Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3609

Visuelle Analyse von kognitiven Prozessen

Philipp Schuster

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuer:	Dipl.-Phys. Michael Raschke
Beginn am:	16. Januar 2014
Beendet am:	18. Juli 2014
CR-Nummer:	H.1.2, H.5.2, I.5.3, J.4

Kurzfassung

In dieser Arbeit wird ein Konzept entwickelt, mit dem durch die visuelle Analyse von Abfolgen von Areas of Interest (AOIs), unter Ausnutzung der Eye–Mind–Hypothese, kognitive Prozesse sichtbar gemacht werden. Durch die Analyse der kognitiven Prozesse soll die Gewinnung einer Lesevorschrift für einen Visualisierungstyp ermöglicht werden. Eine Lesevorschrift dient als Eingabe für eine Augenbewegungs–Simulation. Das Konzept sieht vor, eine für eine bestimmte Aufgabe gültige Lesevorschrift für einen Visualisierungstyp aus Eye–Tracking–Daten mehrerer Probanden abzuleiten. Die Eye–Tracking–Daten werden auf verschiedenen Stimuli des Visualisierungstyps während der Lösung der Aufgabe erfasst. Das Konzept sieht ein mehrstufiges Vorgehen zur Ableitung der Lesevorschrift vor. In einem ersten Schritt werden die Stimuli mit AOIs annotiert. Im nächsten Schritt werden für jeden der Stimuli die Abfolgen der AOIs der Probanden analysiert. Daraufhin wird für jeden der Stimuli eine Abfolge von AOIs bestimmt, die die Lösungsstrategie repräsentiert, die die meisten Probanden zur Lösung der Aufgabe auf dem Stimulus wählten. Aus diesen Abfolgen wird im Anschluss diejenige ausgewählt, die die Lösungsstrategie repräsentiert, die trotz der Verschiedenheit der Stimuli eine Lösungsstrategie für beliebige Stimuli des Visualisierungstyps darstellt. Zur Bestimmung der repräsentativen Lösungsstrategien wird ein Visualisierungs– und Analysekonzept entwickelt, das mit einem automatischen, einem halbautomatischen und einem manuellen Verfahren arbeitet. Die Verfahren werden durch Algorithmen zur Ähnlichkeitsbestimmung von Lösungsstrategien und zur Mustersuche in Lösungsstrategien unterstützt. Die wichtigsten Teile des Konzepts wurden prototypisch implementiert und die Funktionsweise anhand zweier Szenarien demonstriert.

Abstract

This work presents a concept which uses the visual analysis of sequences of Areas of Interest (AOIs) to make cognitive processes visible by exploiting the eye–mind–hypothesis. Through analyzing cognitive processes it should be made possible to obtain a reading instruction for a visualization type. A reading instruction is used as input to an eye movement simulation. The concept enables the derivation of a reading instruction valid for a specific task for a visualization type from eye tracking data of multiple participants. The eye tracking data is recorded for different stimuli of the visualization type. The concept is a multi–phase approach. In a first step, the stimuli are annotated with AOIs. In the next step, the sequences of the AOIs of the participants have to be analyzed for each of the stimuli. This analysis results in a sequence of AOIs for each of the stimuli, which represents the solution strategy that most of the participants have chosen to solve the task. From these sequences the one is selected which represents the solution strategy. This solution can be seen as a solution strategy for arbitrary stimuli of the same visualization type. To determine the representative solution strategies a visualization and analysis concept has been developed which works with an automatic, a semi–automatic and a manual method. The methods are supported by algorithms for similarity computation of solution strategies and pattern matching in solution strategies. The most important parts of the concept have been prototypically implemented. The application of the prototype is demonstrated in two scenarios.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Um Visualisierungen auf ihre Lesbarkeit hin zu untersuchen, werden Benutzerstudien mit Hilfe von Eye-Tracking durchgeführt. Solche Studien sind sehr zeit- und kostenintensiv. Die Studien müssen vorbereitet, durch Pilotstudien weiter verfeinert und schließlich mit ausreichend vielen Probanden durchgeführt werden. Zudem belaufen sich die Kosten für einen Eyetracker auf mehrere tausend Euro. Um diesem Zeit- und Kostenaufwand entgegenzuwirken, möchte man den Benutzer simulieren. Hierzu ist es notwendig die kognitiven Prozesse, die beim Benutzer während des Betrachtens einer Visualisierung ablaufen, in der Simulation nachzubilden und die entsprechenden Augenbewegungen auszuführen. Der Zusammenhang zwischen kognitiven Prozessen und Augenbewegungen ist durch die Eye-Mind-Hypothese gegeben. Sie besagt, dass Sehen und Denken sich gegenseitig beeinflussen.

Diese gegenseitige Beeinflussung soll nun auch dazu genutzt werden, möglichst realistische Eingabedaten für die Simulation zu erstellen. Auf Grundlage der Eye-Mind-Hypothese wird davon ausgegangen, dass Eye-Tracking-Daten implizit Informationen über die kognitiven Prozesse enthalten, die während der Betrachtung der zugehörigen Visualisierung ablaufen. Daher ist ein neues Visualisierungskonzept für Eye-Tracking-Daten notwendig, das es ermöglicht, die zugehörigen kognitiven Prozesse zu analysieren und davon ausgehend Eingabedaten für die Simulation zu entwickeln.

Kern des neuen Visualisierungskonzepts ist die Darstellung der Eye-Tracking-Daten im WAS-Raum. Im Gegensatz zu herkömmlichen Eye-Tracking-Visualisierungen wird dabei nicht dargestellt *wo* ein Proband hinsieht, sondern *was* er ansieht. Durch diese Darstellung ist es möglich sowohl deklarative als auch prozedurale kognitive Prozesse besser analysieren zu können als bei herkömmlichen Eye-Tracking-Analyseverfahren.

Um Eye-Tracking-Daten im WAS-Raum darzustellen, ist zum einen die Identifikation von Areas of Interest (AOIs) in den Eye-Tracking-Daten und zum anderen eine geeignete Darstellungsmethode für die Eye-Tracking-Daten auf den identifizierten AOIs notwendig. Diese Darstellungsmethode umfasst die bildliche Repräsentation der AOIs, die Veranschaulichung der Beziehungen zwischen den AOIs und die Darstellung des zeitlichen Verlaufs der Eye-Tracking-Daten über die AOIs. Dadurch wird sichtbar, was wie intensiv in welcher Reihenfolge von welchem Probanden angesehen wurde.

Das Ziel dieser Arbeit ist die Entwicklung eines neuen Visualisierungskonzepts, das Eye-Tracking-Daten im WAS-Raum darstellt. AOIs sollen bildlich, mit ihren Beziehungen untereinander und im zeitlichen Ablauf dargestellt werden. Das Konzept wird prototypisch implementiert und seine Funktionsweise in zwei Szenarien demonstriert.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: Im Kapitel über die Grundlagen werden die Begriffe und Techniken erklärt, die in der Arbeit verwendet werden.

Kapitel 3 – Existierende Arbeiten: Dieses Kapitel behandelt die mit der vorliegenden Ausarbeitung verwandten Arbeiten.

Kapitel 4 – Aufgabe und Lösungsansatz: Hier wird die Aufgabe dieser Arbeit näher beschrieben und der Ansatz zur Lösung der Aufgabe erläutert.

Kapitel 5 – Lösungskonzept: Das Konzeptkapitel beschreibt ausführlich die Zusammensetzung und Funktionsweise des Konzepts zur visuellen Analyse von kognitiven Prozessen.

Kapitel 6 – Implementierung: Die Implementierung des Prototypen wird in diesem Kapitel vorgestellt.

Kapitel 7 – Demonstration: In diesem Kapitel wird die Funktionsweise des Prototypen anhand zweier Szenarien demonstriert.

Kapitel 8 – Zusammenfassung und Ausblick: Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen und beschreibt wie das Konzept erweitert werden könnte.

2 Grundlagen

Dieses Grundlagenkapitel gibt eine Einführung in die Themen dieser Ausarbeitung. Es werden Begriffe, Techniken und Konzepte erläutert, die im weiteren Verlauf der Arbeit immer wieder verwendet werden.

Abschnitt 2.1 stellt das Thema Eye–Tracking vor, definiert Grundbegriffe und erklärt Visualisierungstechniken für Eye–Tracking–Daten. In Abschnitt 2.2 wird die Kognition behandelt. Es wird auf den Begriff Kognition eingegangen, die Eye–Mind–Hypothese vorgestellt und das Thema Wissen erörtert. Schließlich wird in Abschnitt 2.3 die Modellierung und Darstellung von Wissen besprochen.

2.1 Eye Tracking

Eye-Tracking ist die Blickaufzeichnung eines Probanden auf einem gegebenen Stimulus. Laut Holmqvist et al. wurden bereits im späten 19. Jahrhundert Blicke aufgezeichnet [HNA+11]. Yarbus gelang 1967 die Blickaufzeichung mit hoher Genauigkeit [Yar67]. Dieser Abschnitt beschreibt die Grundlagen des Eye-Tracking. Es werden verschiedene Techniken zur Erfassung von Augenbewegungen, wichtige Begriffe und einige Visualisierungstechniken vorgestellt.

2.1.1 Techniken

Es exisitieren vier Techniken, um Augenbewegungen zu erfassen. Diese werden nachfolgend näher beschrieben.

Die *Elektrookulografie* wurde vor über 40 Jahren entwickelt und ist nach wie vor in Verwendung. Dabei wird über rund um das Auge angeordnete Elektroden das elektrische Potential der Haut gemessen. Mit dieser Technik werden die Blicke des Probanden relativ zur Kopfposition erfasst. Dies erfordert eine zusätzliche Erfassung der Kopfposition [Duc07].

Bei *skleralen Kontaktlinsen* wird ein optisches oder mechanisches Referenzobjekt auf Kontaktlinsen befestigt, welche direkt auf dem Auge getragen werden. Ein mögliches Referenzobjekt ist eine Drahtspule deren Position über ein elektromagnetisches Feld erfasst wird. Diese Methode ist sehr genau, aber unangenehm für den Probanden. Weiterhin kann auch bei dieser Technik nur die Blickposition relativ zum Kopf gemessen werden [Duc07].

Mit *Videookulographie* werden Blickaufnahmetechniken bezeichnet, die bestimmte Merkmale des Auges erfassen. Ein Merkmal ist die Position des Limbus. Mit Limbus wird nach Pschyrembel die Übergangszone von der Hornhaut zur Lederhaut des Auges bezeichnet [PDZ86]. Ein anderes Merkmal sind Reflexionen der Hornhaut durch eine naheliegende Lichtquelle. Dabei handelt es sich meist um

Infrarotlicht. Auch diese Techniken haben gemeinsam, dass sie nur die Blickposition relativ zum Kopf erfassen können [Duc07].

Die *Videobasierte Erfassung von Pupille und Hornhautreflexion* ermöglicht die Blickerfassung ohne eine Bestimmung der Kopfposition zu erfordern. Dies wird durch die Messung zweier Referenzpunkte des Auges realisiert. Ein Referenzpunkt ist die Pupillenmitte, der andere die Position der Hornhautreflexion. Der Abstand zwischen Pupillenmitte und Hornhautreflexion ändert sich nur durch die Augrotation und wird kaum durch Kopfbewegungen beeinflusst. Diese Eye-Tracking-Technik ist derzeit die modernste. Sie erhält Unterstützung durch Computer Vision-Techniken und Digitale Signal Prozessoren [Duc07].

2.1.2 Terminologie

Im Folgenden werden wichtige Begriffe des Eye-Trackings erklärt.

Stimulus Nach Blaschek et al. ist ein Stimulus jeglicher visuelle Inhalt, der Probanden während eines Eye-Tracking Experiments gezeigt wird [BKR+]. Es gibt statische und dynamische Stimuli. Ihr Inhalt kann aktiv oder passiv sein und sie können in 2D oder 3D vorliegen. Statische Stimuli zeigen unbewegte Inhalte, wie etwa Bilder. Dynamische Stimuli sind beispielsweise Videos. Bei aktiven Stimuli kann der Proband den Stimulus verändern. Dadurch wird der Stimulus dynamisch. Im Gegensatz dazu steht der passive Stimulus, bei dem keine Möglichkeit zur Veränderung besteht.

Gazepoint Bei Gazepoints handelt es sich um die mit der Aufnahmefrequenz eines Eyetrackers aufgenommen Position der Augen der Probanden [Tob13].

Fixation Dies definiert nach Holmqvist et al. einen Bereich auf dem Stimulus, in dem die Blicke des Betrachters für den Zeitraum von etwa 200 bis 300 Millisekunden verweilen [HNA+11]. Die Software eines Eyetrackers fasst durch die Anwendung eines Fixationsfilters mehrere Gazepoints zu einer Fixation zusammen [Tob13].

Sakkade Dabei handelt es sich nach Holmqvist et al. um die schnelle Bewegung des Auges von einer Fixation zu einer anderen Fixation [HNA+11]. Während einer Sakkade ist die Wahrnehmung unterdrückt.

Area of Interest Eine Area of Interest (AOI) definiert nach Holmqvist et al. eine Region auf dem Stimulus [HNA+11].

AOI Besuch Bezeichnet nach Holmqvist et al. einen Besuch in einer AOI vom Eintritt in die AOI bis zum Austritt [HNA+11]. Dies wird in der Literatur häufig auch als „gaze“ bezeichnet. In dieser Arbeit wird „AOI Besuch“ verwendet.

