzeroberfläche ist in der auf XML basierenden Sprache Extensible Application Markup Language[2] (XAML) geschrieben und verwendet Model View ViewModel (MVVM) als Architekturmuster, um die Benutzeroberfläche mit dem Programmcode zu verbinden. Für die Datenhaltung wird ein Microsoft SQL Server[3] eingesetzt.

Als Entwicklungsumgebung von eTaddy wurde Microsoft Visual Studio verwendet. eTaddy selbst liegt als Visual Studio Solution, einer Art Projektmappe, mit mehreren Unterprojekten vor. In Abbildung 7.1 ist der Solution Explorer abgebildet, der die Solution mit ihren Unterprojekten zeigt. Durch die Unterteilung von eTaddy in einzelne Projekte für die Anwendungslogik, die Plug-ins, die Benutzeroberfläche und das Datenmodell können diese getrennt voneinander wiederverwendet werden. Der prototypisch implementierte Web Service verwendet lediglich die Plug-ins und das Datenmodell.

[1] http://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx
[2] http://msdn.microsoft.com/en-us/library/ms752059%28v=vs.110%29.aspx
[3] http://www.microsoft.com/en-us/sqlserver/default.aspx

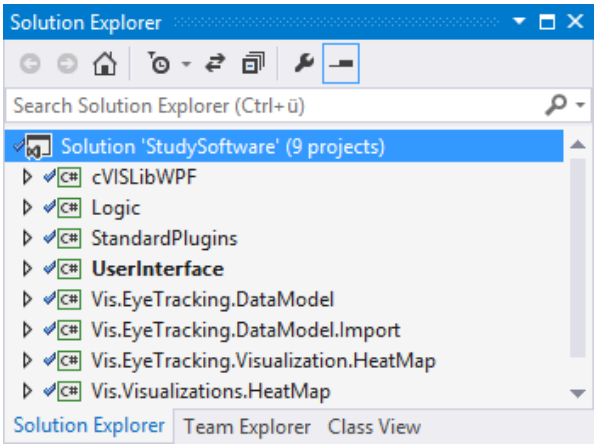

Abbildung 7.1: eTaddy ist in verschiedene Unterprojekte unterteilt. Dazu gehört die Benutzeroberfläche (UserInterface), die Anwendungslogik (Logic), die Plug-ins (StandardPlugins), das Datenmodell (Vis.EyeTrackingDataModel) und eine Heatmap-Visualisierung (Vis.Visualizations.HeatMap).

7.1.2 Datenmodell

Das von eTaddy verwendete Datenbankmodell (siehe Abbildung 7.2) wird auch bei anderen Eye-Tracking-Projekten am Institut für Visualisierung und Interaktive Systeme (VIS) eingesetzt. Somit können alle diese Eye-Tracking-Projekte Daten austauschen. Allerdings müssen die von einem Eye-Tracker aufgenommenen Daten erst in das Format der Datenbank konvertiert werden, bevor sie ausgewertet werden können. Für den Eye-Tracker der Firma Tobii wurde daher ein Importer geschrieben, der nach dem Durchführen einer Studie die gesammelten Daten in die Datenbank einspeist.

7.1.3 Plug-in-System

Die Metriken, Statistiken und Visualisierungen zur Auswertung von Eye-Tracking-Benutzerstudien wurden in eTaddy als Plug-ins realisiert. So lässt sich das Programm einfach um neue Funktionen erweitern. Für jede der drei Plug-in-Kategorien steht ein Interface zur Verfügung, das von den Plug-ins implementiert werden muss. Das Interface legt fest, dass jedes Plug-in einen Namen, die Anzahl der erwarteten Probanden und eine Methode zur eigentlichen Berechnung zur Verfügung stellt. Die Methode zur Berechnung erwartet eine Liste von Probanden und einen Stimulus. In Tabelle 3.1 sind alle implementierten Metriken, Statistiken und Visualisierungen mit den erwarteten Eingaben und den Ausgaben aufgeführt. In Abbildung 7.3 sind die Plug-ins abgebildet, wie sie im Solution Explorer angezeigt werden.

Die Plug-ins erwarten eine unterschiedliche Anzahl von Probanden, da als Ergebnis bei den Metriken und Statistiken immer eine zweispaltige Tabelle erzeugt wird. Manche Metriken generieren nur einen Ergebniswert pro Proband und Stimulus, andere mehrere. So wird von der Metrik „Fixationsanzahl“ nur der Probandenname (1. Spalte) und die Anzahl (2. Spalte) zurückgegeben. In diesem Fall erlaubt eTaddy die Auswahl von mehreren Probanden. Dagegen gibt die Metrik „Fixationsdauer“ mehrere Indizes (1. Spalte) und dazugehörige Zeiten

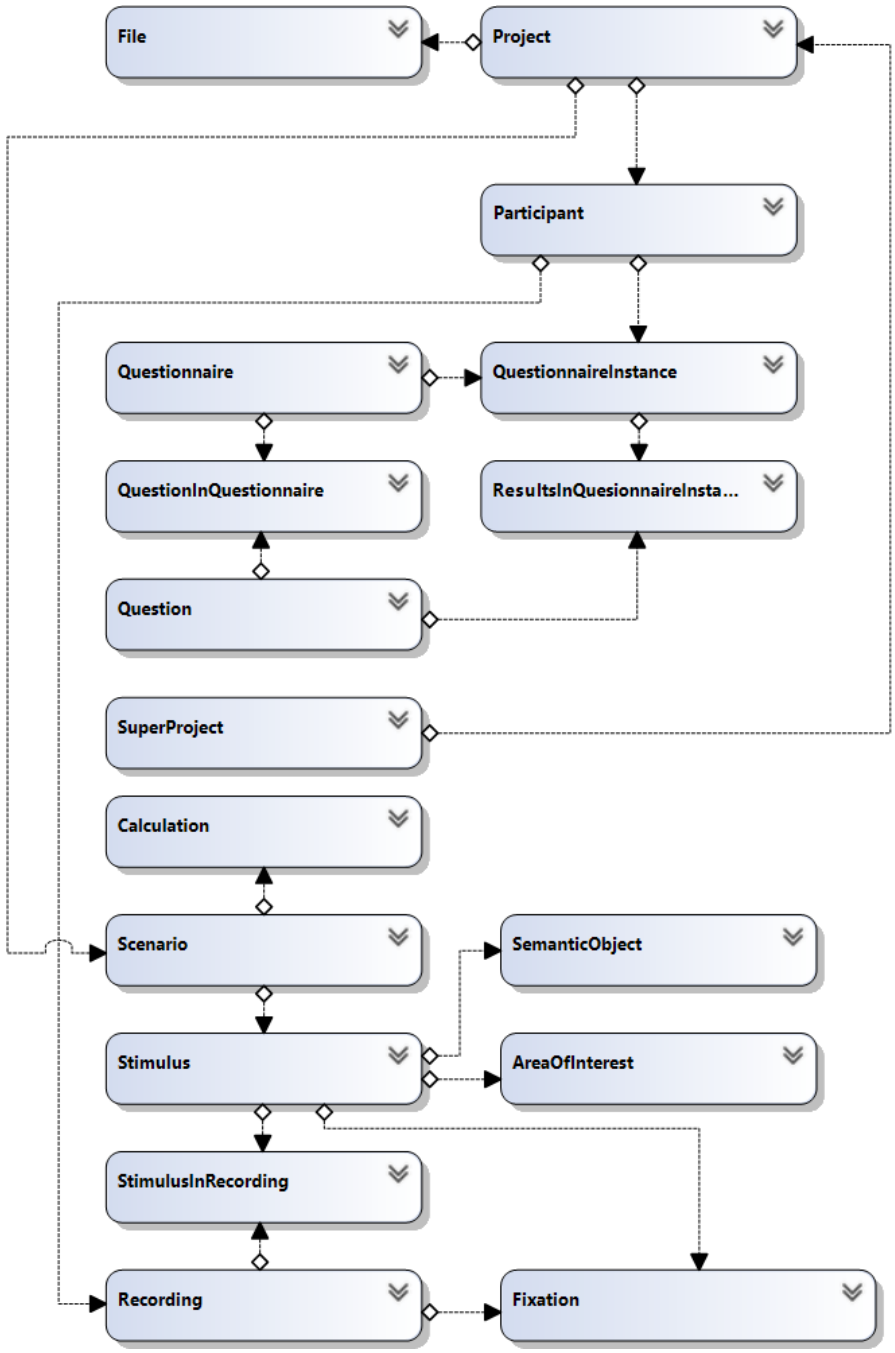

Abbildung 7.2: Das von eTaddy verwendete VIS-Datenmodell für Eye-Tracking-Benutzerstudien. Wichtig für diese Arbeit sind die Tabellen der Projekte (Project), der Probanden (Participant) und der Stimuli (Stimulus). Die eTaddy Plug-ins verwenden als Grundlage für die Berechnungen die Fixationen (Fixation).

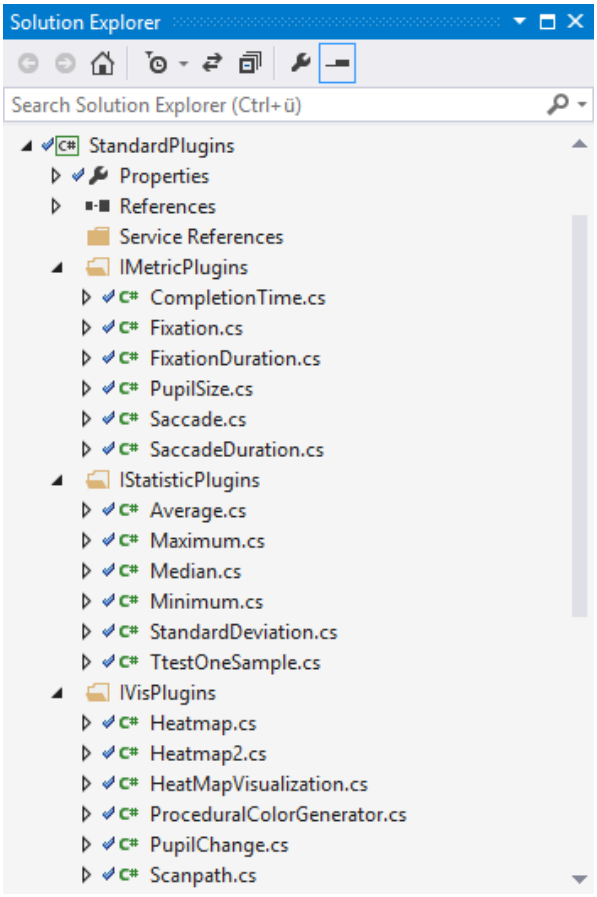

Abbildung 7.3: Die Plug-ins in eTaddy sind in die Kategorien Metriken (IMetricPlugins), Statistiken (IStatisticPlugins) und Visualisierungen (IVisPlugins) unterteilt.