AOI Transition Damit wird nach Holmqvist et al. die Augenbewegung von einer AOI zu einer anderen AOI bezeichnet [HNA+11].

Smooth Pursuit Bezeichnet nach Holmqvist et al. die Verfolgung einer Bewegung auf dem Stimulus [HNA+11]. Dies passiert typischerweise nur bei dynamischen Stimuli. Im Gegensatz zu Sakkaden ist hierbei die Wahrnehmung nicht unterdrückt.

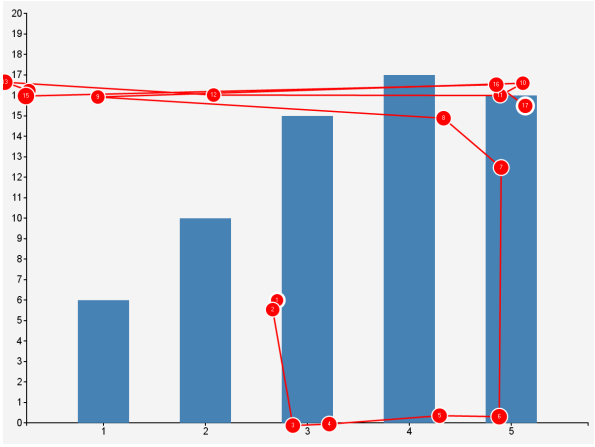

Abbildung 2.1: Scanpfad Visualisierung. Der Scanpfad zeigt die Fixationen und Sakkaden eines Probanden beim Betrachten des Stimulus. Der Scanpfad wurde mit einem Stimulus der Eye–Tracking–Studie „Visual Elements III“ an einem Eyetracker des Typs Tobii T60 XL mit der Software Tobii Studio erzeugt.

Scanpfad Damit wird nach Goldberg und Kotval eine alternierende Abfolge von Sakkaden und Fixationen bezeichnet [GK99]. Von einem Scanpfad kann man Informationen über das Suchverhalten eines Probanden ableiten.

Cross–Checking Bezeichnet nach Raschke et al. das Verhalten von Probanden, nach dem Auffinden einer Lösung für eine Stimulus bezogene Aufgabe, diese Lösung mehrmals auf ihre Richtigkeit hin zu überprüfen [RBR+14].

2.1.3 Visualisierungstechniken

Für die Visualisierung von Eye–Tracking–Daten gibt es verschiedene Möglichkeiten. Zum einen gibt es die Visualisierung im WO–Raum. Dies bedeutet, dass die Eye–Tracking–Daten bezogen auf den Stimulus, auf dem sie aufgezeichnet wurden, visualisiert werden. Zur Visualisierung von Eye–Tracking–Daten im WO–Raum gibt es beispielsweise die Scanpfad Visualisierung oder die Heatmap Visualisierung. Abbildung 2.1 zeigt ein Beispiel für eine Scanpfad Visualisierung auf einem Balkendiagramm. Dabei werden abwechselnd Fixationen und Sakkaden auf dem Stimulus eingezeichnet. In Abbildung 2.2 ist eine Heatmap dargestellt. Die Heatmap in Abbildung 2.2 basiert auf denselben Eye–Tracking–Daten wie die Scanpfad Visualisierung in Abbildung 2.1. Eine Heatmap zeigt die Verteilung von Fixationen auf einem Stimulus mittels einer Farbskala an. Die Farbskala reicht dabei meist von grün nach rot.

Alternativ zur Darstellung im WO–Raum kann die Visualisierung der Eye–Tracking–Daten im WAS–Raum erfolgen. Dazu ist die Annotation des Stimulus mit AOIs erfoderlich. Eine Visualisierung im WAS–Raum zeigt dann die Abfolge der AOI–Besuche eines Probanden. Dabei muss die Visualisierung nicht auf dem Stimulus erfolgen. Ein Beispiel für eine Visualisierung von Eye–Tracking–Daten im WAS–Raum ist die Parallel Scan–Path Visualisierung von Raschke et al. [RCE12]. Dazu wird in einem

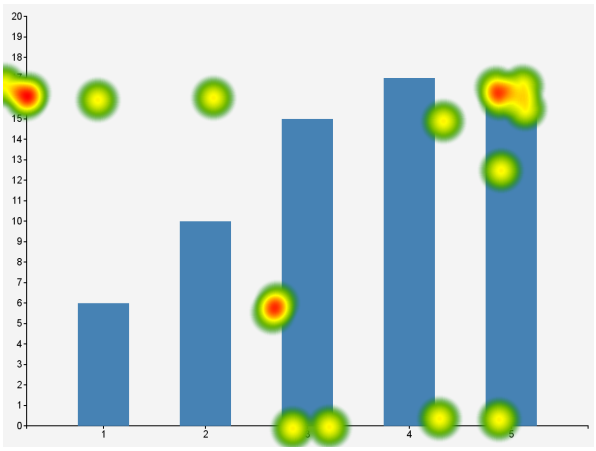

Abbildung 2.2: Heatmap Visualisierung. Die Heatmap zeigt die Verteilung der Fixationen eines Probanden beim Betrachten des Stimulus an. Die Heatmap wurde mit einem Stimulus der Eye–Tracking–Studie „Visual Elements III“ an einem Eyetracker des Typs Tobii T60 XL mit der Software Tobii Studio erzeugt.

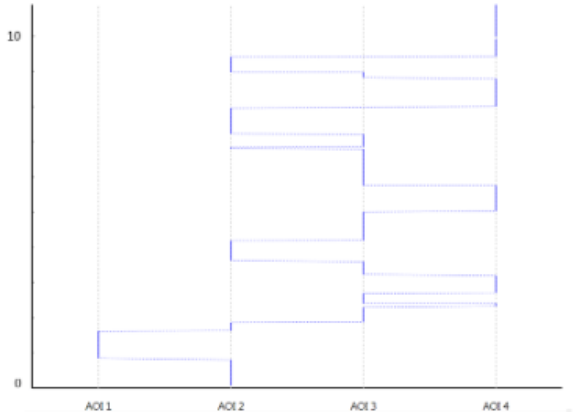

Abbildung 2.3: Parallel Scan–Path Visualisierung. Die Visualisierung zeigt die Abfolge von AOIs über die Zeit. Ein senkrechter Verlauf zeigt einen AOI–Besuch und ein waagerechter Verlauf eine Transition zwischen zwei AOIs [RCE12].

zweidimensionalen Koordinatensystem auf der senkrechten Achse die Zeit abgetragen und auf der waagrechten Achse die AOIs. Ein Beispiel für die Parallel Scan–Path Visualisierung ist in Abbildung 2.3 dargestellt.

2.2 Kognition

Im Folgenden werden die Grundlagen der Kognition vorgestellt. Zuerst wird der Begriff der Kognition definiert. Daraufhin wird die Eye–Mind–Hypothese vorgestellt und die Geschichte der Kognition wird kurz erläutert. Danach werden zwei verschiedene Ansätze der Kognitionswissenschaft dargelegt schließlich erklärt was Wissen ist und welche verschiedenen Formen von Wissen es gibt. Abschließend werden kognitive Architekturen vorgestellt.

2.2.1 Definition

> *„cognoscere“* (lat.), *„gignoskein“* (griech.): *erkennen, wahrnehmen, wissen*
>
> (in „Wörterbuch der Kognitionswissenschaft“ [Str96])

Das Wort „Kognition“ leitet sich aus dem Lateinischen bzw. Griechischen ab. Die Definition der Kognition nach Neisser bezeichnet Prozesse, durch die der sensorische Input umgesetzt, reduziert, weiter verarbeitet, gespeichert, wieder hervorgeholt und schließlich benutzt wird [NSA74]. Dies entspricht der anschaulicheren Definition nach Strube, der Kognition mit den intervenierenden Prozessen zwischen Wahrnehmung und Motorik bezeichnet [Str96]. Dies sind Prozesse des Gedächtnisses, Lernens, Denkens, Problemlösens und der Sprache. Kognition ist der Gegenstand der Kognitionswissenschaft, der kognitiven Psychologie und der Teilbereiche einer Reihe anderer Disziplinen wie etwa der Neurowissenschaft, der Linguistik oder der Informatik. Dabei hat jede Wissenschaft ihren eigenen Zugang und ihre eigenen Untersuchungsinteressen am Forschungsgegenstand Kognition.

Nach Strube ist die Grundhypothese der Kognitionswissenschaft, die Annahme, dass kognitive Prozesse als Berechnungsprozesse angesehen werden können [Str96]. Dadurch kann man sich von der Bindung der Kognition an natürliche Organismen lösen und sie auf künstliche Systeme übertragen und untersuchen. Es ergibt sich damit das Forschungsgebiet der künstlichen Intelligenz und der kognitiven Modellierung. Ziel der kognitiven Modellierung ist die Simulation natürlicher, meist menschlicher, kognitiver Prozesse auf Computern. Wichtig hierbei ist, dass neben einer mit dem natürlichen Vorbild vergleichbaren Leistung auch die gleichen Berechnungen simuliert werden. Dies steht bespielsweise im Gegensatz zu einem Schachcomputer, bei dem nur das Berechnungsergebnis nicht aber die zugehörigen Berechnungsvorgänge von Interesse sind.

2.2.2 Eye-Mind-Hypothese

Die Eye-Mind-Hypothese besagt, dass Sehen und Denken sich gegenseitig beeinflussen [JC80]. Just und Carpenter sagen dass das Auge so lange auf einem Wort beziehungsweise Stimulus fixiert bleibt, wie das Wort beziehungsweise der Stimulus verarbeitet wird. Sie führen zur Unterstreichung der Eye-Mind-Hypothese weiter aus, dass das Auge auf Wörtern pausiert, die mehr Verarbeitung benötigen.

Die Hypothese ist für diese Arbeit von fundamentaler Bedeutung. Die visuelle Analyse kognitiver Prozesse an Hand der Visualisierung für Eye-Tracking-Daten im WAS-Raum basiert auf der Annahme, dass die Probanden auch tatsächlich an das denken, was sie ansehen.

Nach der Arbeit von Just und Carpenter gibt es eine starke Korrelation zwischen Blicken und kognitiven Prozessen. Jedoch beschreiben Anderson et al., dass Augenbewegungen nicht zwingend mit kognitiven Prozessen zusammenhängen, sie aber mit kognitiven Prozessen zusammenhängen, die mit der Enkodierung von Informationen zusammenhängen [ABD04]. Enkodierung bezeichnet nach Funke und Frensch die Überführung von ankommenden physikalischen Reizen in neuronalen Code, den das Gehirn verarbeiten kann. Anderson führt aus, dass einige kognitive Aufgaben auf die Informationskodierung warten müssen, für viele andere jedoch ein „Denkpfad“ im Gehirn existiert, den Augenbewegungen nicht widerspiegeln [FF06].

Holmqvist et al. führen aus, dass Scanpfade zu einem großen Teil durch kognitive Faktoren beeinflusst werden, dass es jedoch unklar ist, durch welche genau [HNA+11]. Weiterhin ist es schwierig einen pauschalen Zusammenhang zwischen einem kognitiven Prozess und einem prototypischen Scanpfad Muster herzustellen. Jedoch zeigen sie Wege, um einen kognitiven Prozess mit einem Scanpfad für eine bestimmte Situation zu assoziieren. Ein Weg ist ein Versuchsaufbau, der die Anzahl der möglichen Interpretationen eines Scanpfad beschränkt. Dies setzt Kontrolle über Stimuli, Aufgabe und Hintergrund der Probanden voraus.

2.2.3 Geschichte

Nach Anderson wurde zu Beginn des 20. Jahrhunderts in der Psychologie das Verfahren der Introspektion angewandt [And01]. Dabei handelt es sich um ein Verfahren zur Selbstbeobachtung. Probanden wird ein Wort als Reiz vorgegeben und sie müssen all ihre Bewusstseinserfahrungen schildern, die nach dem Auftritt des Reizes vorkommen. Viele dieser Schilderungen handelten von eher unbeschreibbaren Bewusstseinserfahrungen. Als Reaktion auf dieses Problem entstand der Behaviorismus. Dieser versuchte psychologische Theorien rein über das äußerlich beobachtbare Verhalten eines Menschen zu begründen und ließ innere mentale Aspekte völlig außer acht. Durch den Behaviorismus nicht beantwortbare Fragen der Linguistik und aus der Zeit des Zweiten Weltkriegs sowie Fortschritte in der Computerwissenschaft führten schließlich zur kognitiven Wende, wie Miller es bezeichnet [Mil03]. Gemeint ist damit der Übergang vom Behaviorismus zum Kognitivismus. Im Kognitivismus werden innere mentale Vorgänge beachtet. Nach Strube lehnen viele Anhänger des Kognitivismus jedoch die kognitionswissenschaftliche Grundthese ab, nach der Kognition als Berechnung aufzufassen ist [Str96].

2.2.4 Ansätze

In der Kognitionswissenschaft gibt es nach Anderson zwei verschiedene Ansätze wie versucht wird Kognition zu erklären [And01]. Anhänger des Konnektivismus verfolgen eine Art Bottom-Up-Ansatz, der mit verknüpften Grundelementen, den Neuronen arbeitet. Konnektivismus deshalb, weil es um die Konnektion, die Verknüpfung neuronaler Elemente, geht. Dadurch wird versucht höhere kognitive Prozesse darzustellen und zu untersuchen. Im Gegensatz dazu stehen die Anhänger der

Symbolmanipulation. Dabei wird Kognition an Hand der Verarbeitung abstrakter Symbole erklärt. Es handelt sich dabei um eine Art Top-Down-Ansatz bei der sich tiefer auf Neuronenebene vorgearbeitet wird.

2.2.5 Wissen

In der Definition von Kognition ist der Prozess des Gedächtnisses enthalten. Nach Strube bezieht sich der Begriff Gedächtnis auf den Erwerb, die Speicherung und Nutzung von Wissen [Str96].

Definition

In der Philosophie wird nach Strube der Begriff Wissen folgendermaßen definiert:

> Wenn *P weiß, dass X*, ist zu fordern
> (1) *X ist wahr*
> (2) *P glaubt, dass X wahr ist*
> (3) *P kann begründen, warum X wahr ist*

Dies bedeutet, dass nach der philosophischen Definition von Wissen dieses immer wahr ist.

Strube führt weiterhin aus, dass in der Psychologie die erste Forderung aufgegeben wird [Str96]. Dort ist Wissen definiert als relativ dauerhafter Inhalt des Langzeitgedächtnisses. Wissen wird also als subjektiver und begründeter Glauben verstanden. In der Kognitionswissenschaft wird Wissen als die Menge von mentalen systeminternen Repräsentationen angesehen, die zusammen mit geeigneten Inferenztechniken ein kognitives System zur Bewältigung einer Aufgabe befähigen.

Gedächtnisarten

Nach Strube können drei Arten von Gedächtnis unterschieden werden. Die sensorischen Puffer, das Kurzzeitgedächtnis und das Langzeitgedächtnis [Str96]. Die sensorischen Puffer übernehmen die kurzfristige Speicherung von Informationen aus wahrgenommenen Stimuli zur potentiellen Weiterverarbeitung. Das Kurzzeitgedächtnis speichert temporäre Informationen und beteiligt sich an der Verarbeitung von Informationen. Es wird daher auch Arbeitsgedächtnis genannt. Im Langzeitgedächtnis können verschiedene Formen von Wissen abgelegt werden. Da in dieser Arbeit kognitive Prozesse visuell analysiert werden, wird im Folgenden näher auf die verschiedenen Wissensformen eingegangen.

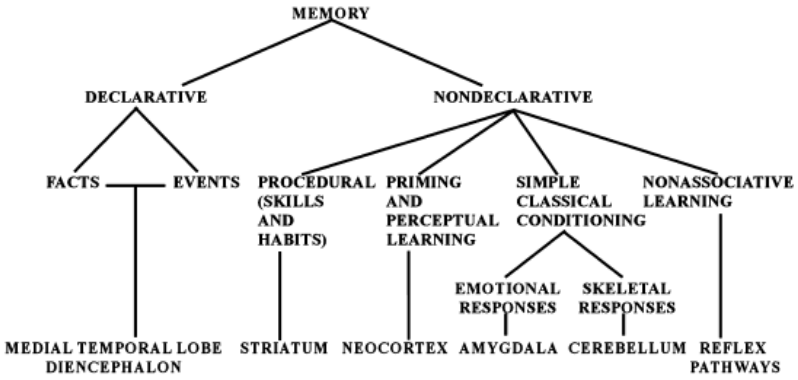

Abbildung 2.4: Übersicht über Wissensformen. Bei den Wissensformen wird grundsätzlich deklaratives und nicht–deklaratives Wissen unterschieden. Diese beiden Wissensformen werden dann erneut in noch feiner unterschiedene Wissensformen unterteilt [Squ04].

Wissensformen

Squire gibt eine Übersicht über die verschiedenen Wissensformen [Squ04]. Grundsätzlich unterscheidet er zwischen deklarativem und nicht-deklarativem Wissen. In der Literatur finden sich häufig die Bezeichnungen explizites und implizites Wissen. In einer früheren Arbeit identifizieren Squire und Zola-Morgan das deklarative Wissen mit dem expliziten Wissen und das nicht-deklarative Wissen mit dem impliziten Wissen [SZM91]. Die Wissensformen sind in Abbildung 2.4 dargestellt. In der Abbildung sind der Vollständigkeit wegen zusätzliche Wissensformen sowie die Zuordnung der Wissensformen zu den jeweiligen Hirnregionen dargestellt. Für diese Arbeit sind deklaratives Wissen und prozedurales Wissen von Bedeutung. Auf die weiteren Wissensformen wird in dieser Arbeit nicht näher eingegangen.

Nach Squire erkannte man bereits im 19. Jahrhundert, dass das Gedächtnis verschiedene Wissensformen unterscheidet [Squ04]. Die folgende Aufzählung gibt einen Überblick über die Entwicklung der Unterscheidung .

1804 Maine de Biran schreibt über mechanisches und repräsentatives Gedächtnis sowie über das Gefühlsgedächtnis [Bir04].

1890 James schreibt in seinem Buch „Principles of Psychology“ über Gedächtnis und Gewohnheiten [Jam90] .

1923 McDougall unterscheidet explizites und implizites Wissen [McD23].

1949 Im Kontext der Philosophie schreibt Ryle über „knowing that“ und „knowing how“ [Ryl49].

1962 Milner zeigt am amnestischen Patienten Henry Molaison, dass Fertigkeiten gelernt werden können, ohne den Erwerb selbst in Erinnerung zu behalten [Mil62]. Dies wird als Indiz für die Existenz verschiedener Gedächtnisformen und zugehöriger Speicherorte gewertet.

1972 Tulving unterscheidet zwischen episodischem und semantischem Gedächtnis [Tul72]. In den 1970er Jahren wird vor allem im Bereich der Forschung an künstlicher Intelligenz die Unterscheidung in deklaratives und prozedurales Wissen diskutiert. Bei Winograd werden Ansätze der expliziten Benennung der beiden Wissensformen erkennbar [Win72].

1975 In Winograds Arbeit zur künstlichen Intelligenz bekommen die beiden Wissensformen einen Namen [Win75].

1980 Auch im Bereich der Psychologie wird die Benennung bei Cohen und Squire sichtbar [CS80].

Deklaratives Wissen

Anderson beschreibt deklaratives Wissen als explizites Wissen, über das Auskunft gegeben werden kann und dessen man sich bewusst ist [And01]. Beispielsweise kann eine Person wissen, dass Berlin die Hauptstadt von Deutschland ist. Nach Strube handelt es sich bei deklarativem Wissen um Faktenwissen [Str96].

Nach Squire wird deklaratives Wissen weiter in semantisches und episodisches Wissen aufgeteilt [Squ04]. Semantisches Wissen enthält Fakten über die Welt. Episodisches Wissen enthält die Möglichkeit, ein Ereignis in dem Kontext widerzuerfahren, in dem es ursprünglich stattfand. Strube erklärt semantisches Wissen, mit Wissen über die Bedeutung von Begriffen, ihren Merkmalen und Beziehungen zueinander [Str96]. Episodisches Wissen bezeichnet Wissen, das mitsamt Ort und Zeit des erinnerten Ereignisses im Gedächtnis abgelegt wird.

Prozedurales Wissen

Anderson beschreibt prozedurales Wissen als Wissen, wie man etwas macht [And01]. Es ist oftmals implizit und man kann keine Auskunft darüber geben. Beispielsweise kann eine Person wissen, wie man Auto fährt ohne genau widergeben zu können wie. Nach Strube handelt es sich um Wissen, dass in Handlungen umgesetzt wird [Str96].

Prozeduralisierung

Nach Anderson wird mit Prozeduralisierung der Prozess bezeichnet, währenddessen die Verwendung von deklarativem Wissen durch die direkte Anwendung prozeduralen Wissens abgelöst wird [And01]. Dieser Prozess lässt sich in drei Phasen aufteilen, den *drei Phasen beim Erwerb von Fertigkeiten*. Die Phasen werden im Folgenden kurz beschrieben:

- Kognitive Phase: Deklarative Enkodierung der Fertigkeit.
- Assoziative Phase:
  - Fehler aufdecken und entfernen.
  - Einzelne Elemente, die zur Ausführung erforderlich sind, werden stärker miteinander verbunden.

- Autonome Phase: Prozedur wird immer automatisierter und immer schneller.

Durch zunehmende Übung verbessern sich Geschwindigkeit und Genauigkeit.

2.2.6 Kognitive Architekturen

Ausgehend von der Grundhypothese der Kognitionswissenschaft, dass es sich bei kognitiven Prozessen um Berechnungsprozesse handelt, wird kognitive Modellierung ermöglicht. In der Literatur wird dabei von kognitiven Architekturen gesprochen. Wichtige Vertreter sind Soar und ACT-R. In dieser Arbeit wird näher auf ACT-R von John Anderson eingegangen. ACT-R steht für „Adaptive Control of Thought-Rational". Für ausführliche Informationen über Soar sei der interessierte Leser auf das Buch von John Laird verwiesen, einem der Entwickler von Soar [Lai12].

ACT-R blickt auf eine über 40-jährige Entstehungsgeschichte zurück über die im Folgenden ein Überblick gegeben wird.

1973 Anderson und Bower veröffentlichen die HAM Theorie [AB73]. HAM steht für „Human Associative Memory". ACT-R hat seine Wurzeln in der HAM Theorie.

1976 Es folgt ACT [And76].

1983 Auf ACT folgt ACT* [And83].

1993 Mit der Umbenennung in ACT-R erfolgt der Einbau einer rationalen Entscheidungskomponente [And93].

1998 Veröffentlichung von ACT-R 4.0 [AL98].

2004 ACT-R erscheint in der Version 5.0 [ABB+04].

2005 ACT-R erscheint komplett neu geschrieben in der Version 6.0[1].

Nach Anderson et al. handelt es sich bei ACT-R um ein Produktionensystem [ABB+04] [And01]. Das Produktionensystem stellt auch die zentrale Komponente des Systems dar. Über verschiedene Puffer ist das Produktionensystem an die verschiedenen ACT–R–Module angebunden. Mit der Außenwelt kommunizieren das visuelle und das motorische Modul. Weiterhin gibt es noch das Absichtsmodul und ein Modul für deklaratives Wissen. Prozedurales Wissen wird durch das zentrale Produktionensystem simuliert, welches Produktionsregeln enthält. Die Module und Buffer korrespondieren mit bestimmten Hirnregionen.

[1] http://act-r.psy.cmu.edu/software

2.3 Modellierung und Darstellung von Wissen

Das Ziel dieser Arbeit ist die visuelle Analyse von kognitiven Prozessen anhand einer Visualisierung von Eye-Tracking-Daten im WAS-Raum. Dadurch wird die geometrische Darstellung der Eye-Tracking-Daten durch eine semantische Darstellung ersetzt. Die Voraussetzung dafür ist die Annotierung der Stimuli mit AOIs und die Modellierung der AOIs durch eine geeignete Repräsentation. Für eine solche Repräsentation wird im Folgenden ein Konzept beschrieben. Dazu wird zuerst vorgestellt um welches Konzept es sich handelt und anschließend wird beschrieben wie das Konzept visualisiert werden kann.

2.3.1 Modellierung von Wissen

Die Modellierung von Wissen ist eng verknüpft mit der Idee des *Semantic Web*. Diese besteht nach Pellegrini und Blumauer darin, Inhalte des World Wide Web dahingehend anzureichern, dass sie nicht nur für Menschen lesbar sind, sondern es Rechnern ermöglicht wird, auf der Ebene der Bedeutung Automatismen anzuwenden [PB06]. Insbesondere das automatische Schlussfolgern ist hier von Interesse. Ein Konzept das eine hohe Ausdruckstärke besitzt und zur Modellierung von Wissen geeignet ist, stellt nach Gruber eine Ontologie dar [Gru93].

Ontologien werden nach Guarino et al. durch logische Sprachen beschrieben [GOS09]. Dazu gehören die Prädikatenlogik erster und höherer Stufe, Beschreibungslogiken und logische Programmiersprachen. Prädikatenlogiken haben die größte Ausdrucksstärke und bieten somit am wenigsten Mehrdeutigkeit und die größte Unterstützung für automatisches Schließen. Ein großer Nachteil besteht in der Unentscheidbarkeit von Prädikatenlogikgen. Den Vorteil der Entscheidbarkeit bieten zwei Untermengen der Prädikatenlogiken, die logische Programmierung und die Beschreibungslogiken. Zur logischen Programmierung gehört beispielsweise F-Logic und zu den Beschreibungslogiken gehört beispielsweise OWL. Abbildung 2.5 zeigt die Ausdruckstärke der verschiedenen Sprachen. Nach der Betrachtungsweise von Guarino ist jegliches Modell zur Wissensrepräsentation eine Ontologie, je nach Modell aber mit entsprechend abgeschwächter Ausdrucksstärke.

Ontologien in der Philosophie

In der Philosophie bezieht sich der Begriff der Ontologie nach Guarino auf eine philosophische Disziplin [GOS09]. Diese untersucht die Struktur und Natur der Realität. Bereits Aristoteles untersuchte diese Sachverhalte und bezeichnete mit Ontologie die Untersuchung von Attributen, die zu Dingen gehören, nur aufgrund ihrer Natur. Dabei ist es unerheblich, ob die betrachteten Dinge in der Realität tatsächlich existieren.