(2. Spalte) aus. Hier kann nur ein Proband ausgewählt werden. Auch die Visualisierungen unterscheiden sich in der Anzahl der erwarteten Probanden. Die Heatmap wird immer von allen Probanden eines Stimulus generiert. Das Scanpath Plug-in kann mehrere Probanden gleichzeitig einzeichnen. In dem Pupillengrößendiagramm kann immer nur ein Proband visualisiert werden.

Alle Berechnungen der Statistik-Plug-ins werden in R[4], einer freien Programmiersprache für statistisches Rechnen und statistische Grafiken, durchgeführt. R ist für eine Vielzahl von UNIX Systemen, Windows und Mac OS verfügbar. Die Sprache wird häufig in den Bereichen des Data Mining, statistischer Software und Datenanalyse eingesetzt. Um R in C# verwenden zu können, wird R.NET[5] als Wrapper eingesetzt.

[4] http://www.r-project.org/
[5] http://rdotnet.codeplex.com/

7.2 Web Service

Der erste Schritt bei der Implementierung des Web Service war die Wahl eines geeigneten Web Service-Frameworks. In den folgenden Abschnitten wird erklärt, wie die Auswahl getroffen wurde.

7.2.1 Wahl des Web Service-Frameworks

Das von Microsoft entwickelte Framework ASP.NET[6] erlaubt, dynamische Webanwendungen in allen von .NET[7] unterstützten Programmiersprachen zu erstellen. Dazu gehört C#, die Implementierungssprache von eTaddy. Zusätzlich existiert eine SOAP-Framework-Erweiterung für ASP.NET, welche die Verarbeitung von SOAP-Nachrichten erlaubt. Aufgrund der guten Integration mit der Implementierungssprache von eTaddy wurde ASP.NET als Framework für den Web Service gewählt.

7.2.2 Erstellen des Web Services

Das Erstellen einer ASP.NET Webanwendung in Visual Studio ist denkbar einfach. In dieser Arbeit wurde die Webanwendung als ein weiteres Projekt in die Solution von eTaddy hinzugefügt. Diese dichte Kopplung an eTaddy hat den Vorteil, dass der Benutzer die Wahl hat eTaddy mit einer grafischen Benutzeroberfläche oder als Web Service zu starten. Um ein ASP.NET Projekt hinzuzufügen, muss zunächst im Kontextmenü der Solution im Solution Explorer das Menü „Add“ und „New Project...“ ausgewählt werden. Im folgenden Dialogfenster (siehe Abbildung 7.4) muss anschließend die Implementierungssprache gewählt werden. Das ist im Fall von eTaddy „Visual C#“. Als Projektyp wird daraufhin aus der Kategorie „Web“ der Eintrag „ASP.NET Web Application“ ausgewählt. Als letzter Schritt wird noch ein Name für den Service vergeben. Angelehnt an den Namen der Solution von eTaddy („StudySoftware“) wurde der Service „StudyService“ genannt. Da ASP.NET Teil des .NET Frameworks ist, muss eine .NET Frameworkversion angegeben werden. Um den Web Service später ausführen zu können, muss diese Frameworkversion auf dem Hostrechner installiert sein. Die Struktur des neu erstellten ASP.NET Projekts wird im folgenden Abschnitt vorgestellt.

7.2.3 Struktur einer ASP.NET Webanwendung

Beim Erstellung einer ASP.NET Webanwendung wird ein neues Projekt in der Visual Studio-Solution erzeugt (siehe Abbildung 7.5). Dabei werden drei Dateien automatisch erzeugt, die für die Implementierung wichtig sind. Zum einen die „Web.config“ Datei, die Informationen für das Deployment des Web Services enthält. Dazu gehört unter anderem die für die Ausführung notwendige Version des .NET Frameworks. Die „Web.config“ Datei basiert auf XML und kann daher einfach gelesen und bearbeitet werden. Allerdings wird sie bereits so generiert, dass der Web Service ohne weitere Anpassungen gestartet werden kann. Die zweite erstellte Datei trägt den Namen des Projektes mit der Dateiendung „.asmx“. In diesem Fall

[6] http://www.asp.net/
[7] http://msdn.microsoft.com/en-US/vstudio/aa496123

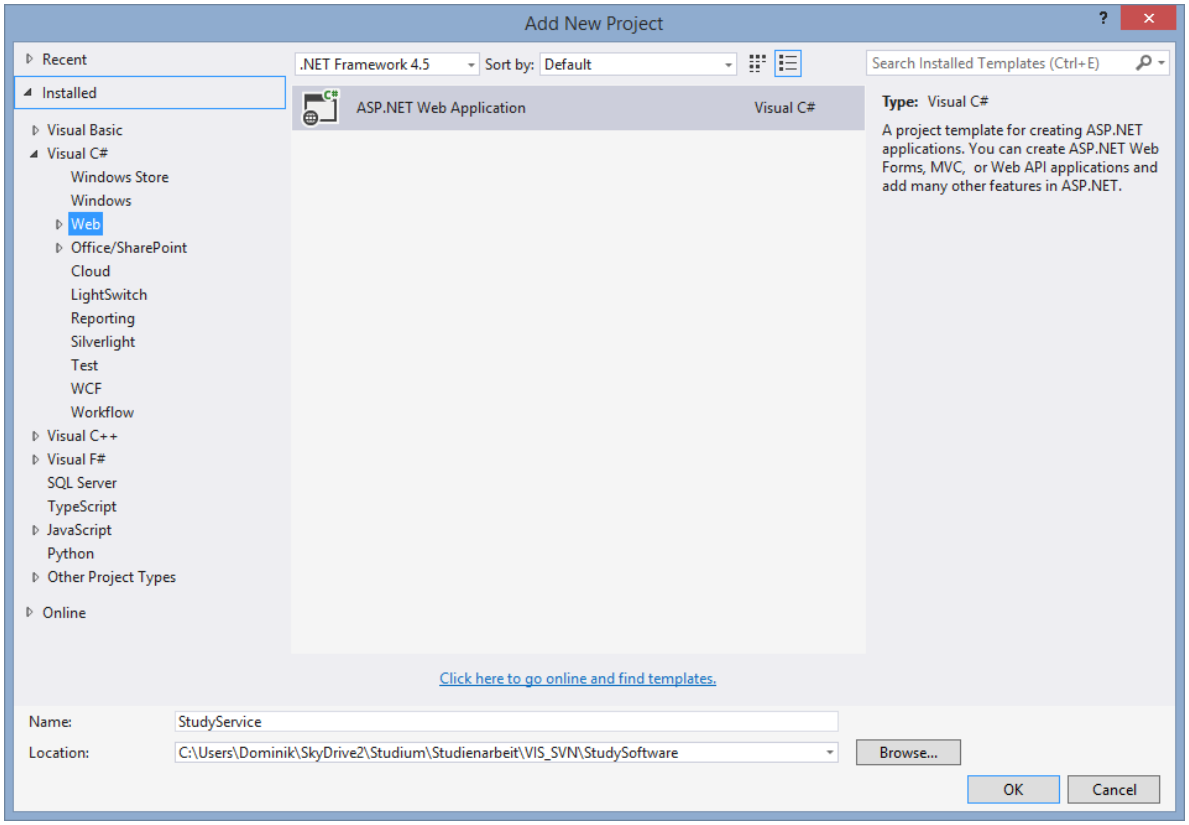

Abbildung 7.4: Das Dialogfenster, um ein neues Projekt zu einer Solution hinzuzufügen. Ausgewählt ist als Implementierungssprache „Visual C#“ und als Projekttyp „ASP.NET Web Application“. Als Name des neuen Projekts ist „StudySoftware“ eingetragen. Der Speicherpfad zeigt standardmäßig auf die Solution, kann aber beliebig angepasst werden.

wurde eine Datei mit dem Namen „StudyService.asmx“ generiert. Die Datei enthält nur eine Anweisung:

```
1 <%@ WebService Language="C#" CodeBehind="StudyService.asmx.cs"
2                Class="WebService.StudyService" %>
```

ASP.NET Webanwendungen können, wie bereits im vorherigen Abschnitt beschrieben, in allen von .NET unterstützten Programmiersprachen implementiert werden. In dem oben aufgeführten Ausdruck wird daher im „Language“ Attribut die verwendete Sprache festgelegt. Das Attribut „CodeBehind“ definiert die Datei, in der die eigentliche Implementierung des Web Service zu finden ist (StudyService.asmx.cs). Da als Programmiersprache C# zum Einsatz kommt, hat die Datei die Dateiendung „.cs“ für „C Sharp“ (C#). Innerhalb der Datei befindet sich die Klasse „StudyService“ im Namespace „WebService“. Dies wird mit dem „Class“ Attribut festgelegt.

```
1  [WebService(Namespace = "http://www.iaas.uni-stuttgart.de/sa/eyetracking")]
2  public class StudyService : System.Web.Services.WebService
3  {
4
5     [WebMethod(Description = "A simple Hello World example.")]
6     public String HelloWorld()
7     {
8        return "Hello World";
9     }
10
11 }
```

Listing 7.1: Durch Annotieren einer C#-Klasse und den Klassenmethoden kann nach dem Bottom-Up-Prinzip ein Web Service generiert werden.

Die Klasse „StudyService“ in der Datei „StudyService.asmx.cs“ hat eine für C# übliche Struktur. Nach dem Bottom-Up Prinzip für das Erstellen eines Web Services (siehe Abschnitt 2.2.4), wird die Klasse und deren Methoden annotiert. Die Annotationen werden vom Web Serive-Framework verwendet, um die Methoden als Web Service-Operationen zur Verfügung zu stellen. Im Listing 7.1 wird ein Web Service mit dem Namen „StudyService“ (Zeile 2) und dessen Namespace (Zeile 1) definiert. Die Methode „HelloWorld“ (Zeile 6) wurde mit „WebMethod“ und einer optionalen Beschreibung annotiert (Zeile 5), damit sie das Web Service-Framework als Web Service-Operation zur Verfügung stellen kann. Der Name der annotierten Methode entspricht dann dem Namen der Web Service-Operation. Die Parameter der Methode entsprechen den Parametern einer eingehenden SOAP-Nachricht. In gleicher Weise entspricht der Rückgabewert der SOAP-Antwort. Für primitive Datentypen wird ein automatisches Mapping von C#- auf XML-Schema-Datentypen, und umgekehrt, vorgenommen. Die „HelloWorld“ Beispieloperation gibt einen XML-Schema String zurück, da der Rückgabewert der Methode ein C#-String ist (Zeile 6 und 8). Durch die Annotationen kann ASP.NET automatisch eine Servicebeschreibung in Form eines WSDL-Dokuments generieren.

7.2.4 Verbindung mit eTaddy

Das neu hinzugefügte ASP.NET Projekt hat, obwohl es sich in der selben Solution wie die Plug-ins und das Datenmodell befindet, nach dem Erstellen noch keinen Zugriff auf die anderen Projekte von eTaddy. Um den Zugriff herzustellen, müssen die Projekte vom ASP.NET Projekt aus referenziert werden. Dafür müssen die Projekte von eTaddy im Kontextmenü des „StudyService“ Projekts unter dem Menüpunkt „Add“ und „References...“ hinzugefügt werden. Danach sind das Datenmodell und die Implementierungen der Plug-ins innerhalb der „StudyService.asmx.cs“ Datei sichtbar und können instanziiert werden.

7.2.5 Datenverwaltung

Um auf die Daten in der Datenbank zugreifen zu können, muss vom Web Service zunächst eine Verbindung zur Datenbank hergestellt werden. Dies geschieht im Konstruktor der „Stu-

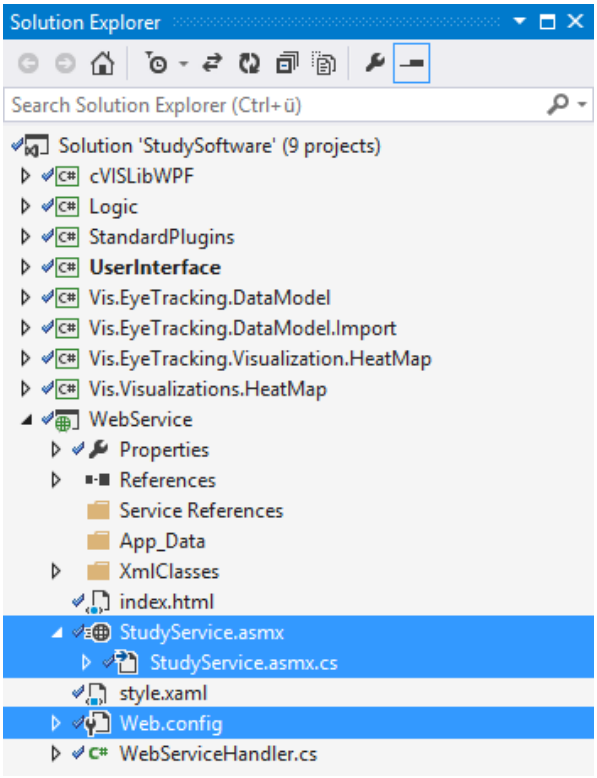

Abbildung 7.5: Der Solution von eTaddy wurde ein WebService-Projekt hinzugefügt. Dabei wurden die Dateien „Web.config“, „StudyService.asmx“ und „StudyService.asmx.cs“ automatisch erzeugt. Die „Web.config“ Datei enthält Informationen zum Deployment. Die „Study.Service.asmx“ Datei verweist auf die „StudyService.asmx.cs“, in der sich die eigentliche Implementierung des Web Services befindet.

dyService“ Klasse. Bei einer Anfrage an den Web Service wird die Klasse instanziiert und der Konstruktor der Klasse aufgerufen. Die Verbindung zur Datenbank wird in einer Klassenvariable gespeichert, die anschließend von allen Methoden, und damit allen Web Service-Operationen, verwendet werden kann. Im Prototyp sind die Verbindungseinstellungen fest in den Konstruktor eingetragen.

7.2.6 Serialisierung der Parameter und Rückgabewerte

Primitive Datentypen werden in ASP.NET automatisch von C#- in XML-Schema-Datentypen, und umgekehrt, übersetzt. Die meisten Web Service-Operationen erwarten allerdings nicht nur primitive Datentypen als Parameter und Rückgabewerte, sondern auch komplexere Strukturen. Damit auch hier ein Mapping möglich ist, wurden in C# geschriebene Hilfsklassen angelegt. In Listing 7.2 ist eine beispielhafte Hilfsklasse mit dem Namen „Participant“ aufgeführt. Die Hilfsklasse besitzt nur Klassenvariablen und `get/set`-Methoden. Die Klasse und die Klassenvariablen werden als „XmlRoot“ (Zeile 1) beziehungsweise „XmlElement“ (Zeile

```
1  [XmlRoot(ElementName = "Participant")]
2  public class Participant
3  {
4
5     [XmlElement(Type = typeof(string), ElementName = "Forename")]
6     public string Forename { get; set; }
7
8     [XmlElement(Type = typeof(string), ElementName = "Surname")]
9     public string Surname { get; set; }
10
11    [XmlElement(Type = typeof(int), ElementName = "Age")]
12    public int Age { get; set; }
13
14 }
```

Listing 7.2: Eine C#-Hilfsklasse, die mit XML-Annotationen versehen wurde. Der Klassenname wird als XML-Root (XmlRoot) gekennzeichnet und die Klassenvariablen als Kindelemente (XmlElement).

5, 8 und 11) annotiert. Bei den Klassenvariablen ist es zusätzlich notwendig einen Datentyp anzugeben, um eine Konvertierung zu einem XML-Schema-Datentyp zu ermöglichen.

In Listing 7.3 ist eine beispielhafte WebMethod mit dem Namen „GetParticipant“ (Zeile 6) und einem Rückgabewert vom Datentyp „Participant“ (Zeile 6 und 14), mit den Variablen „Forename“ (Zeile 10), „Surname“ (Zeile 11) und „Age“ (Zeile 12) gesetzt auf „Max“, „Mustermann“ und „42“ aufgeführt. Bei der Rückgabe des Objekts wird es automatisch in XML serialisiert und in den Body der SOAP-Nachricht eingefügt. Aus der Klasse wird ein XML-Schema ComplexType generiert, welcher in der von ASP.NET erstellten WSDL-Datei innerhalb des Type-Elements eingebunden wird. Auf diese Art ist es auch möglich Listen zu serialisieren, etwa um eine Liste von allen Probanden zurückzugeben. Listen werden als XML-Schema Sequence umgesetzt. Das Ergebnis der Serialisierung aus Listing 7.3 ist in Listing 7.4 aufgeführt.

Umgekehrt kann eine WebMethod das Objekt einer annotierten Klasse als Paramter erwarten. In diesem Fall wird der Body der eingehenden SOAP-Nachricht extrahiert und automatisch ein Mapping von der XML Struktur auf eine C#-Hilfsklasse durchgeführt. Der WebMethod wird dann eine Instanz der jeweiligen Hilfsklasse als Parameter übergeben.

7.3 Bereitstellung der eTaddy-Plug-ins als Web Service-Operationen

Die folgenden Abschnitten beschreiben, wie die Plug-ins von eTaddy als Web Service-Operationen zur Verfügung gestellt wurden und welche Herausforderungen bei der Implementierung aufgetreten sind.

```
1  [WebService(Namespace = "http://www.iaas.uni-stuttgart.de/sa/eyetracking")]
2  public class StudyService : System.Web.Services.WebService
3  {
4
5     [WebMethod(Description = "Returns one participant.")]
6     public Participant GetParticipant()
7     {
8        Participant participant = new Participant()
9        {
10          Forename = "Max",
11          Surname  = "Mustermann",
12          Age      = 42
13       };
14       return participant;
15    }
16
17 }
```

Listing 7.3: Eine beispielhafte Web Service-Implementierung, mit einer Operation „GetParticipant“. Die Operation gibt ein Objekt einer mit XML-Annotationen versehenen Hilfsklasse zurück. Das Objekt wird anhand der Annotationen in eine XML-Struktur serialisiert.

```
1  <Participant>
2     <Forename>Max</Forename>
3     <Surname>Mustermann</Surname>
4     <Age>42</Age>
5  </Participant>
```

Listing 7.4: Beispielhafte Serialisierung eines Objekts der Klasse „Participant“ in XML.

7.3.1 Einheitliche Schnittstelle

Wie bereits in Abschnitt 3.1 erwähnt, erwarten die verschiedenen Plug-ins von eTaddy unterschiedlich viele Probanden als Parameter. Diese Einschränkung ist auf die grafischen Benutzeroberfläche zurückzuführen, da es sich bei den Ergebnissen der Berechnungen immer um Tabellen mit zwei Spalten handelt. Bei der Implementierung der Web Service-Operationen existiert dieses Problem nicht und die unterschiedlichen Parameter würden den Aufruf der Operationen unnötig komplex gestalten. Daher wird im Web Service von den zugrundeliegenden Plug-ins abstrahiert. Alle Web Service-Operationen erwarten als Paramter einen Stimulus und beliebig viele Probanden. Der Web Service kümmert sich um die eventuell notwendige Mehrfachausführung von Plug-ins, die nur einen Proband erwarten. Die Ergebnisse der Berechnungen werden konkateniert und wie in Abschnitt 7.2.6 beschrieben als Liste serialisiert.

```
1 Stimulus stimulus = (from s in Database.Stimulus
2                      where s.Id == Guid.Parse(input.StimulusId)
3                      select s).First();
```

Listing 7.5: Instanziierung eines LINQ-Objekts.

7.3.2 Parameter

Die Plug-ins erwarten als Parameter Objekte, die von Language Integrated Query[8] (LINQ) erzeugt wurden. LINQ ist eine .NET-Komponente, die .NET-Sprachen um Datenbankabfragen erweitert. Außerdem ermöglicht LINQ das Anlegen von Objekten, die Tupel in der Datenbank entsprechen. Wird ein Attribut des Objekts geändert, wird automatisch die Spalte in der Datenbank angepasst. In Listing 7.5 wird ein Objekt vom Typ „Stimulus“ erzeugt, das einem Eintrag aus der Datenbanktabelle „Stimulus“ mit der ID „StimulusId“ entspricht.

Da die Web Service-Operationen nur die IDs von den Stimuli und den Probanden erhalten, ist als erster Schritt der Bearbeitung immer ein Mapping von den IDs auf ein LINQ-Objekt notwendig.