Ontologien in der Informatik

In der Informatik bezeichnet eine Ontologie nach Guarino eine bestimmte Art von Informationsobjekt oder ein Berechnungsartefakt [GOS09]. Ontologien sind dazu gedacht, die Struktur eines Systems

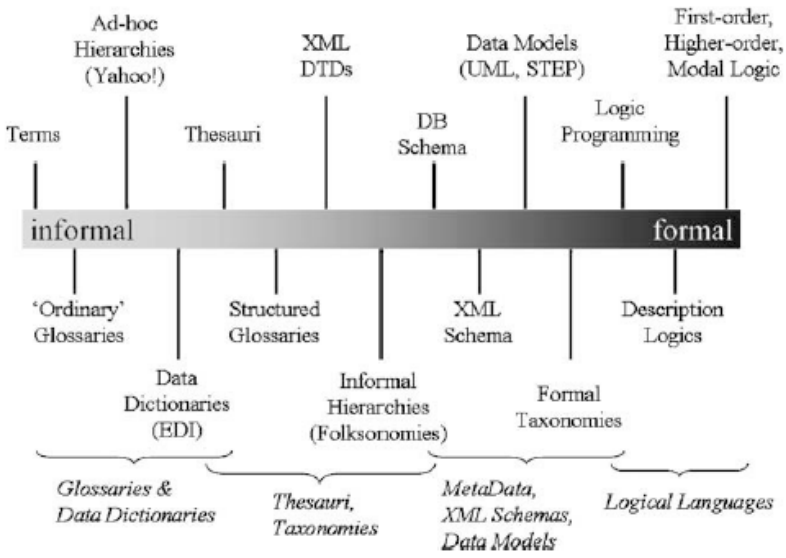

Abbildung 2.5: Ausdruckstärke von verschiedenen Sprachen zur Beschreibung von Ontologien. [GOS09].

formal zu modellieren. Entitäten und Relationen zwischen ihnen sollen möglichst exakt beschrieben werden.

Gruber definierte 1993 den Begriff der Ontologie als „explicit specification of a conceptualization“ [Gru93]. Borst definierte 1997 Ontologien als „formal specification of a shared conceptualization“ [Bor97]. Diese beiden Definitionen wurden 1998 von Studer et al. zusammengeführt in „an ontology is a formal, explicit specification of a shared conceptualization“ [SBF98]. Diese Definition beinhaltet, dass eine Konzeptualisierung eine gemeinsame Sichtweise zwischen mehreren Beteiligten darstellt.

Im Semantic Web wird die Beschreibungssprache OWL dazu benutzt, eine Ontologien zu beschreiben. OWL steht für Web Ontology Language und erlaubt den Aufbau einer Ontologie aus Klassen, Properties und Instanzen.

2.3.2 Darstellung von Wissen

Zur Modellierung von Wissen kommen OWL Ontologien zum Einsatz. Diese können visuell als Graph dargestellt werden. Negru und Lohmann stellen dazu die VOWL Visualisierung von Ontologien vor [NL13]. VOWL bedeutet Visual Notation for OWL Ontologies. Abbildung 2.6 zeigt die Visualisierung einer Beispiel Ontologie mit VOWL.

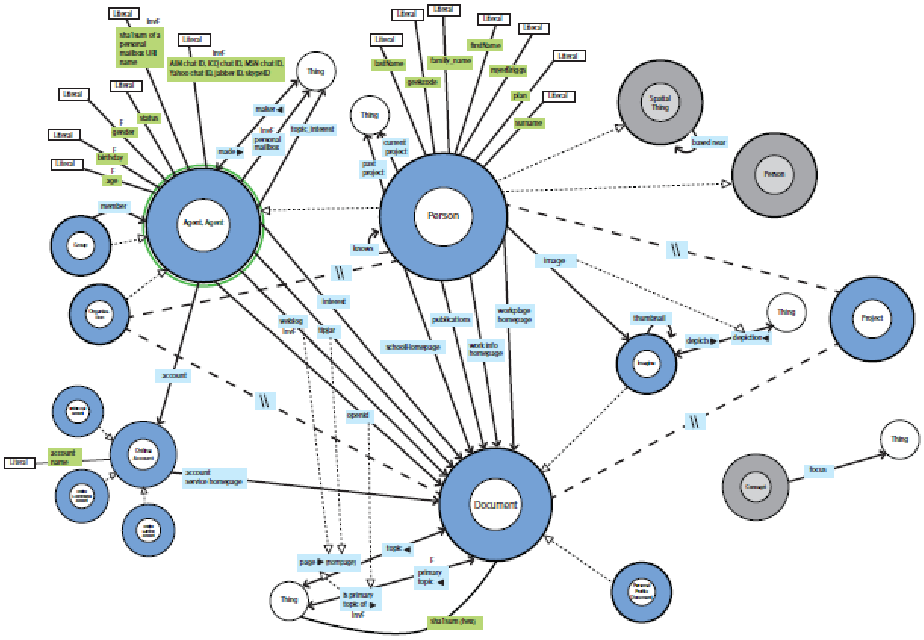

Abbildung 2.6: VOWL Visualisierung einer Beispiel Ontologie [NL13].

3 Existierende Arbeiten

Dieses Kapitel bietet einen Einblick in Arbeiten die mit der vorliegenden Arbeit verwandt sind. Insbesondere wird in diesem Kapitel die Augenbewegungs–Simulation beschrieben, für die durch das Konzept der vorliegenden Arbeit Eingabedaten erstellt werden sollen.

Vorgestellt wird in Abschnitt 3.1 das Simulations–Framework von Engelhardt zur Simulation von Augenbewegungen [Eng13]. Abschnitt 3.2 beschäftigt sich mit einer Visualisierungstechnik für dynamische Prozesse und Abschnitt 3.3 stellt eine Software vor, mit der Eye–Tracking–Daten verschiedener Probanden verglichen werden können und mit der nach Mustern in den Daten gesucht werden kann.

3.1 Augenbewegungs Simulation

In der Arbeit von Engelhardt wird eine Simulation entwickelt, die Augenbewegungen simuliert und Fixationszeiten vorhersagen kann. Beispielsweise kann der Simulation als Stimulus ein Balkendiagramm mit zwölf Balken übergeben werden. Dann stellt man an die Simulation die Anfrage, die Höhe zweier Balken zu vergleichen und anzugeben welcher der höhere der beiden Balken ist. Die Simulation führt dann die Augenbewegungen durch, die notwendig sind, um die Anfrage beantworten zu können. Die Oberfläche der Simulation ist in Abbildung 3.1 dargestellt. Daran erkennt man, dass die von der Simulation zur Lösung der Aufgabe ausgeführten Augenbewegungen und Fixationen auf dem Stimulus in Bereich (6) eingezeichnet sind.

Die Simulation ist mit Produktionsregeln ausgestattet, aus denen hervorgeht welche Augenbewegungen für die Anfrage auf dem Stimulus auszuführen sind. Dabei sind die Produktionsregeln so gestaltet, dass sie für beliebige Stimuli eines Visualisierungs–Typs zur Beantwortung der Anfrage anwendbar sind. Dies wird erreicht, durch die Nutzung eines sogenannten *Visualisierungsschemas*. Dabei handelt es sich um eine abstrakte Beschreibung eines Visualisierungs–Typs. Diese Beschreibung basiert auf dem Schema von Pinker zur abstrakten Beschreibung des Aufbaus von grafischen Darstellungen [Pin90]. Ein Visualisierungsschema beschreibt den Aufbau einer Visualisierung durch das Zusammensetzen von Visualisierungs–Elementen und den Beziehungen der Visualisierungs–Elemente untereinander. Bei einem Visualisierungs–Element handelt es sich beispielsweise um eine Koordinatenachse, eine Linie oder einen Balken. Die Beziehungen der Visualisierungs–Elemente untereinander werden durch sogenannte *Orientierungsrelationen* beschrieben. Eine Orientierungsrelation gibt an in welchem räumlichen Zusammenhang zwei Visualisierungs–Elemente zueinander stehen, wie etwa „rechts“, „links“, „oben“, „unten“, „parallel“, „orthogonal“. Das Visualisierungs–Element Balken ist beispielsweise über die Orientierungsrelation „oben“ mit dem Visualisierungs–Element X–Achse verbunden. Das Visualisierungs–Element X–Achse wiederum ist über die Orientierungsrelation „unten“ mit dem Visualisierungs–Element Balken verbunden. Das Visualisierungsschema wird

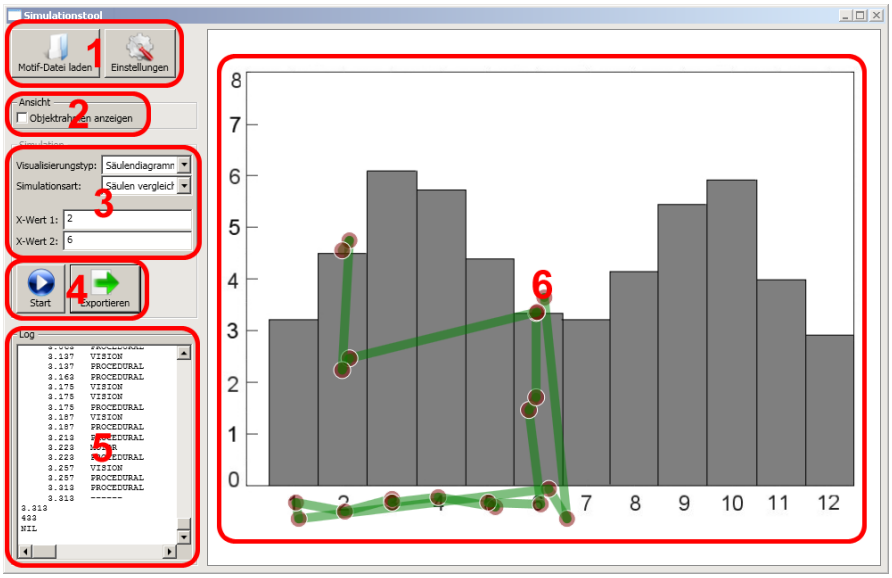

Abbildung 3.1: Die Oberfläche der Augenbewegungs–Simulation. In Bereich (6) wird der Stimulus mit den zur Lösung einer Aufgabe ausgeführten Augenbewegungen und Fixationen angezeigt. Die Aufgabe lautet die Höhe zweier Balken zu vergleichen und anzugeben welcher der höhere der beiden Balken ist [Eng13].

in Form einer Ontologie umgesetzt. Es ist Teil einer Ontologie–Hierarchie, deren einzelne Schichten jeweils Ontologien sind und deren Klassen jeweils Instanzen der darüber liegenden Ontologie bilden. Die Ontologie–Hierarchie ist in Abbildung 3.2 dargestellt. Die oberste Schicht enthält die *Ontologie*. Sie definiert Visualisierungs–Elemente für den Aufbau von Visualisierungen. Die mittlere Schicht enthält das eben beschriebene *Visualisierungsschema*. In der untersten Schicht ist das Visualisierungsmodell enthalten, das die zu simulierende Visualisierung modelliert, mit allen Relationen und Visualisierungs–Elementen.

Bei der Durchführung einer Simulation werden die Anfrage, das Visualisierungsmodell, die Visualisierungsschemata und die Ontologie als Eingabe an die Simulation übergeben. Diese berechnet mit Hilfe einer ACT–R Kognitions–Simulation den Pfad der Augenbewegungen und Fixationen, die zur Beantwortung der Anfrage notwendig sind. Dabei werden für jedes visuelle Element, das zur Beantwortung der Anfrage fokussiert werden muss, die folgenden Schritte ausgeführt:

1. Zu suchendes Element ermitteln
2. Element per visueller Suche suchen
3. Element fokussieren

Die Produktionsregeln werden in der Arbeit von Engelhardt künstlich auf technisch motivierte Art und Weise defininiert. In der vorliegenden Ausarbeitung wird ein Konzept entwickelt, durch das es möglich wird, die Produktionsregeln aus realen Augenbewegungen von Menschen abzuleiten. Sie sollen dazu aus den Daten von Eye–Tracking–Studien gewonnen werden. Dazu wird die in der

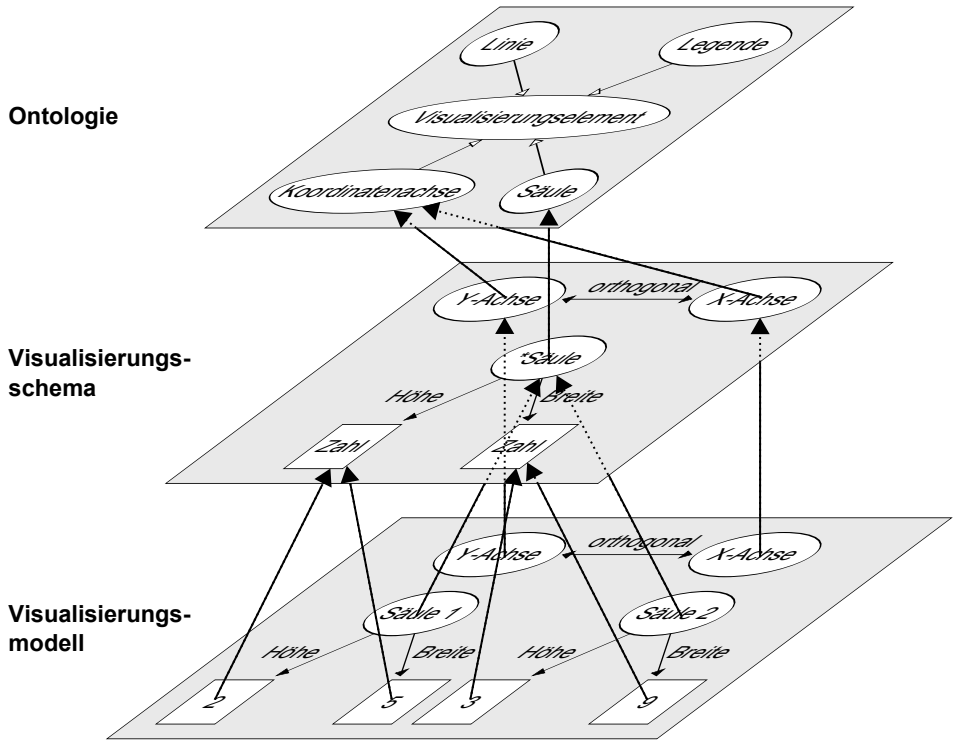

Abbildung 3.2: Die Ontologie–Hierarchie von Engelhardt. Die Ontologie–Hierarchie enthält drei Ontologien in drei Schichten. Die Klassen der jeweils unteren Ontologien bilden Instanzen der jeweiligen Klassen der darüberliegenden Ontologie [Eng13].