7.3.3 Metriken

Um die Metriken als Web Service-Operationen zur Verfügung zu stellen, wurde für jedes Plug-in, welches das IMetricPlugin-Interface implementiert, eine Methode als WebMethod annotiert. In Abbildung 7.6 ist der Aufbau der „GetFixationDuration“ WebMethod exemplarisch als Nassi-Shneiderman-Diagramm aufgeführt. Als Parameter erhalten die Methoden ein Objekt, das anhand der annotierten Hilfsklassen aus dem Body der eingehenden SOAP-Nachricht erstellt wird. Bei den Metriken beinhaltet dieses Objekt die ID des Stimulus und eine Liste der IDs von Probanden. Der erste Schritt bei der Verarbeitung ist daher eine Datenbankabfrage, um die passenden LINQ-Objekte zu den IDs zu erhalten. Anschließend wird das Plug-in der Metrik instanziiert und die „Calculate“ Methode mit den LINQ-Objekten aufgerufen. Erwartet das Plug-in nur einen Probanden, so wird die „Calculate“ Methode für jeden Probanden in der Liste erneut aufgerufen. Mit den Ergebnissen der Berechnungen wird ein Hilfsklassen-Objekt initialisiert und von der Methode zurückgegeben. Das Objekt wird anschließend automatisch in XML serialisiert und in den Body der ausgehenden SOAP-Nachricht geschrieben.

7.3.4 Statistiken

Die Statistiken wurden ähnlich wie die Metriken implementiert. Ein zusätzlicher Bearbeitungsschritt ist aber die Ausführung der entsprechenden Metrik, bevor deren Ergebnis an das Statistik-Plug-in übergeben wird. Auch hier wird abhängig davon, wie viele Probanden von der Metrik erwartet werden, eine Reihe von Mehrfachausführungen der Metrik und anschließend der Statistik vorgenommen.

[8] http://msdn.microsoft.com/en-us/library/vstudio/bb397926.aspx

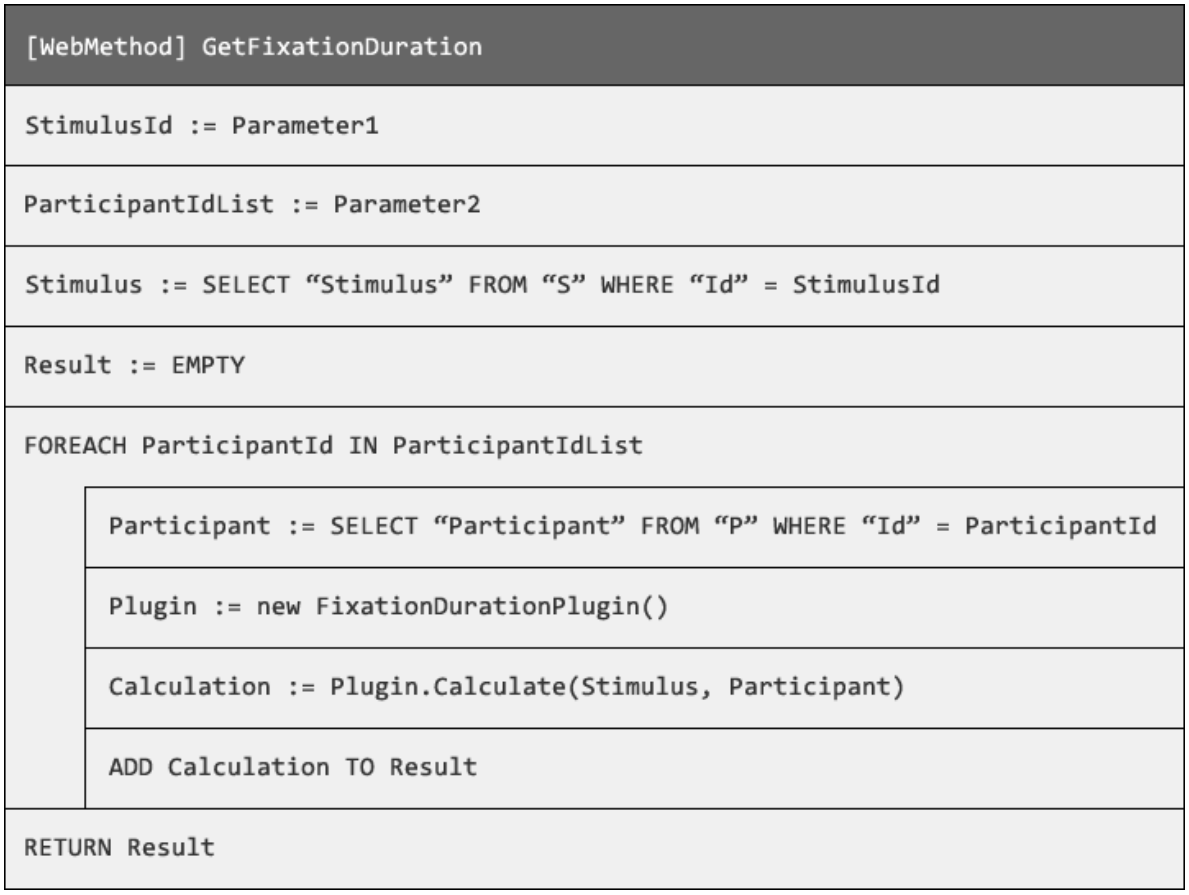

Abbildung 7.6: Nassi-Shneiderman-Diagramm der „GetFixationDuration“ WebMethod. Da das „FixationDuration“ Plug-in nur einen Probanden als Parameter erwartet, wird es für jeden Probanden einzeln aufgerufen.

Die Berechnung der Statistiken erfolgt in den Plug-ins mit der Statistikprogrammiersprache R, die mit Hilfe von R.NET in das Projekt eingebunden ist, und somit eine Schnittstelle in C# besitzt. Die Statistiken erwarten eine Instanz von R.NET im globalen Namensraum. Daher war es notwendig, R.NET vor dem Aufruf der Statistik-WebMethods zu instanziieren. Dies geschieht, wie auch das Herstellen der Datenbankverbindung, im Konstruktor der „StudyService“ Klasse. Dabei ist zu beachten, dass von R.NET immer nur eine Instanz gleichzeitig aktiv sein darf. Daher war eine Abfrage notwendig, die überprüft, ob bereits eine Instanz existiert und gegebenenfalls diese wiederverwendet. Somit ist es trotz dieser Einschränkung möglich, mehrere Anfragen gleichzeitig an den Web Service zu stellen.

7.3.5 Visualisierungen

Die Implementierung der Visualisierungen als Web Service-Operationen stellte eine größere Herausforderung dar. Die „Calculate“ Methode der Plug-ins, die das IVisPlugin Interface implementieren, gibt ein „ContentControl“ Objekt zurück. Dabei handelt es sich um eine

grafische Komponente der Benutzeroberfläche und ist Teil von WPF. In eTaddy wurden die Objekte, und damit die berechneten Visualisierungen, direkt in die Timeline eingefügt. Im Web Service konnten diese Objekte nicht verwendet werden, da sich die Typen der Threads von grafischen Komponenten und ASP.NET-Anwendungen unterscheiden. Die grafischen Komponenten laufen in so genannten „Single Thread Apartment“ (STA) Threads, während ASP.NET Anwendungen in „Multi Thread Apartment“ (MTA) Threads laufen. Deshalb wurde nach der Anleitung des Artikels „Creating STA COM compatible ASP.NET Applications“[9] der Web Service Handler umgeschrieben, dass er als STA-Thread startet. Somit ist es möglich, Komponenten aus einer Benutzeroberfläche in den WebMethods zu erstellen.

Damit die Web Service-Operationen die grafischen Komponenten zurückgeben können, werden aus den „ContentControl“-Objekten PNG-Bilder gerendert. Dafür werden sie zunächst auf ein programmatisch angelegtes Canvas-Element eingezeichnet und dieses wird anschließend gerendert. Die nächste Herausforderung stellte das Einbinden der PNG-Bilder in die SOAP-Nachricht dar. Um dies zu ermöglichen, werden die PNG-Bilder in einen Byte-Array überführt. ASP.NET serialisiert Byte-Arrays automatisch als Base64-String, wenn in der für die Rückgabe verwendeten Hilfsklasse die Klassenvariable als XML-Element vom Typ Byte-Array annotiert wird.

Dieses Vorgehen konnte für das Heatmap- und das Scanpath-Plug-in eingesetzt werden. Bei der Implementierung der Operation für die Visualisierung der Pupillengröße im Verlauf der Zeit (siehe Abbildung 7.7) wurden allerdings nur leere Diagramme generiert. Das Plug-in verwendet für die Darstellung eine WPF-Chart Komponente. Diese wurde für das Einbinden in die Benutzeroberfläche entwickelt und blendet die Punktwerte im Diagramm mit einer kurzen Animation ein. Beim Rendering als PNG-Bild wird das erste Frame der Animation gewählt, in dem noch keine Punkte sichtbar sind. Die Animation der WPF-Chart Komponente kann nicht einfach deaktiviert werden. Vielmehr musste der interne Style überschrieben werden. Da das Überschreiben erfolgen muss, bevor die Komponente instanziiert wird, war hier zum ersten Mal eine Änderung an einem Plug-in notwendig. Des Weiteren sind die Punktwerte im Diagramm mit einer Linie verbunden. Auch diese wurde beim Rendering nicht angezeigt. Der Grund dafür war, dass die Linie erst gezeichnet wird, wenn das WPF-Chart in die Benutzeroberfläche eingebunden und angezeigt wird. Deshalb wird von der WebMethod programmatisch ein Window-Objekt erzeugt, in welches das Diagramm eingebunden wird und somit das Event zum Zeichnen auslöst. Erst so war es möglich, das Diagramm genau wie die anderen Visualisierungen als PNG-Bild zu rendern und anschließend in einen Base64-String zu serialisieren.

7.3.6 Weitere Operationen

Da die Metriken, Statistiken und Visualisierungen die IDs von den Stimuli und den Probanden erwarten, war es notwendig, weitere Operationen als Einstiegspunkte zu erstellen, die alle Stimuli und alle Probanden zurückgeben.

[9] http://weblog.west-wind.com/posts/2012/Sep/18/Creating-STA-COM-compatible-ASPNET-Applications

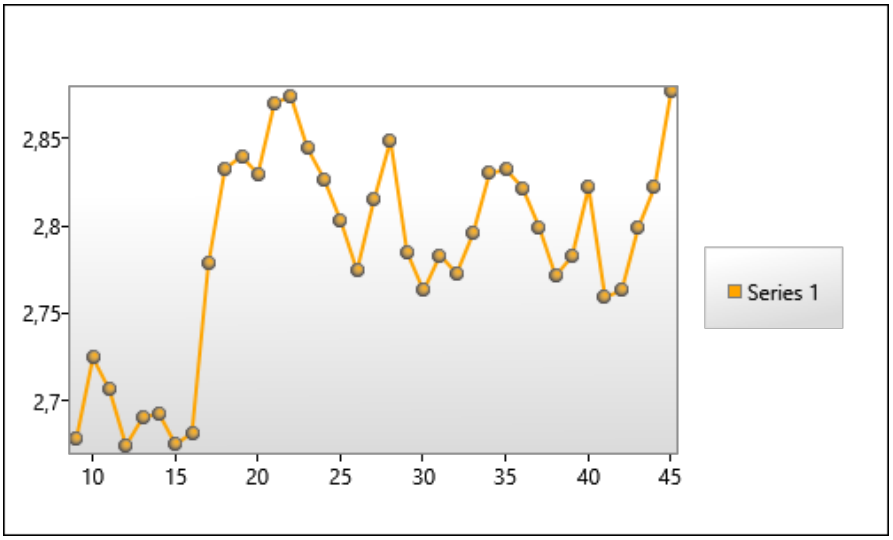

Abbildung 7.7: Das Diagramm zur Visualisierung der Pupillengröße im Verlauf der Zeit. Die Punkte und die Verbindungslinien wurden mit einer Animation eingeblendet und waren beim Rendern des Diagramms als PNG-Bild nicht zu sehen, da hierbei das erste Frame der Animation verwendet wurde. Um die Animation zu deaktivieren war ein Anpassen des Plug-ins notwendig.

7.4 Modularität und Wiederverwendbarkeit

Um die Modularität und Wiederverwendbarkeit des Web Services zu zeigen, wurde eine neue Version des Heatmap-Plug-ins implementiert. Die in eTaddy vorhandene Heatmap-Visualisierung ist in DirectX geschrieben und erzeugt in den vorliegenden Version nur Heatmaps in Graustufen (siehe Abbildung 7.8 links). Daher wurde ein neues Plug-in mit dem Namen „Heatmap2“ angelegt. Das Plug-in implementiert das IVisPlugin-Interface und ist damit direkt in eTaddy nutzbar. Ein Vergleich der beiden Plug-ins ist in Abbildung 7.8 zu sehen.

Neue Funktionalität kann durch Plug-ins einfach hinzugefügt werden. In eTaddy sind diese direkt verfügbar. Für den Web Service müssen die als Parameter übergebenen IDs in LINQ-Objekte aufgelöst und eine Hilfsklasse angelegt werden, damit das Ergebnis des Plug-ins in XML serialisiert werden kann. Solange die Parameter und der Rückgabewert des Interfaces gleich bleiben, kann die Implementierung eines Plug-ins einfach ausgetauscht werden, ohne dass Änderungen an eTaddy oder am Web Service notwendig sind.

7.5 Fehlerbehandlung

Tritt bei der Berechnung innerhalb eines Plug-ins ein Fehler auf, darf der Web Service davon nicht beeinflusst werden. Bei einem Fehler wirft das Plug-in eine Exception, die vom Web Service eingefangen wird. Der Web Service generiert dann anstelle der eigentlichen SOAP-Antwort eine SOAP-Fault-Nachricht, die den Stacktrace der Exception enthält. Wird der Web

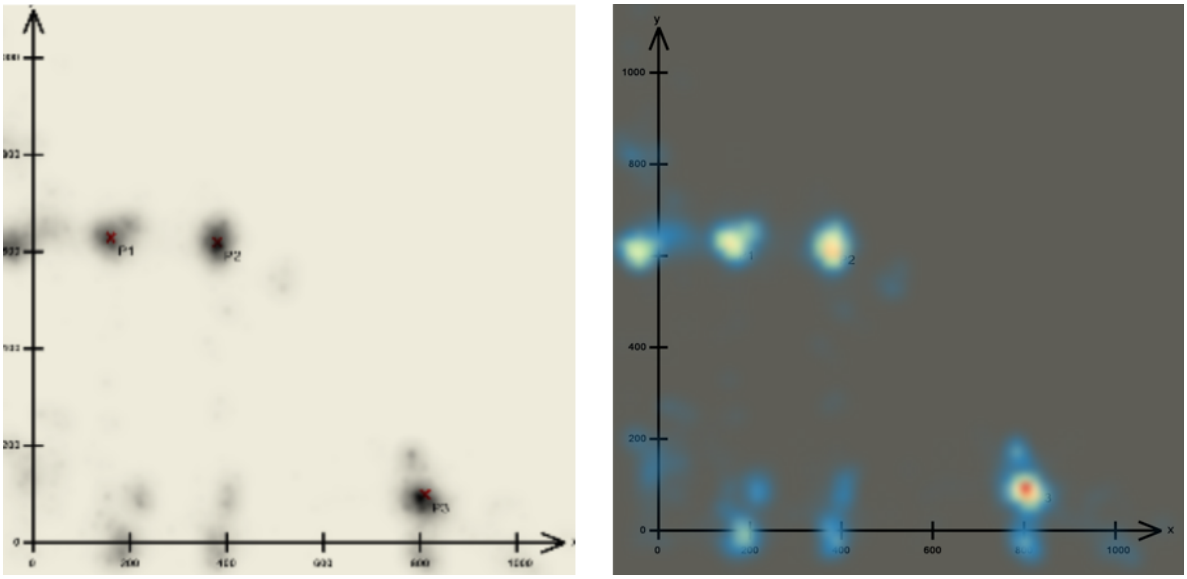

Abbildung 7.8: Links: Die ursprüngliche Implementierung der Heatmap-Visualisierung. Rechts: Die neue Implementierung des Plug-ins. Die Funktionalität des Plug-ins wird von eTaddy und dem Web Service genützt.

Service von einem Worfklow aufgerufen, kann dieser auf die SOAP-Fault-Nachricht reagieren und die fehlgeschlagene Aktivität kompensieren.

7.6 Deployment

Der Web Service wurde auf einer virtuellen Maschine (VM) eingerichtet, auf der Windows Server 2012 als Betriebssystem läuft. Dafür war es zunächst notwendig, einen Microsoft SQL-Server auf der VM zu installieren. In diesen wurde das VIS-Datenmodell und die Daten einer Eye-Tracking-Benutzerstudie importiert. Für den Web Service selbst wurde der Web Server Internet Information Services[10] (IIS) Express in Version 8.0 installiert. Außerdem war die Installation des .NET Frameworks 4.5, des WPF-Toolkits und der Statistiksoftware R notwendig, damit der Web Service ausgeführt werden kann.

Der ASP.NET Web Service generiert automatisch eine Übersicht von allen verfügbaren Operationen (siehe Abbildung 7.9). Ein Klick auf eine Operation zeigt die Struktur der eingehenden und ausgehenden SOAP-Nachrichten an. Wenn die WebMethod nur primitive Datentypen als Parameter erwartet, werden entsprechende Eingabefelder dargestellt und die Operation kann direkt getestet werden. Die vom Web Service generierte WSDL-Datei kann mit einem Klick auf „Dienstbeschreibung“ aufgerufen werden. Die WSDL-Datei ist beispielsweise über die URL `http://localhost:2427/StudyService.asmx?WSDL` erreichbar und enthält alle Operationen und XML-Schema-Datentypen. Die Operationen selbst werden über den Paramter „op“ und dem entsprechenden Namen der WebMethod aufgerufen. Die

[10] `http://www.iis.net/learn/extensions/introduction-to-iis-express/iis-express-overview`

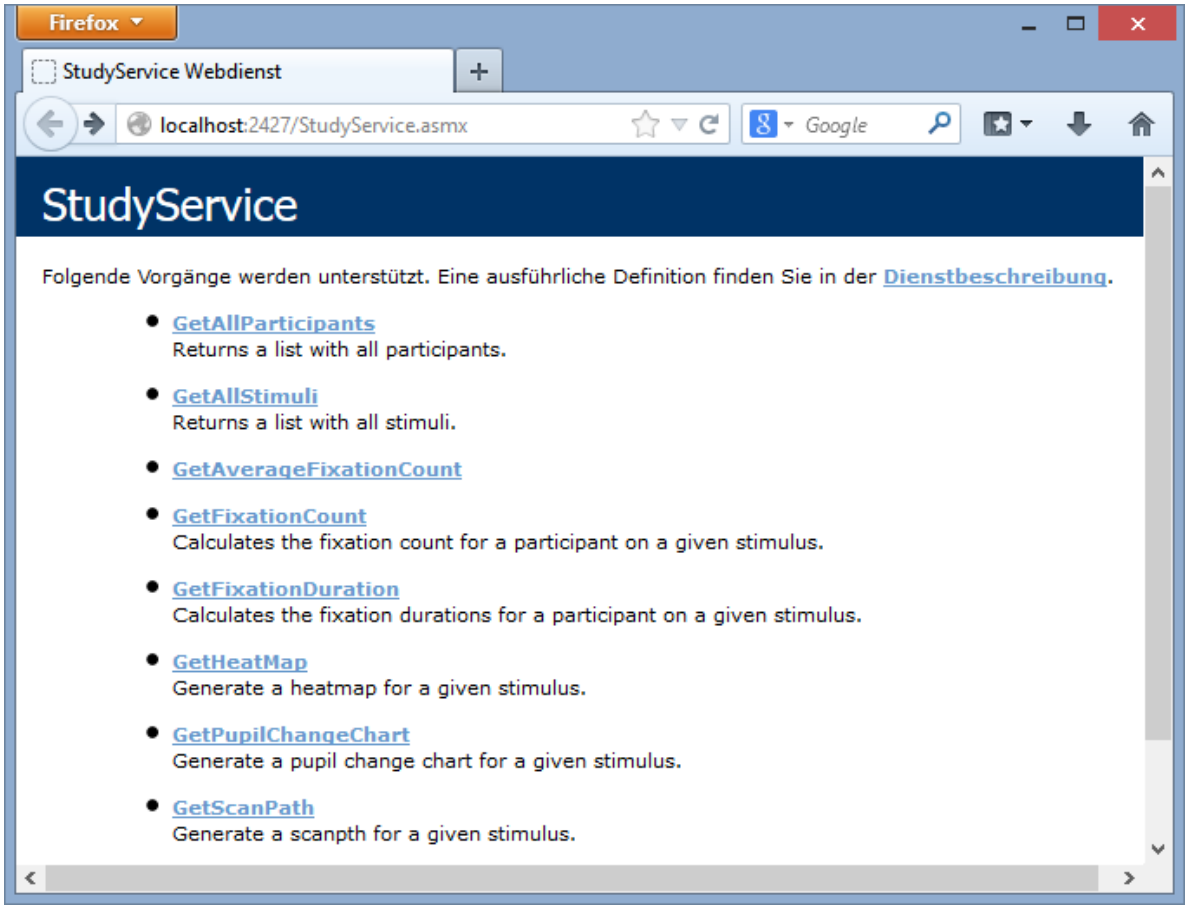

Abbildung 7.9: Die vom ASP.NET Web Service generierte Übersicht aller unterstützen Operationen. Hinter dem Link „Dienstbeschreibung“ verbirgt sich die WSDL-Datei des Web Services. Die Beschreibung der Operationen stammt von den Annotierungen der WebMethods.