Arbeit von Engelhardt verwendete Ontologie–Hierarchie im Konzept der vorliegenden Ausarbeitung weiterverwendet.

3.2 Visualisierung dynamischer Prozesse

In der vorliegenden Arbeit findet die Analyse von Eye–Tracking–Daten im WAS–Raum statt. Dies bedeutet, dass Abfolgen von betrachteten AOIs von mehreren verschiedenen Probanden im zeitlichen Verlauf untersucht werden. Dazu ist eine geeignete Visualisierungstechnik notwendig. Im Folgenden wird dazu die Visualisierungstechnik des Scarf Plots vorgestellt.

Ein Scarf Plot stellt eine Abfolge von AOIs in ihrem zeitlichen Verlauf dar. Das englische Wort „scarf“ bedeutet im Deutschen „Schal“. Dies liegt an dem optischen Eindruck einer Scarf Plot Visualisierung, sie erinnert vom Aussehen her an einen Schal. Bei der Scarf Plot Visualisierung können Abfolgen von AOIs von mehreren Probanden in ihrer zeitlichen Reihenfolge dargestellt werden. Verschiedene AOIs werden dabei durch verschiedene Farben repräsentiert.

Richardson und Dale verwenden die Scarf Plot Darstellung in ihrer Arbeit über den Vergleich von Augenbewegungen von Sprechern und Zuhörern [RD05]. Dabei wird zuerst den Sprechern auf einem

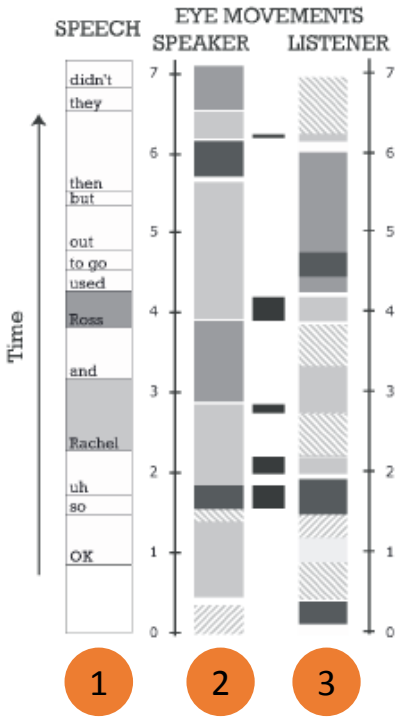

Abbildung 3.3: Zwei Scarf Plots. Die beiden Scarf Plots vergleichen die Augenbewegungen eines Sprechers und eines dem Sprecher zuhörenden Zuhörers während beide dieselbe Fernsehsendung gezeigt bekommen und der Sprecher über die Fernsehsendung spricht. (1) zeigt das Gesprochene, (2) die Augenbewegungen des Sprechers und (3) die Augenbewegungen des Zuhörers [RD05].

Fernsehbildschirm eine Fernsehsendung gezeigt, über die die Sprecher während des Zusehens erzählen. Es werden dabei die Augenbewegungen der Sprecher mittels Eye–Tracking aufgenommen. Weiterhin wird die Erzählung der Sprecher aufgezeichnet. Die Aufnahme der Erzählung wird anschließend den Zuhörern vorgespielt während diese dieselbe Fernsehsendung gezeigt bekommen und ihre Augenbewegungen aufgezeichnet werden. Anschließend werden die Augenbewegungen der Sprecher und Zuhörer anhand eines Scarf Plots miteinander verglichen. Abbildung 3.3 zeigt den Vergleich mittels zweier Scarf Plots. Bereich (1) der Abbildung beinhaltet das Gesprochene, Bereich (2) zeigt den Scarf Plot des Sprechers und in Bereich (3) ist der Scarf Plot des Zuhörers dargestellt.

Kurzhals et al. verwenden einen Scarf Plot in ihrer Software „ISeeCube“ [KHW14]. Mit der Software können Eye–Tracking–Daten auf dynamischen Stimuli, wie etwa Videos, untersucht werden. Die AOI Abfolgen eines Probanden werden farbkodiert anhand eines horizontalen Scarf Plots visualisiert. Abfolgen mehrerer Probanden werden durch mehrere dieser Scarf Plots dargestellt, die übereinander gestapelt werden. Durch die Zuweisung eines eindeutigen Buchstaben an jede AOI können Ähnlich-

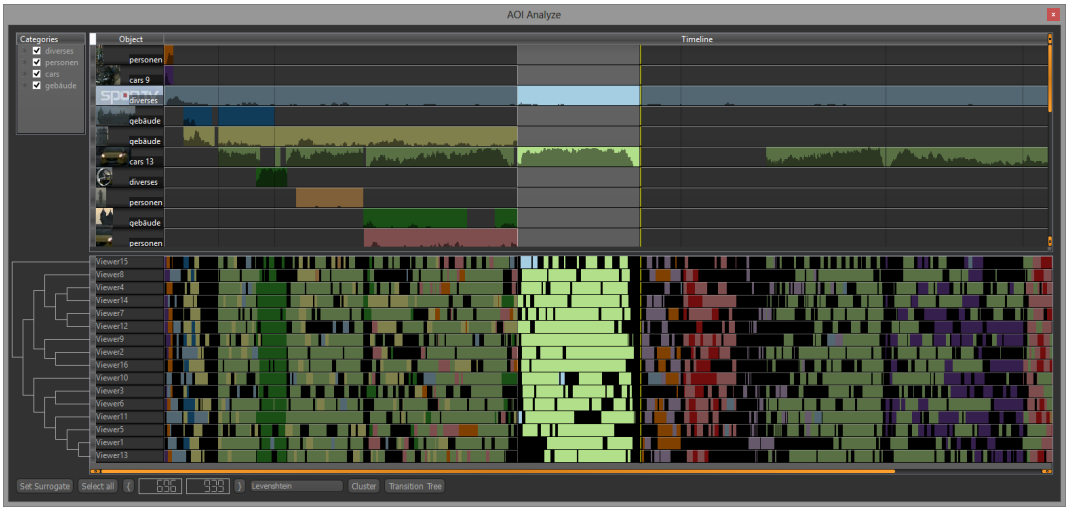

Abbildung 3.4: Scarf Plots in der Software ISeeCube. Über die Scarf Plots in ISeeCube können die AOI Abfolgen von mehreren Probanden während der Betrachung dynamischer Stimuli analysiert werden [KHW14].

keitsberechnungen mittels der Levenshtein–Distanz auf den AOI Abfolgen durchgeführt werden. Abbildung 3.4 zeigt die Scarf Plot Visualisierung in ISeeCube.

Scarf Plots bieten die Möglichkeit lange AOI Abfolgen von mehreren Probanden zu analysieren. Durch die Zuweisung von Farben an die AOIs bietet sich für den Benutzer die Möglichkeit Muster in den Abfolgen verschiedener Probanden zu erkennen. Die Scarf Plot Visualisierung wird in einer modifizierten Variante und ergänzt um Möglichkeiten zur Mustererkennung in das Konzept der vorliegenden Ausarbeitung aufgenommen.

3.3 Mustersuche und Vergleich von Eye–Tracking–Daten

Um die Produktionsregeln für die Augenbewegungs–Simulation zu erstellen ist es notwendig eine Lösungsstrategie auf den Eye–Tracking–Daten zu erkennen, aus denen die Produktionsregeln abgeleitet werden sollen. Da die Eye–Tracking–Daten die Augenbewegungen mehrerer Probanden enthalten, ist zu erwarten, dass verschiedene Probanden ähnliche Lösungsstrategien verwenden. Dabei werden die Scanpfade der Probanden sich zwar ähneln, aber nicht exakt übereinstimmen. Um untersuchen zu können, ob und wenn ja wie viele der Probanden eine ähnliche Lösungsstrategie verwenden, werden Verfahren zur Ähnlichkeitsbestimmung und zur Mustersuche in das Konzept dieser Ausarbeitung integriert. Die nachfolgend vorgestellte Arbeit beinhaltet Verfahren zur Ähnlichkeitsbestimmung von Eye–Tracking–Daten verschiedener Probanden und zur Suche von Mustern auf den Daten.

Die Software „eyePatterns“ von West et al. visualisiert Sequenzen von Fixationen in AOIs. Den AOIs werden Buchstaben zugewiesen. Eine Abfolge von Fixationen eines Probanden über verschiedene AOIs wird als Zeichenkette der zu den AOIs zugehörigen Buchstaben dargestellt. Dabei besteht

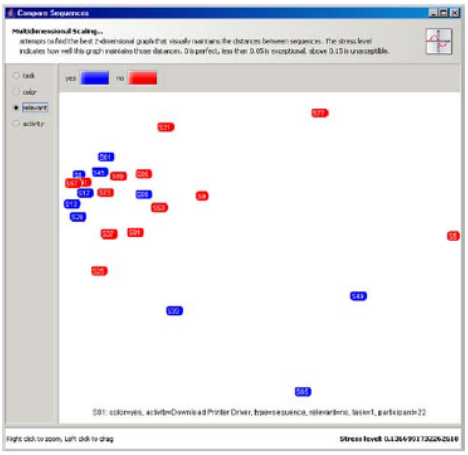

Abbildung 3.5: Abbildung der Ähnlichkeit von Fixations–Sequenzen. Mittels des Verfahrens der multidimensionalen Skalierung wird die Ähnlichkeit mehrerer Fixations–Sequenzen auf ein zweidimensionales Koordinatensystem abgebildet [WHRK06].

die Möglichkeit mehrfache direkt aufeinanderfolgende Fixationen innerhalb derselben AOI, durch mehrfaches Auftreten desselben Buchstabens in der resultierenden Zeichenkette zu repräsentieren. Dies wird in der Software mit „Expanded Sequence“ bezeichnet. Die Darstellung kann auch verkürzt erfolgen, welche in der Software den Namen „Collapsed Sequence“ trägt. Dabei werden mehrfache direkt aufeinanderfolgende Fixationen eines Probanden innerhalb derselben AOI durch nur einen Buchstaben repräsentiert. Beispielsweise werden die Fixationen in den AOIs „AABCBB“ in der langen Variante zu „AABCBB“ und in der kurzen Variante zu „ABCB“. Die Zeichenkette wird für Ähnlichkeitsvergleiche herangezogen. Dazu stehen der Levenshtein–Distanz–Algorithmus und der Algorithmus von Needleman und Wunsch zur Verfügung [NW70]. Weiterhin steht das Verfahren der multidimensionalen Skalierung von Kruskal und Wish zur Verfügung [KW78]. Dabei wird die Ähnlichkeit zwischen den Zeichenketten und damit den zugehörigen Sequenzen auf ihren Abstand in einem zweidimensionalen Koordinatensystem abgebildet. Dies ist in Abbildung 3.5 abgebildet.

EyePatterns bietet zudem die Funktionalität in den Zeichenketten der Sequenzen nach Teilzeichenketten zu suchen. Für die Suche kann die Länge der Teilzeichenkette bestimmt werden. Dabei kann nach exakten und unscharfen Übereinstimmungen mit der Zeichenkette der Suchanfrage gesucht werden. Die unscharfe Suche wird mit regulären Ausdrücken durchgeführt.

In das Konzept der vorliegenden Ausarbeitung soll ein Verfahren zur Ähnlichkeitsbestimmung der Eye–Tracking–Daten von verschiedenen Probanden und zur Suche von Mustern in den Daten integriert werden

4 Aufgabe und Lösungsansatz

Das Ziel dieser Arbeit ist die Entwicklung eines Konzepts, mit dem durch die visuelle Analyse von AOI Abfolgen, über die Ausnützung der Eye–Mind–Hypothese, kognitive Prozesse sichtbar gemacht werden. Basierend auf der visuellen Analyse der kognitiven Prozesse soll eine Lesevorschrift für einen Visualisierungstyp erzeugt werden.

Abschnitt 4.1 stellt das Szenario vor, aus dem sich die Aufgabe dieser Arbeit ergibt, Abschnitt 4.2 enthält eine Beschreibung der Aufgabe und Abschnitt 4.3 beschreibt den Ansatz zur Lösung der Aufgabe.

4.1 Szenario

Um festzustellen, ob eine neue Visualisierung einen Mehrwert gegenüber vorhandenen Visualisierungen bietet und ob sie grundsätzlich leicht zu lesen ist, werden typischerweise Eye–Tracking–Studien durchgeführt. Solche Studien sind zeitaufwändig und teuer. Sie erfordern einen Eye–Tracker dessen Anschaffung mehrere tausend Euro kostet. Es werden ein oder mehrere Personen benötigt, die sowohl die Pilotstudie als auch die eigentliche Studie vorbereiten und durchführen. Weiterhin werden Probanden benötigt, die eine Aufwandsentschädigung erhalten. Wird die neue Visualisierung aufgrund der Ergebnisse der Studie verbessert, so muss erneut eine Studie durchgeführt werden.