Liste aller Stimuli kann beispielsweise über die URL `http://localhost:2427/StudyService.asmx?op=GetAllStimuli` abgerufen werden. In Abbildung 7.10 wird ein beispielhafter Aufruf einer Web Service-Operation gezeigt und erklärt.

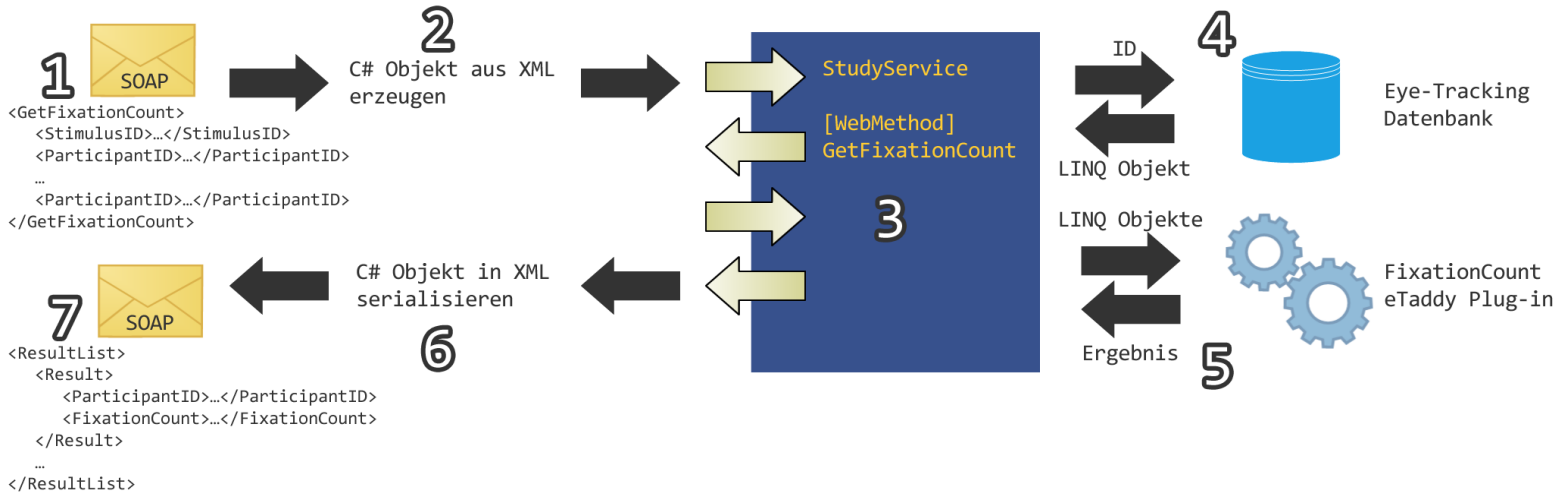

Abbildung 7.10: Veranschaulichung des prototypisch implementierten Web Servers. Eine Anfrage wird in Form einer SOAP-Nachricht (1) an den Web Server geschickt. Im Body der SOAP-Nachricht steht als Root-Element der Name der aufzurufenden Methode und als Kind-Elemente die Parameter. Aus den Parametern wird mit Hilfe einer mit XML-Annotationen versehenen C#-Klasse ein C#-Objekt erzeugt (2). Der Web Service (3) wird instanziiert und stellt eine Verbindung zur Datenbank her. Anschließend wird eine als WebMethod annotierte Methode des Web Servers, deren Namen dem Root-Element aus der SOAP-Nachricht entspricht, aufgerufen. Das erzeugte C#-Objekt wird als Parameter übergeben. Das C#-Objekt enthält die ID von einem Stimulus und eine Liste von IDs von Probanden. Für jede ID wird eine Anfrage an die Datenbank gestellt, die jeweils ein LINQ-Objekt zurück gibt, das die zugrundeliegenden Daten in der Datenbank repräsentiert (4). Danach wird das eTaddy Plug-in zur Berechnung instanziiert. Jedes Plug-in implementiert eine „Calculate“-Methode, die einen Stimulus und verschieden viele Probanden als Parameter erwartet. Plug-ins, die nur einen Probanden erwarten, werden mehrfach ausgeführt, wenn eine Liste von Probanden an den Web Service gesendet wurde (5). Das Ergebnis der Berechnung wird in eine Instanz von einer annotierten C#-Hilfsklasse geschrieben, die in eine XML-Struktur serialisiert wird (6). Das in XML serialisierte Ergebnis wird in den Body der SOAP-Response-Nachricht geschrieben (7).

8 Demonstration

In diesem Kapitel wird die Funktionalität des in Kapitel 7 prototypisch implementierten Web Service demonstriert. Dafür wurden exemplarisch zwei Eye-Tracking-Analyseszenarien ausgewählt, für die jeweils ein Workflowmodell erstellt wurde. In den beiden folgenden Abschnitten werden die zwei Beispielszenarien und Workflowmodelle vorgestellt. Am Ende des Kapitels wird die Funktionalität des implementierten Prototyps kurz diskutiert.

8.1 Beispielszenario 1

Das erste Beispielszenario ist die Analyse einer Eye-Tracking-Benutzerstudie, bei der die Augenbewegungen von zwanzig Probanden auf jeweils dreißig Stimuli aufgezeichnet wurden. Nun soll die durchschnittliche Fixationsdauer für jeden Proband auf jedem Stimulus berechnet werden.

In eTaddy müsste dafür jede Proband- und Stimulus-Kombination einzeln ausgewählt und anschließend mit einem Klick auf „Fixation Duration“ und „Average“ berechnet werden.

Abbildung 8.1 zeigt, wie dieser Prozess auf einen Workflow abgebildet werden kann. Zunächst wird die Operation `GetAllStimuli` aufgerufen. Die Operation gibt als SOAP-Response eine Liste mit allen Stimuli-IDs zurück. Für jede Stimuli-ID (`ForEachStimuli`) kann anschließend die Operation `GetAllParticipants` ausgeführt werden, um eine Liste der Probanden-IDs zu erhalten, die den jeweiligen Stimulus bearbeitet haben. Mit der Stimulus-ID und den Probanden-IDs kann nun die `GetAverageFixationDuration`-Operation für jeden Stimulus aufgerufen werden. Die Ergebnisse der Berechnungen können abschließend konkateniert werden.

8.2 Beispielszenario 2

Im zweiten Beispielszenario wird eine Eye-Tracking-Benutzerstudie mit nur einem Stimulus behandelt. Die Probanden sollen nach ihrem Alter in zwei Gruppen eingeteilt werden. Die erste Gruppe soll nur aus Probanden bestehen, die jünger als 22 sind. Die andere Gruppe aus den restlichen Probanden, die folglich 22 oder älter sind. Für jede der beiden Gruppen soll anschließend eine Heatmap-Visualisierung erstellt werden, um einen Vergleich bei der Betrachtung des Stimulus zu ermöglichen.

In eTaddy ist die Bearbeitung dieses Szenarios nicht möglich, da neben dem Probandenname keine weiteren Informationen angezeigt werden.

In Abbildung 8.2 ist zu sehen, wie der Prozess als Workflow modelliert werden kann. Zunächst wird die Stimulus-ID eingelesen (`AssignStimulusID`). Dann werden alle zugehörigen Probanden ausgelesen (GetAllParticipants). Die Liste der Probanden enthält neben der ID des Probanden auch das Alter und kann nun mit einem XPath-Ausdruck gefiltert werden.

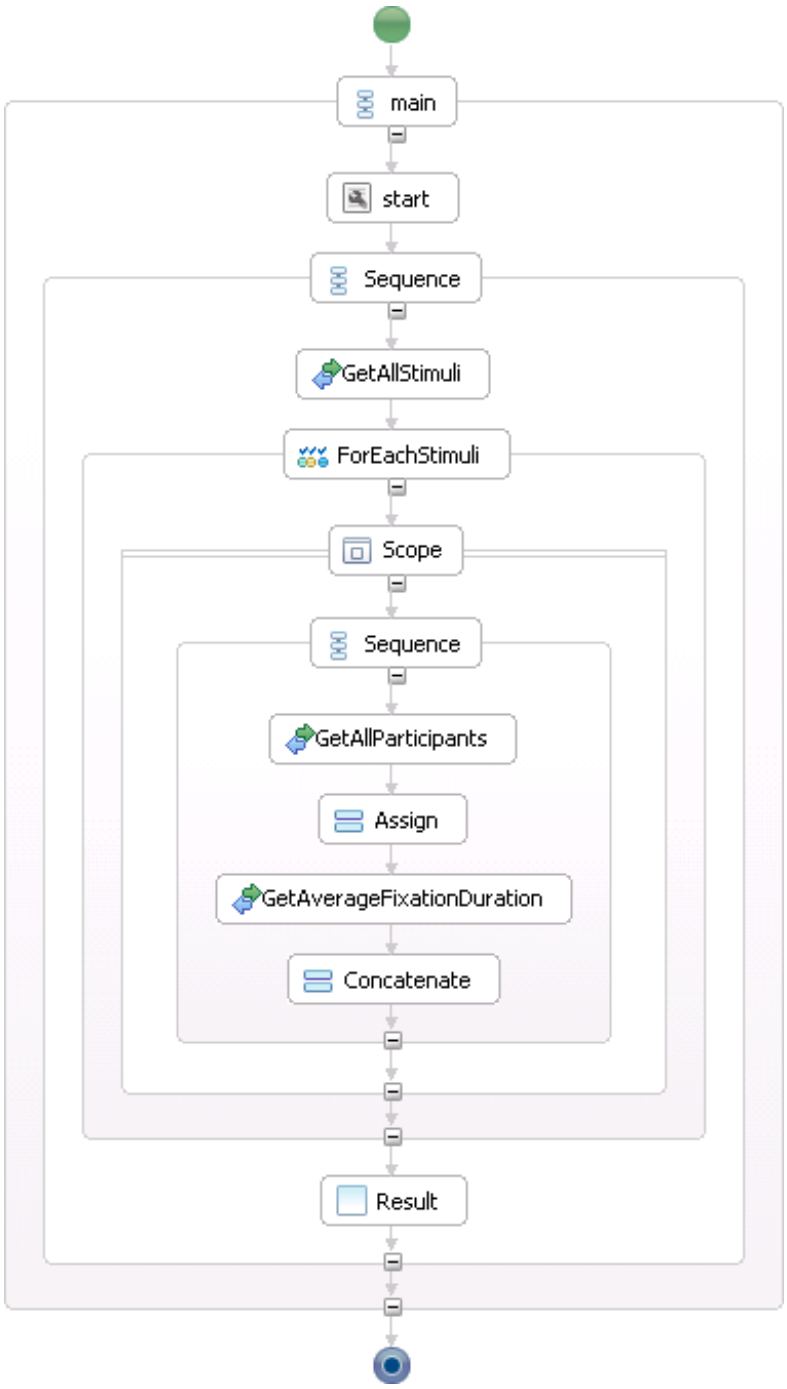

Abbildung 8.1: Das erste Beispielszenario ist die Berechnung der durchschnittlichen Fixationsdauer (GetAverageFixationDuration) für alle Probanden (GetAllParticipants) auf allen Stimuli (GetAllStimuli).

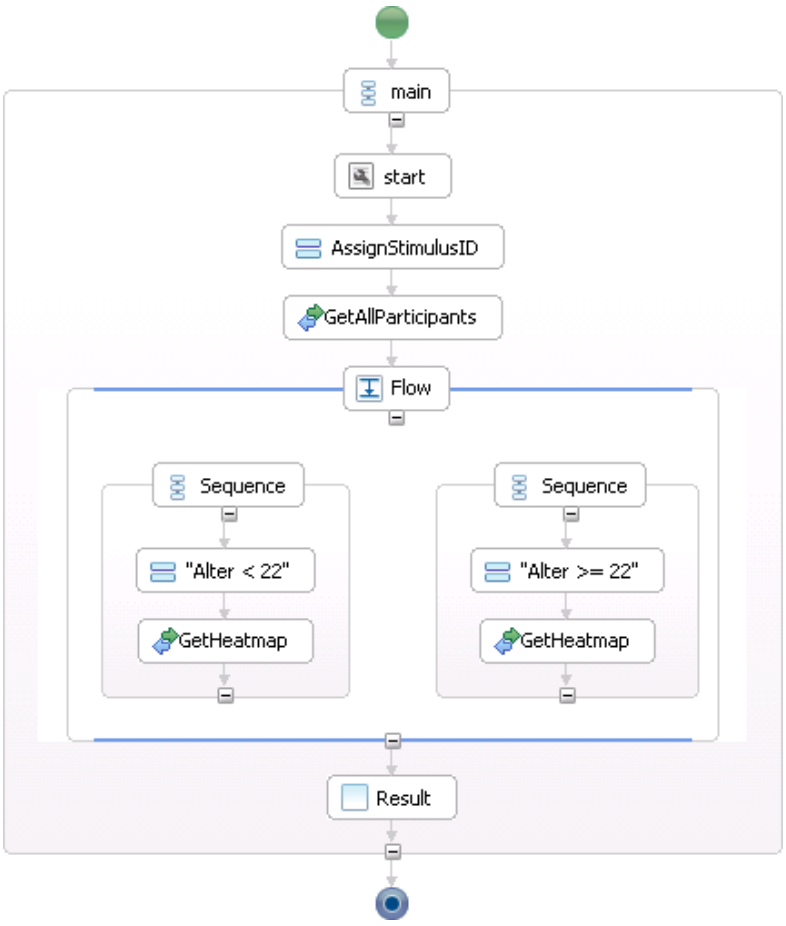

Abbildung 8.2: Das zweite Beispielszenario ist das Erstellen von zwei Heatmaps (GetHeatmap) für einen gegebenen Stimulus, wobei alle Probanden (GetAllParticipants) nach ihrem Alter in zwei Gruppen aufgeteilt werden.

Das Filtern kann parallel durchgeführt werden (Flow), wobei einmal die Probanden gefiltert werden, die jünger als 22 sind (links in der Abbildung) und einmal alle anderen (rechts in der Abbildung). Für beide Gruppen kann anschließend eine Heatmap erstellt werden, indem mit der Stimulus-ID und den Probanden-IDs die Operation `GetHeatmap` aufgerufen wird. Durch das Ersetzen des festgelegten Alters („22“) durch eine BPEL-Variable, die zusammen mit der Stimulus-ID beim Start festgelegt wird, kann das Workflowmodell bei zukünftigen Analysen wiederverwendet werden.

8.3 Diskussion

Die Abbildung der Analyseschritte auf Web Service-Operationen ermöglicht, häufig wiederkehrende Berechnungen zu speichern und bei Bedarf erneut auszuführen. Die Workflowmodelle können zwischen Wissenschaftlern ausgetauscht und iterativ verbessert werden. Allerdings muss dazu gesagt werden, dass eine wirkliche Automatisierung momentan nur mit

den Metriken und Statistiken möglich ist. Diese stellen Berechnungen an, deren Ergebnisse anschließend weiterverarbeitet werden können und die den Kontrollfluss des Workflows steuern. Im Gegensatz dazu können die Visualisierungen zwar auch automatisch erstellt werden, müssen aber von einem Mensch bewertet werden.

9 Zusammenfassung und Ausblick

Dieses Kapitel fasst die in der Studienarbeit erarbeiteten Konzepte, Erkenntnisse und Ergebnisse zusammen. Außerdem wird ein Ausblick gegeben, wie das Lösungskonzept weiterentwickelt werden kann und welche Änderungen dafür am Prototyp notwendig sind.

9.1 Zusammenfassung

In dieser Studienarbeit wurde ein Konzept entwickelt, das einen Benutzer bei der Analyse von einer Eye-Tracking-Benutzerstudie unterstützt. Die Grundidee des entwickelten Konzepts ist, den Analyseprozess einer Eye-Tracking-Benutzerstudie als Workflow zu modellieren. Dabei wird jeder Analyseschritt auf eine Web Service-Operation abgebildet, die von einer Workflow-Aktivität aufgerufen werden kann. Ein Workflowmodell ermöglicht es, die Analyseschritte zu speichern und für ähnliche Eye-Tracking-Benutzerstudien wiederzuverwenden. Die entwickelten Modelle können zwischen Wissenschaftlern ausgetauscht und in Kollaboration iterativ verbessert werden.

Auf der Basis dieses Konzepts wurde prototypisch ein Web Service implementiert, dessen Operationen die Funktionalität eines bestehenden Frameworks für die Analyse von Eye-Tracking Daten zur Verfügung stellt. Die Funktionalität umfasst dabei zum Beispiel die Abfrage von allgemeinen Informationen der durchgeführten Studien, wie die verwendeten Stimuli und die zugehörigen Probanden. Weiterhin ist es möglich, auf den Eye-Tracking-Daten verschiedene Metriken, Statistiken und Visualisierungen zu berechnen. Die Daten werden in einer bestehenden Datenbank gehalten und vom Web Service abgerufen. Um die Größe der Nachrichten zwischen den Workflow-Aktivitäten und den Web Service-Operationen gering zu halten, werden für den Aufruf der Operationen nur Referenzen auf die Eye-Tracking-Daten versendet.

Anhand von zwei Beispielszenarien wurde die Funktionalität des Prototyps demonstriert, indem für jedes Beispielszenario ein Workflowmodell erstellt wurde, welches die Web Service-Operationen nutzt.

9.2 Ausblick

Die in Kapitel 8 vorgestellten Workflows dienen nur zur Demonstration des entwickelten Prototyps. Damit das hier erarbeitete Konzept tatsächlich eingesetzt werden kann, ist es notwendig, umfassende Workflowmodelle für die Analyse von Eye-Tracking-Benutzerstudien zu erstellen. Dabei kommt zwangsläufig die Frage auf, wie sich der Analyseprozess von verschiedenen Eye-Tracking-Benutzerstudien unterscheidet.

Der prototypisch implementierte Web Service kann natürlich um zusätzliche Operationen erweitert werden. Diese können in Form von eTaddy-Plug-ins umgesetzt werden und

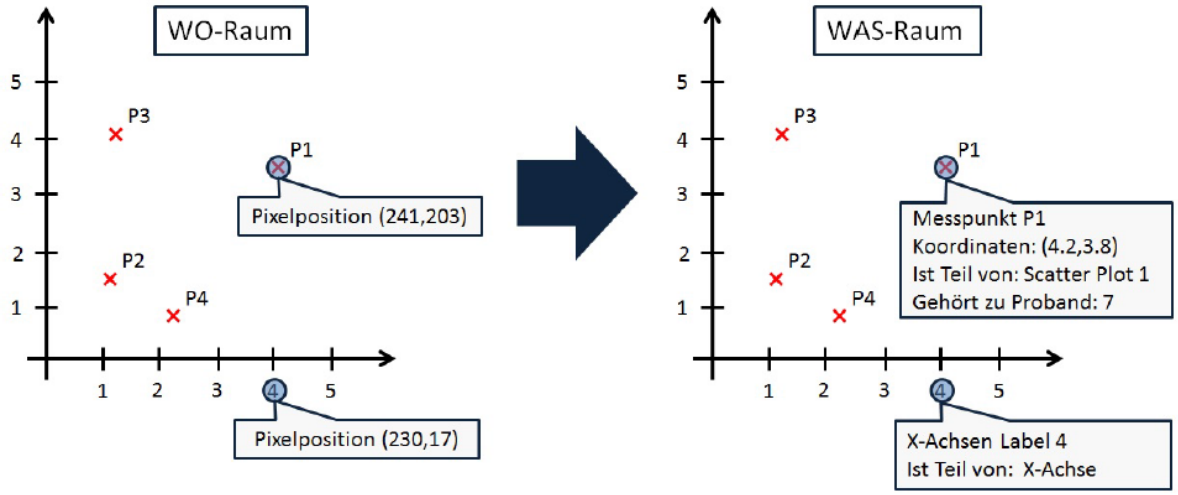

Abbildung 9.1: Abbildung vom WO-Raum (links) in den WAS-Raum (rechts) [32].

ergänzen damit gleichzeitig eTaddy um neue Funktionen. Interessant hierbei sind vor allem die Visualisierungen. Die aktuellen Implementierungen der Heatmap- und Scanpath-Visualisierungen und dem Pupillengrößendiagramm sind statisch und bieten außer der Wahl des Stimuli und der Probanden keine weiteren Optionen. Angelehnt an die Arbeit von Marcel Russ [27] (siehe Abschnitt 4.3) können die Visualisierungen auf mobile Endgeräte und große Displays angepasst werden. Weiterhin ist zu überlegen, wie dynamische Visualisierungen, beispielsweise die Bee-Swarm-Visualisierung, umgesetzt werden können. Umgekehrt wurden in diesem Konzept nur statische Stimuli betrachtet. Die Evaluation von dynamischen Stimuli hat, aufgrund der zusätzlichen Zeit-Dimension, weitere Anforderungen.