Zur Verminderung oder gar Vermeidung dieses Aufwands und der Kosten, wurde von Engelhardt eine Software zur Simulation von Augenbewegungen entwickelt [Eng13]. Die Simulation simuliert das Blickverhalten, das zur Lösung einer Aufgabe auf einem Stimulus notwendig ist. Ein Beispiel dafür ist etwa das Ablesen der Höhe eines bestimmten Balkens in einem Balkendiagramm. Die Simulation simuliert die Fixationen und Sakkaden, die notwendig sind, um die Höhe des Balkens abzulesen.

Das Herzstück der Simulation ist eine ACT–R Kognitions–Simulation. Sie simuliert die kognitiven Prozesse, die bei einem realen Probanden die Augenbewegungen steuern, die zur Lösung der gestellten Aufgabe ausgeführt werden. Anhand der simulierten kognitiven Prozesse werden die Augenbewegungen simuliert. Zur Ausführung der Simulation werden die Ontologie–Hierarchie, der Stimulus, die Aufgabe, die auf dem Stimulus bearbeitet werden soll und ein Satz von Produktionsregeln benötigt. Die Ontologie–Hierarchie, der Stimulus, die Aufgabe und der Satz von Produktionsregeln werden daher als Eingabedaten an die Simulation übergeben.

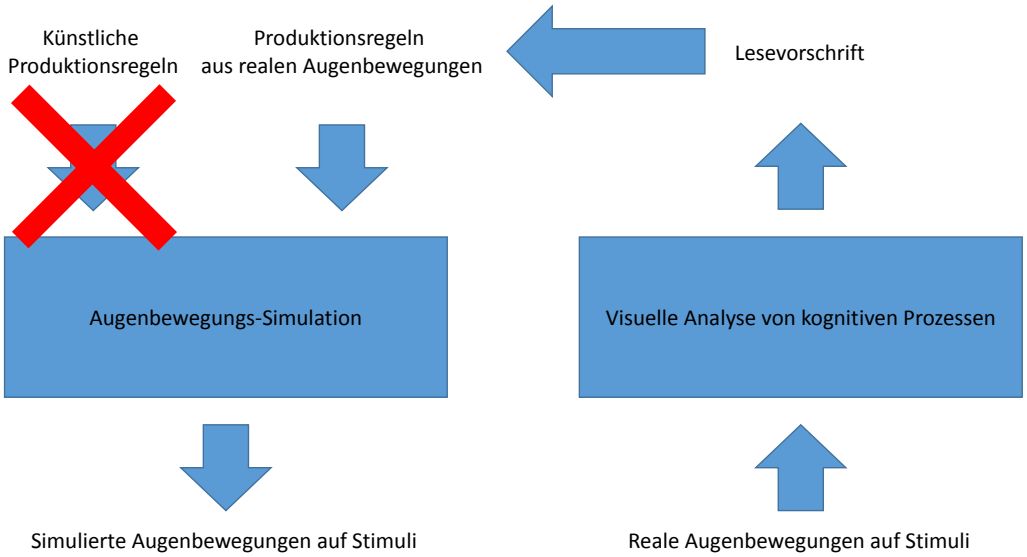

Abbildung 4.1: Zusammenhang zwischen dieser Ausarbeitung und der Augenbewegungs–Simulation. Für die Augenbewegungs–Simulation sollen anhand des Konzepts der vorliegenden Ausarbeitung Produktionsregeln aus realen Augenbewegungen abgeleitet werden.

4.2 Aufgabe

Im existierenden Simulations–Framework werden künstliche Produktionsregeln verwendet, deren Definition technisch motiviert ist. Um Augenbewegungen wie bei einem Menschen simulieren zu können, sollen die Produktionsregeln aus Daten von Eye–Tracking–Studien abgeleitet werden. Die Produktionsregeln sollen also nicht definiert sondern abgeleitet werden. In der vorliegenden Arbeit wird ein Konzept entwickelt, dass die Gewinnung einer Lesevorschrift aus Eye–Tracking–Daten ermöglicht. Eine Lesevorschrift bildet die Vorstufe zu einem Satz von Produktionsregeln. Diese Vorstufe kann in einen Produktionsregelsatz überführt werden. Für eine Lesevorschrift gelten die gleichen Besonderheiten und Einschränkungen wie für einen Satz von Produktionsregeln. Abbildung 4.1 zeigt den Zusammenhang zwischen dem existierenden Framework und der vorliegenden Arbeit.

4.3 Lösungsansatz

Die Herausforderung dieser Arbeit liegt in der Überführung von Eye–Tracking–Daten in eine Lesevorschrift, der Vorstufe eines Satzes von Produktionsregeln. Dazu wird ein Verfahren entwickelt, das die Besonderheiten und Einschränkungen von Lesevorschriften berücksichtigt. Das Verfahren soll die Überführung von Stimulus bezogenen Eye–Tracking–Daten in eine Visualisierungstyp bezogene Lesevorschrift ermöglichen. Um diese Überführung zu bewerkstelligen, wird die Ontologie–Hierarchie

in einer modifizierten Form verwendet. Die Hierarchie ist notwendig für die Zuordnung von Stimulus bezogenen Daten zu Visualisierungstyp bezogenen Daten.

Das Verfahren soll dem Benutzer die Wahl überlassen, ob er eine weitgehend automatisierte Vorgehensweise bevorzugt, lieber von Hand die Analysen vornimmt und mit den Ergebnissen weiterarbeitet oder eine Kombination aus beidem nutzen möchte. Für das automatisierte Vorgehen sollen auf Stringvergleichen basierende Verfahren zur Ähnlichkeitsbestimmung von Scanpfaden zum Einsatz kommen. Für die manuelle Herangehensweise werden mehrere Visualisierungen entwickelt oder bestehende Visualisierungen angepasst, um mit deren Hilfe ein Ergebnis erzeugen zu können. Zur Stärkung der Aussagekraft soll das Konzept große Datenmengen verarbeiten können. Da durch große Datenmengen Probleme wie Visual Clutter zu erwarten sind, soll die manuelle Analyse zusätzlich durch Verfahren zur Mustersuche unterstützt werden.

Eine Lesevorschrift gilt nur für eine Aufgabe. Mit einer Sammlung von verschiedenen Lesevorschriften, die für verschiedene Aufgaben auf einem Visualisierungstyp gelten, können somit die Augenbewegungen für verschiedene Aufgaben auf beliebigen Instanzen dieses Visualisierungstyps simuliert werden. Erstellt man weiterhin mehrere dieser Sammlungen für unterschiedliche Visualisierungstypen, ermöglicht dies die Simulation der Augenbewegungen auf den unterschiedlichsten Stimuli mit den verschiedensten Aufgaben.

Ein Satz von Produktionsregeln legt fest in welcher Reihenfolge visuelle Elemente einer Visualisierung per visueller Suche gesucht und fixiert werden müssen, um eine gestellte Aufgabe lösen zu können. Das Besondere hierbei ist, dass dieser Produktionsregelsatz sich auf einen Visualisierungstyp bezieht und nicht nur auf einen einzigen Stimulus, also der Instanz eines Visualisierungstyps. Dies hat den Vorteil, dass die Simulation der Augenbewegungen für beliebige Instanzen eines Visualisierungstyps durchgeführt werden kann. Erreicht wird dies dadurch, dass ein Produktionsregelsatz die Klassen des zum Visualisierungstyp zugehörigen Visualisierungsschemas in der Reihenfolge enthält, wie sie von der Simulation zur Aufgabenlösung zu traversieren sind. Wichtig hierbei ist, dass bei der Erzeugung eines Produktionsregelsatzes darauf geachtet wird, eine Klasse nur dann aufzunehmen, wenn ihre Instanz, also das visuelle Element auf dem Stimulus, zur Aufgabenlösung fokussiert wird und nicht wenn die Fokussierung während der visuellen Suche oder während des Cross–Checkings geschieht. Dies erfolgt aus dem Grund, da die visuelle Suche nicht im Produktionsregelsatz enthalten sein soll, da sie ohnehin für jede Klasse im Produktionsregelsatz ausgeführt wird, um die passende Instanz dieser Klasse zu finden. Ebenso soll ein Produktionsregelsatz kein Cross–Checking enthalten, da dies von der Simulation durch künstliches Rauschen simuliert wird. Die Beschränkung eines Satzes von Produktionsregeln auf zur Aufgabenlösung relevante Klassen muss bei der Erzeugung eines Produktionsregelsatzes, und damit auch bei der Erzeugung einer Lesevorschrift, berücksichtigt werden.

5 Lösungskonzept

Dieses Kapitel beschreibt das Konzept zur Gewinnung einer Lesevorschrift mit Hilfe der visuellen Analyse von kognitiven Prozessen. Die kognitiven Prozesse werden durch die visuelle Analyse von AOI Abfolgen, unter Ausnützung der Eye–Mind–Hypothese, sichtbar gemacht. Eine Lesevorschrift ist die Vorstufe zu einem Satz von Produktionsregeln für die Simulation von Augenbewegungen. Jede Lesevorschrift kann in einen Produktionsregelsatz überführt werden. Zur Erzeugung einer Lesevorschrift werden die Daten aus Eye–Tracking–Studien analysiert und basierend auf der Analyse in eine Lesevorschrift überführt. Für die Überführung von Eye–Tracking–Daten in eine Lesevorschrift werden die Stimuli, auf denen die Augenbewegungen erfasst wurden, die Eye–Tracking–Daten selbst und eine modifizierte Variante der Ontologie–Hierarchie von Engelhardt als Eingabedaten für das Konzept der vorliegenden Ausarbeitung verwendet [Eng13]. Die Ontologie–Hierarchie dient in ihrer unveränderten Variante als Eingabe für das Simulations–Framework, dessen Funktionsweise in Abschnitt 3.1 beschrieben ist. Für das Konzept dieser Ausarbeitung wird ein Ablauf entwickelt, in dessen Verlauf mit Hilfe eines ebenso zu entwickelnden Visualisierungs– und Analysekonzepts die Eingabedaten in eine Lesevorschrift überführt werden. Dabei werden die Verläufe der AOI–Besuche der Probanden auf den Eingabe–Stimuli untersucht und die Ergebnisse der Untersuchung zu einer Lesevorschrift weiterverarbeitet. Abbildung 5.1 zeigt die Zusammenhänge zwischen der Augenbewegungs–Simulation, dem Konzept der vorliegenden Arbeit und den jeweiligen Eingabedaten.

Dieses Konzeptkapitel besteht aus folgenden Abschnitten: In Abschnitt 5.1 wird der Zusammenhang zwischen kognitiven Prozessen und AOI Abfolgen erläutert. Abschnitt 5.2 beschreibt die Anpassung der Ontologie–Hierarchie der Augenbewegungs–Simulation zur Verwendung im Konzept der vorliegenden Arbeit. In Abschnitt 5.3 wird erklärt, was eine Lesevorschrift ist und der Zusammenhang zwischen einer Lesevorschrift und einem Satz von Produktionsregeln dargelegt. Abschnitt 5.4 enthält Details darüber warum eine Ontologie für AOIs benötigt wird und wie diese Ontologie aussieht. Der Abschnitt 5.5 zeigt den Ablauf zur Gewinnung einer Lesevorschrift. Wie AOI–Besuche zusammengefasst werden können, zeigt Abschnitt 5.6. Und schließlich wird in Abschnitt 5.7 das Visualisierungs– und Analysekonzept vorgestellt mit dessen Hilfe eine Lesevorschrift gewonnen werden kann.

5.1 Zusammenhang zwischen kognitiven Prozessen und AOI Abfolgen

Die Eye–Mind–Hypothese besagt, dass ein Zusammenhang zwischen Denken und Sehen besteht. Auf dieser Grundlage wird davon ausgegangen, dass Eye-Tracking-Daten implizit Informationen über die kognitiven Prozesse enthalten, die während der Betrachtung der zugehörigen Visualisierung ablaufen. Dieser Zusammenhang erlaubt die Zuordnung einer AOI zu dem ihr entsprechenden deklarativen Gedächtnisinhalt. Die Übergänge zwischen zwei AOIs werden durch prozedurales Wissen gesteuert.

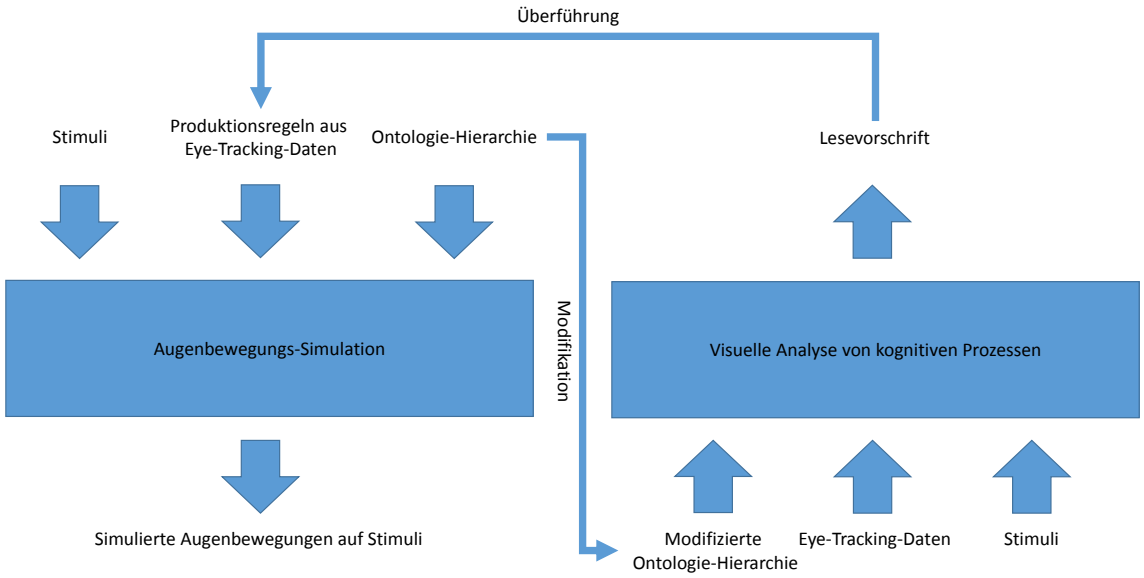

Abbildung 5.1: Zusammenhang zwischen Lösungskonzept, Augenbewegungs–Simulation und jeweiligen Eingabedaten. Aus Eye–Tracking–Daten und Stimuli sollen mit Hilfe der modifizierten Ontologie–Hierarchie Lesevorschriften erzeugt werden. Die Lesevorschriften sind die Vorstufe eines Produktionsregelsatzes für die Augenbewegungs–Simulation.

Somit werden durch die Visualisierung von Abfolgen von AOIs die dazugehörigen deklarativen und prozeduralen kognitiven Prozesse sichtbar gemacht. Die Augenbewegungs–Simulation arbeitet auf die gleiche Weise. Deklarative Gedächtnisinhalte entsprechen visuellen Elementen auf dem Stimulus und werden durch Chunks repräsentiert. Die Steuerung der Augenbewegungen erfolgt durch Produktionsregeln, die auf den Chunks operieren.

5.2 Anpassung der Ontologie–Hierarchie

Für die Gewinnung einer Lesevorschrift ist die Ontologie–Hierarchie von Engelhardt in einer modifizierten Variante notwendig [Eng13]. Benötigt wird insbesondere das Visualisierungsschema, das sich auf den Visualisierungstyp bezieht, für den die Lesevorschrift erstellt werden soll. Das Schema bildet die mittlere Schicht der Drei–Schichten–Ontologie–Hierarchie. Zum einen werden für die vorliegende Ausarbeitung die obersten beiden Schichten der Hierarchie umbenannt. Zum anderen wird die Ontologie der untersten Schicht durch eine andere ersetzt. Zunächst wird die Umbenennung der Schichten beschrieben. Die folgende Aufzählung zeigt welche Schicht welchen neuen Namen erhält:

- Die Schicht „Ontologie“ wird in „Visualisierungs–Elemente–Ontologie“ umbenannt.
- Die mittlere Schicht „Visualisierungsschema“ heißt nun „Visualisierungs–Schema–Ontologie“.

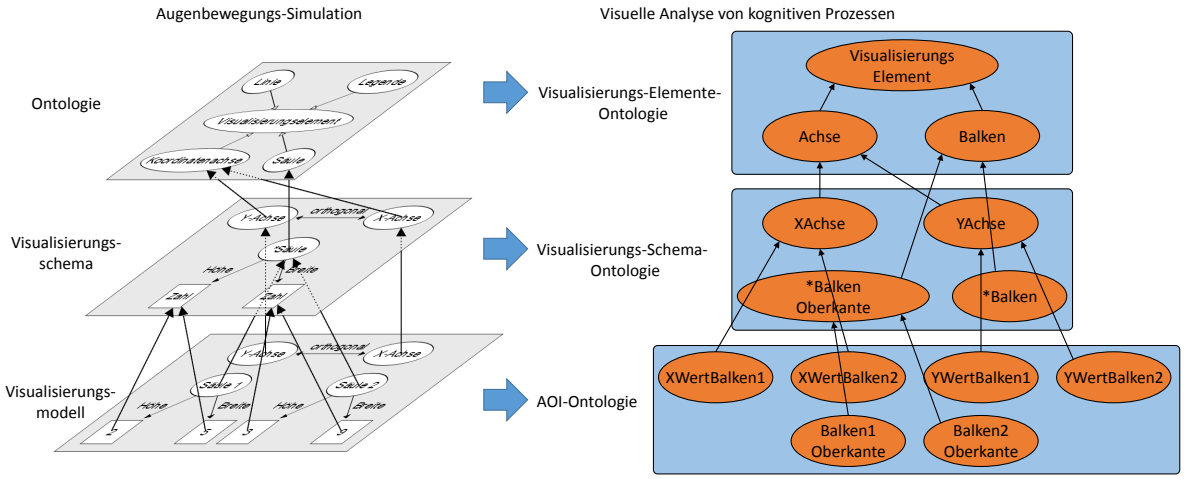

Abbildung 5.2: Ontologie–Hierarchie im Original und modifiziert. Links befindet sich die Ontologie–Hierarchie in der Form wie sie bei der Augenbewegungs–Simulation zum Einsatz kommt [Eng13]. Rechts ist die für die vorliegende Ausarbeitung angepasste Variante der Ontologie–Hierarchie zu sehen.

Im Folgenden wird die Ersetzung der untersten Schicht beschrieben. Um die Ontologie–Hierarchie für das Konzept der vorliegenden Arbeit verwenden zu können, wird das Visualisierungs–Modell durch eine Ontologie aus den AOIs ersetzt, die auf den Stimuli der Eingabedaten annotiert sind. Abbildung 5.2 zeigt die Ontologie–Hierarchie der Augenbewegungs–Simulation und die modifizierte Ontologie–Hierarchie bei der visuellen Analyse von kognitiven Prozessen. In der Abbildung beschreibt die Ontologie aus AOIs ein Balkendiagramm mit zwei Balken. Relationen zwischen und Informationen zu den einzelnen Klassen werden in der Abbildung nicht dargestellt. Die Klassen der jeweiligen unteren Schicht stehen in einer subclass–of Beziehung zu der entsprechenden Klasse der jeweiligen darüber liegenden Schicht. Dies ermöglicht die Vererbung von Merkmalen. Die Einbindung der Ontologie von AOIs in die Hierarchie ermöglicht die Zuordnung von AOIs über ihre Ontologie–Klassen zu ihren Visualisierungs–Schema–Ontologie–Klassen.

5.3 Lesevorschrift

Eine Lesevorschrift ist die Vorstufe zu einem Satz von Produktionsregeln für eine Augenbewegungs–Simulation. Die Lesevorschrift bezieht sich auf einen bestimmten Visualisierungstyp und gilt für eine bestimmte Aufgabe. Die Lesevorschrift enthält die Klassen der zum Visualisierungstyp zugehörigen Visualisierungs–Schema–Ontologie in der Reihenfolge, wie sie von der Simulation zur Aufgabenlösung zu traversieren sind. Dadurch kann die Simulation für beliebige Stimuli des Visualisierungstyps durchgeführt werden. Listing 5.1 zeigt eine Lesevorschrift für den Visualisierungstyp Balkendiagramm mit der Aufgabe die Höhe eines bestimmten Balkens abzulesen. Die Klassen sind hierbei durch ihre

Listing 5.1 Lesevorschrift für den Visualisierungstyp Balkendiagramm. Die Aufgabe, für die die Lesevorschrift gilt, ist die Höhe eines bestimmten Balkens abzulesen. Die Lesevorschrift enthält die Klassen der Visualisierungs–Schema–Ontologie für Balkendiagramme in der Reihenfolge, wie sie von der Simulation zur Aufgabenlösung abzuarbeiten sind.

```
<!-- Visuelle Suche und Fokussierung des X-Werts des abzulesenden Balkens auf der X-Achse -->
http://www.semanticweb.org/vaocp/BalkenDiagrammVisualisierungsSchemaOntologie#XAchse

<!-- Visuelle Suche und Fokussierung der Oberkante des abzulesenden Balkens -->
http://www.semanticweb.org/vaocp/BalkenDiagrammVisualisierungsSchemaOntologie#BalkenOberkante

<!-- Visuelle Suche und Fokussierung des Y-Werts des abzulesenden Balkens auf der Y-Achse -->
http://www.semanticweb.org/vaocp/BalkenDiagrammVisualisierungsSchemaOntologie#YAchse
```

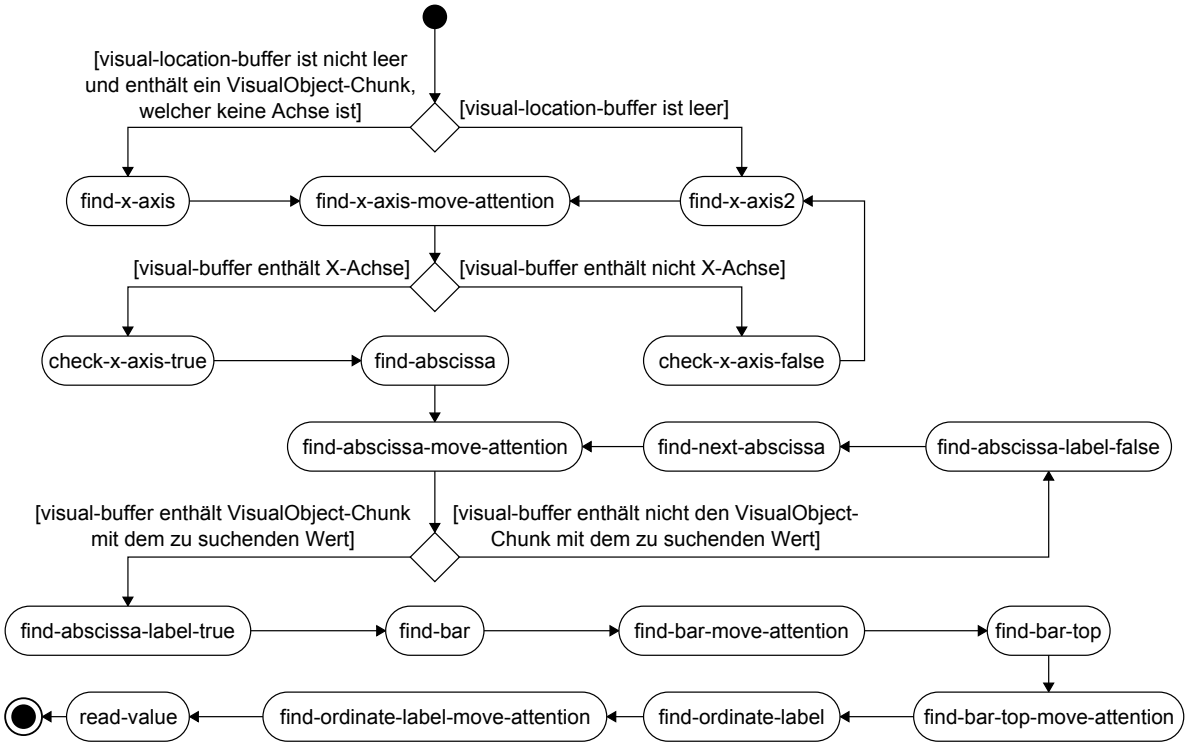

Abbildung 5.3: Produktionsregelsatz für den Visualisierungstyp Balkendiagramm. Der Produktionsregelsatz gilt für die Aufgabe die Höhe eines bestimmten Balkens abzulesen [Eng13].

URIs angegeben. Zum Vergleich ist in Abbildung 5.3 ein Beispiel für einen Satz von Produktionsregeln zum Ablesen der Höhe in einem Balkendiagramm als Aktivitätsdiagramm dargestellt.

Die Simulation arbeitet mit den Instanzen der Klassen der Lesevorschrift. Diese Instanzen entsprechen den zu fokussierenden visuellen Elementen. Während der Abarbeitung der Lesevorschrift benötigt die Simulation zur Suche des nächsten zu fixierenden visuellen Elements eine ungefähre Suchrichtung. Diese entspricht der Orientierungssrelation (siehe Abschnitt 3.1) zwischen den beiden beteiligten

Klassen und kann der Visualisierungs–Schema–Ontologie entnommen werden. Sie wird daher nicht explizit in der Lesevorschrift angegeben.

5.4 AOI–Ontologie

Eine Ontologie von AOIs dient dazu, die AOIs eines Stimulus mit Informationen über die AOIs anzureichern und durch die Einbindung der Ontologie von AOIs in die Ontologie–Hierarchie eine Verbindung zwischen den AOIs und den Schema–Klassen der Visualisierungs–Schema–Ontologie herzustellen. Es wird zunächst darauf eingegangen, warum eine Ontologie für AOIs verwendet wird und wie sie genau aussieht, daraufhin wird erläutert wie sich die Ontologie der AOIs in die Ontologie–Hierarchie einfügt und schließlich wird ausgeführt welche Informationen über die AOIs in der Ontologie abgelegt werden und welche Auswirkungen diese Informationen auf die grafische Darstellung der AOIs hat.

5.4.1 Warum eine Ontologie für AOIs?

Eine Ontologie besitzt nach Guarino et al. die höchste Ausdrucksstärke, um Wissen zu repräsentieren und ist visuell als Graph darstellbar [GOS09]. Ein Graph wiederum kann als Assoziationsnetz betrachtet werden, welches die symbolische Speicherung von deklarativem Wissen im menschlichen Gehirn modelliert. Somit wird diese Art der Darstellung von Wissen als Repräsentationsform für AOIs verwendet. Die AOIs eines Stimulus werden durch die Klassen einer Ontologie repräsentiert, der sogenannten *AOI–Ontologie*. Die Verwendung einer Ontologie bietet den großen Vorteil, dass die AOIs mit Informationen angereichert und auch Relationen zwischen den AOIs dargestellt werden können. Dies soll die Untersuchung eines Stimulus unterstützen. Abbildung 5.4 zeigt eine AOI–Ontologie am Beispiel eines Balkendiagramms. In der Abbildung sind die Klassen mit einer zusätzlichen Information über das Gewicht ausgestattet sowie die Orientierungsrelationen (siehe Abschnitt 3.1) zwischen den Klassen angegeben. Beide Merkmale werden von der Visualisierungs–Elemente–Ontologie vererbt. Die Gewichtsinformation wird zum Herausfiltern von AOIs aus einer Menge von AOIs benutzt.

5.4.2 Gewichtung von AOIs

Eine Lesevorschrift darf keine Klassen der Visualisierungs–Schema–Ontologie beinhalten, deren Instanzen während der visuellen Suche oder während des Cross–Checkings fokussiert werden (siehe Abschnitt 4.3). Da die Gewinnung einer Lesevorschrift auf Abfolgen von AOIs basiert, müssen diejenigen AOIs aus den AOI Abfolgen herausgefiltert werden, die während der visuellen Suche oder während des Cross–Checkings fokussiert werden. Die Filterung von AOIs, die während des Cross–Checkings fokussiert werden, wird in Abschnitt 5.5.4 erläutert. Um die Filterung von AOIs zu ermöglichen, die während der visuellen Suche fokussiert werden, können Visualisierungs–Elemente gewichtet werden. Die Möglichkeit zur Gewichtung wird von der Visualisierungs–Elemente–Ontologie vererbt. Dadurch kann eine AOI gewichtet werden. Ein positives Gewicht dürfen nur AOIs erhalten, die zur Lösung der Aufgabenstellung notwendig sind. AOIs, die nicht zur Lösung der Aufgabenstellung nötig sind, sind AOIs die während der visuellen Suche fokussiert werden. Diese AOIs erhalten daher ein negatives

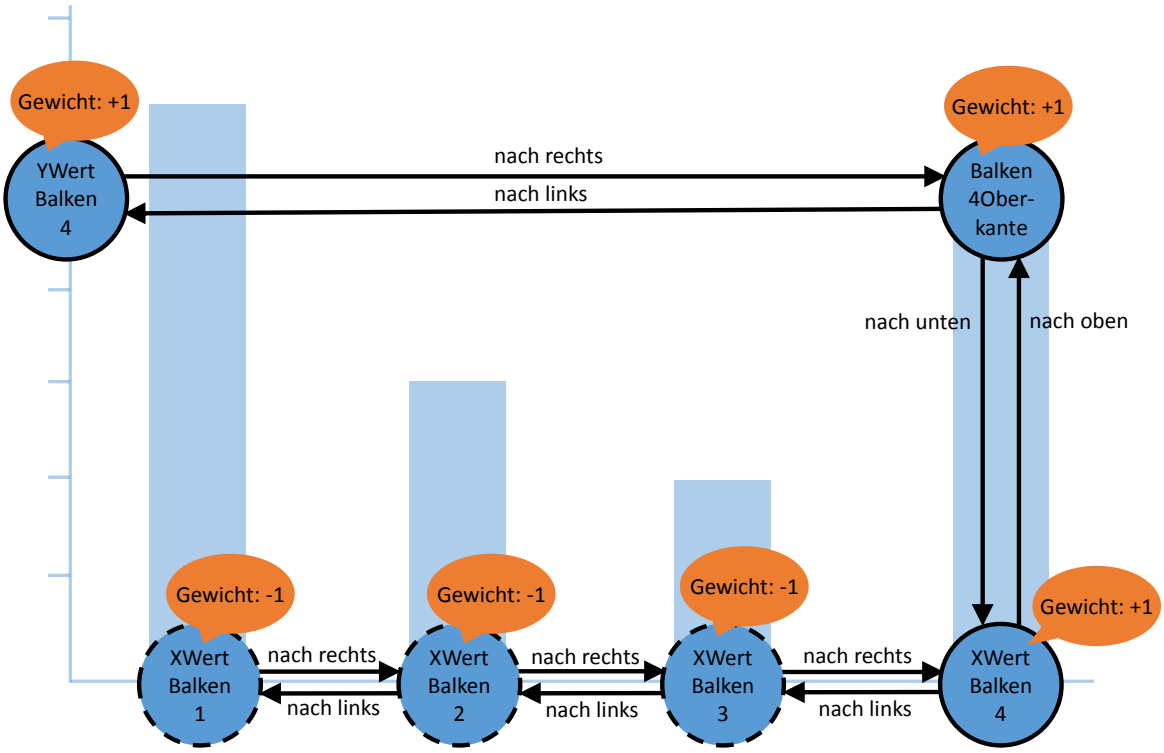

Abbildung 5.4: AOI–Ontologie für ein Balkendiagramm. Die AOI–Ontologie dient zur Darstellung von Relationen zwischen AOIs wie *nach rechts*, *nach links*, *nach oben*, *nach unten* und Informationen über AOIs wie das Gewicht. Die Gewichtsinformation wird zum Herausfiltern von AOIs aus einer Menge von AOIs verwendet.

Gewicht. Positiv wie negativ gewichtete AOIs können darüber hinaus durch den Betrag ihres Gewichts zusätzlich voneinander unterschieden werden. Auf diese Weise kann die besondere Relevanz oder Irrelevanz einer AOI hervorgehoben werden. In den Grafiken der vorliegenden Ausarbeitung und in der Implementierung des Prototyps werden positiv gewichtete AOIs mit durchgezogenem Rand, negativ gewichtete AOIs mit gestricheltem Rand dargestellt. Dasselbe gilt für AOI–Besuche. Dies soll am Beispiel der AOI–Ontologie auf dem Balkendiagramm in Abbildung 5.4 erklärt werden. Die Aufgabe für das Balkendiagramm in der Abbildung lautet „Geben Sie die Höhe des Balkens mit der Nummer vier an.“ Unbedingt notwendig zur Lösung der Aufgabe sind somit die AOIs „XWertBalken4“, „Balken4Oberkante“ und „YWertBalken4“. Sie erhalten daher ein positives Gewicht. Die restlichen drei AOIs erhalten ein negatives Gewicht.

Diese Darstellung der AOIs ermöglicht es zudem die kognitive Ergonomie einer Visualisierung mit dem Visualisierungskonzept dieses Konzepts zu untersuchen. Beinhaltet eine Abfolge von AOIs eine große Anzahl von negativ gewichteten AOI–Besuchen, so deutet dies darauf hin, dass der Proband viel Zeit mit der visuellen Suche verbracht hat und die zu suchenden visuellen Elemente schlecht auffindbar sind. Diese Untersuchung wird durch eine statistische Auswertung unterstützt. Sie enthält

Angaben über den Anteil negativ gewichteter AOI–Besuche an den gesamten AOI–Besuchen. Dies kann dazu genutzt werden die untersuchte Visualisierung zu verbessern.

5.5 Ablauf zur Gewinnung einer Lesevorschrift

Der Ablauf zur Ableitung einer Lesevorschrift beinhaltet mehrere Schritte. Er beginnt mit der Übergabe der Eingabedaten, darauf folgt die Erstellung oder Erweiterung der Visualisierungs–Elemente–Ontologie und der Visualisierungs–Schema–Ontologie. Es schließt sich die Verarbeitung auf AOI–Ebene an. Daraufhin folgt die Verarbeitung auf Schema–Klassen–Ebene. Abschließend wird die Ausgabe erzeugt. Die einzelnen Schritte sind sind in Abbildung 5.5 dargestellt. In den folgenden Abschnitten werden diese Schritte detailliert erläutert. Zuerst wird der Zusammenhang zwischen repräsentativen Reihenfolgen und Repräsentanten erläutert und anschließend werden die erforderlichen Eingabedaten für das Konzept zur Gewinnung einer Lesevorschrift beschrieben. Im Anschluss folgt ein Abschnitt über die Erstellung oder Erweiterung der beiden Ontologien in den Schichten über der AOI–Ontologie. Daran schließt sich eine Beschreibung der Verarbeitung auf AOI–Ebene an. Dabei werden für die N Stimuli der Eingabe die AOI Abfolgen der M Probanden analysiert. Der darauffolgende Abschnitt enthält die Ausführungen über die Verarbeitung auf Schema–Klassen–Ebene. Hierbei werden die N Abfolgen von Schema–Klassen untersucht. Den Abschluss bildet die Ausgabe der Lesevorschrift.

5.5.1 Repräsentative Reihenfolgen und Repräsentanten

Der Ablauf zur Gewinnung einer Lesevorschrift sieht an zwei Stellen die Erzeugung von repräsentativen Reihenfolgen von AOIs oder Schema–Klassen vor. Einmal bei der Verarbeitung auf AOI–Ebene und einmal bei der Verarbeitung auf Schema–Klassen–Ebene. Eine Lesevorschrift besteht aus einer repräsentativen Reihenfolge von Schema–Klassen. Weiterhin ist die Rede von einem Repräsentanten. Der Begriff des Repräsentanten und der Begriff einer repräsentativen Reihenfolge wird im Folgenden erklärt und die beiden Begriffe in Zusammenhang gebracht. Mit *Repräsentant* wird eine Abfolge von AOIs oder Schema–Klassen bezeichnet, die als Stellvertreter für eine Menge von AOI Abfolgen oder Schema–Klassen Abfolgen gilt. Eine *repräsentative* Abfolge kann aus einem Repräsentanten hervorgehen oder aber von Hand erstellt werden. Sie ist somit auch ein Stellvertreter für eine Menge von AOI oder Schema–Klassen Abfolgen. Zusätzlich gilt für eine repräsentative Abfolge, dass sie keine Elemente beinhaltet, die einer visuellen Suche oder einem Cross–Checking entsprechen (siehe Abschnitt 4.3).

5.5.2 Eingabedaten

Zu den Eingabedaten zählen die Stimuli und ihre zugehörigen Eye–Tracking–Daten. Bei den Stimuli handelt es sich ausschließlich um Instanzen des Visualisierungstyps für den die Lesevorschrift erzeugt werden soll. Die Instanz eines Visualisierungstyps bezeichnet eine konkrete Ausprägung eines Visualisierungstyps. Beispielsweise ist ein Balkendiagramm mit fünf Balken eine Instanz des

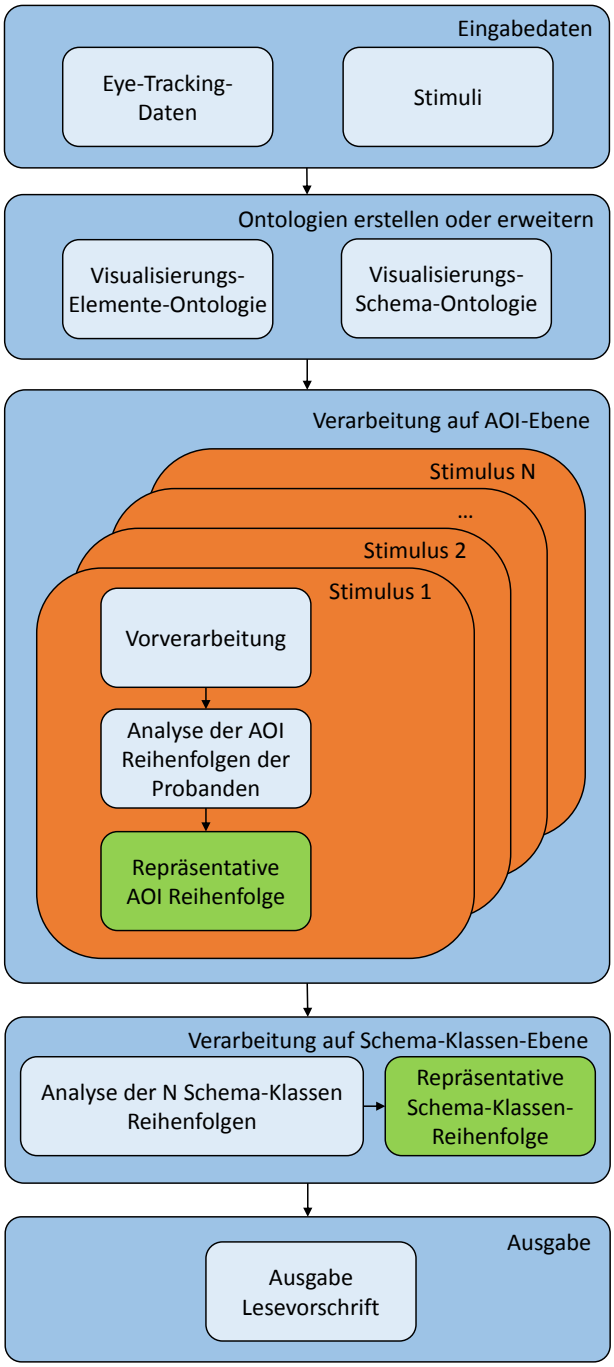

Abbildung 5.5: Ablauf zur Gewinnung einer Lesevorschrift. Auf die Übergabe der Eingabedaten folgt die Erstellung oder Erweiterung der Ontologien. Daran schließt sich die Verarbeitung auf AOI–Ebene an in deren Verlauf eine repräsentative AOI–Reihenfolge für jeden der Eingabestimuli erzeugt wird. Die repräsentativen AOI–Reihenfolgen werden auf der Ebene der Verarbeitung der Schema–Klassen zu einer repräsentativen Schema–Klassen–Reihenfolge weiterverarbeitet, die eine Lesevorschrift darstellen. Diese wird im letzten Schritt ausgegeben.