Außerdem ist zu überlegen, die Stimuli mit semantischen Informationen zu annotieren. In der Studienarbeit von Stefan Strohmaier mit dem Titel „Entwicklung eines Konzepts zur Annotation von grafischen Elementen in Visualisierungen“ [32] wurde das Visualisierungs-Framework D3[1] um eine semantische Komponente erweitert. Mit der Erweiterung werden die von D3 generierten SVG[2]-Dateien automatisch mit RDFa[3] auf Basis einer Visualisierungs-Ontologie annotiert. Da sowohl SVG als auch RDFa maschinell verarbeitet werden können, eignen sich diese Dateien eher für die automatische Verarbeitung als Bitmap-Grafiken. Eine Integration in den Web Service würde bedeuten, dass dieser nicht nur mit den Pixelpositionen der Fixationen rechnen könnte, sondern auch mit den betrachteten Objekten. Abbildung 9.1 zeigt die Abbildung vom WO-Raum in den WAS-Raum anhand eines Koordinatensystems.

Weiterhin ist es denkbar, das Konzept auf die gesamte Durchführung der Eye-Tracking-Benutzerstudien auszuweiten. Der Benutzer könnte dabei beim Erstellen der Fragebögen über die Eingabe der Benutzerdaten und den Import der aufgezeichneten Eye-Tracking-Daten unterstützt werden.

[1] http://d3js.org/
[2] http://www.w3.org/TR/2011/REC-SVG11-20110816/
[3] http://www.w3.org/TR/2013/NOTE-rdfa-primer-20130822/

Literaturverzeichnis

[1] BEECH, D., SPERBERG-MCQUEEN, M., GAO, S., THOMPSON, H., MALONEY, M., AND MENDELSOHN, N. W3C xml schema definition language (XSD) 1.1 part 1: Structures. W3C recommendation, W3C, Apr. 2012. http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/. (Zitiert auf Seite 16)

[2] BLASCHECK, T. Eyetracking basiertes Analysekonzept zur Evaluation von Visualisierungen. Master's thesis, Universität Stuttgart, 2012. (Zitiert auf den Seiten 27, 29, 38 und 39)

[3] BÖHME, M., MEYER, A., MARTINETZ, T., AND BARTH, E. Remote eye tracking: State of the art and directions for future development. In *Proc. of the 2006 Conference on Communication by Gaze Interaction (COGAIN)* (2006), pp. 12–17. (Zitiert auf Seite 6)

[4] CHRISTENSEN, E., CURBERA, F., MEREDITH, G., AND WEERAWARANA, S. W3C web services description language (wsdl) 1.1. W3C note, W3C, Mar. 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315. (Zitiert auf Seite 18)

[5] CURBERA, F., GOLAND, Y., KLEIN, J., LEYMANN, F., WEERAWARANA, S., ET AL. Business process execution language for web services, version 1.1. (Zitiert auf Seite 24)

[6] DODGE, R., AND CLINE, T. S. The angle velocity of eye movements. *Psychological Review 8*, 2 (1901), 145. (Zitiert auf Seite 3)

[7] DREWES, H. *Eye gaze tracking for human computer interaction*. PhD thesis, lmu, 2010. (Zitiert auf Seite 11)

[8] DUCHOWSKI, A. T. *Eye tracking methodology: Theory and practice*, vol. 373. Springer, 2007. (Zitiert auf den Seiten 6, 7 und 11)

[9] FITTS, P. M., JONES, R. E., AND MILTON, J. L. Eye movements of aircraft pilots during instrument-landing approaches. *Ergonomics. 3. Psychological mechanisms and models in ergonomics 3* (2005), 56. (Zitiert auf Seite 4)

[10] HART, S. G., AND STAVELAND, L. E. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload 1*, 3 (1988), 139–183. (Zitiert auf Seite 11)

[11] HARTRIDGE, H., AND THOMSON, L. Methods of investigating eye movements. *The British journal of ophthalmology 32*, 9 (1948), 581. (Zitiert auf Seite 4)

[12] HOLMQVIST, K., NYSTRÖM, M., ANDERSSON, R., DEWHURST, R., JARODZKA, H., AND VAN DE WEIJER, J. *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press, 2011. (Zitiert auf Seite 7)

[13] HOTTA, S. Ausführung von Festkörpersimulationen auf Basis von Workflow Technologie. Master's thesis, Universität Stuttgart, 2010. (Zitiert auf Seite 33)

[14] HUBER, O. *Das psychologische Experiment: eine Einführung*. Aus dem Programm Huber: Psychologie Lehrtexte. H. Huber, 2005. (Zitiert auf Seite 11)

[15] HUEY, E. B. The psychology and pedagogy of reading: With a review of the history of reading and writing and of methods, texts, and hygiene in reading, 1968. (Zitiert auf Seite 4)

[16] JACOB, R. J., AND KARN, K. S. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *Mind 2*, 3 (2003), 4. (Zitiert auf Seite 3)

[17] JAVAL, E. Essai sur la physiologie de la lecture. In *Annales D'Oculistique* (1879). (Zitiert auf den Seiten 1 und 3)

[18] JUST, M. A., AND CARPENTER, P. A. A theory of reading: From eye fixations to comprehension. *Psychological review 87* (1980), 329–354. (Zitiert auf Seite 3)

[19] KARASTOYANOVA, K. G. M. S. D., AND REITER, F. L. M. Conventional workflow technology for scientific simulation. (Zitiert auf den Seiten 22, 23, 24, 30 und 31)

[20] KARMARKAR, A., MOREAU, J.-J., HADLEY, M., MENDELSOHN, N., GUDGIN, M., NIELSEN, H. F., AND LAFON, Y. SOAP version 1.2 part 1: Messaging framework (second edition). W3C recommendation, W3C, Apr. 2007. http://www.w3.org/TR/2007/REC-soap12-part1-20070427/. (Zitiert auf Seite 16)

[21] LEYMANN, F., AND ROLLER, D. *Production workflow: concepts and techniques*. Prentice Hall PTR Upper Saddle River, 2000. (Zitiert auf Seite 21)

[22] LI, D., AND PARKHURST, D. J. Starburst: A robust algorithm for video-based eye tracking. *Elselvier Science* (2005). (Zitiert auf Seite 6)

[23] MONTY, R. A. An advanced eye-movement measuring and recording system. *American Psychologist 30*, 3 (1975), 331. (Zitiert auf Seite 4)

[24] POOLE, A., AND BALL, L. J. Eye tracking in human-computer interaction and usability research: Current status and future. In *Prospects, Chapter in C. Ghaoui (Ed.): Encyclopedia of Human-Computer Interaction. Pennsylvania: Idea Group, Inc* (2005). (Zitiert auf den Seiten 7 und 8)

[25] RASCHKE, M., CHEN, X., AND ERTL, T. Parallel scan-path visualization. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (New York, NY, USA, 2012), ETRA '12, ACM, pp. 165–168. (Zitiert auf Seite 10)

[26] ROSENHOLTZ, R., LI, Y., MANSFIELD, J., AND JIN, Z. Feature congestion: a measure of display clutter. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), ACM, pp. 761–770. (Zitiert auf Seite 10)

[27] RUSS, M. Framework für die Visualisierung von Datenqualität in Simulation-Workflows. Master's thesis, Universität Stuttgart, 2012. (Zitiert auf den Seiten 34, 35 und 66)

[28] RUTSCHMANN, J. Generisches Web Service Interface um Simulationsanwendungen in BPEL-Prozesse einzubinden. Master's thesis, Universität Stuttgart, 2009. (Zitiert auf Seite 33)

[29] SHEBILSKE, W. L., AND FISHER, D. F. Eye movements and context effects during reading of extended discourse. *Eye movements in reading: Perceptual and language processes* (1983), 153–179. (Zitiert auf Seite 7)

[30] SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on* (1996), IEEE, pp. 336–343. (Zitiert auf den Seiten 27 und 39)

[31] ŠPAKOV, O. icomponent-device-independent platform for analyzing eye movement data and developing eye-based applications. *Unpublished PhD Thesis, University of Tampere, Tampere, Finland* (2008). (Zitiert auf Seite 9)

[32] STROHMAIER, S. Entwicklung eines konzepts zur annotation von grafischen elementen in visualisierungen. (Zitiert auf Seite 66)

[33] TINKER, M. A. *Legibility of print*, vol. 1. Iowa State University Press Ames, 1963. (Zitiert auf Seite 4)

[34] WEERAWARANA, S., CURBERA, F., LEYMANN, F., STOREY, T., AND FERGUSON, D. F. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005. (Zitiert auf den Seiten 12, 13 und 24)

[35] YARBUS, A. L., HAIGH, B., AND RIGSS, L. A. *Eye movements and vision*. No. 5.10. Plenum press New York, 1967. (Zitiert auf den Seiten 4 und 5)

[36] YERGEAU, F., SPERBERG-MCQUEEN, M., MALER, E., BRAY, T., AND PAOLI, J. Extensible markup language (XML) 1.0 (fifth edition). W3C recommendation, W3C, Nov. 2008. http://www.w3.org/TR/2008/REC-xml-20081126/. (Zitiert auf Seite 15)

Alle URLs wurden zuletzt am 10. 02. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift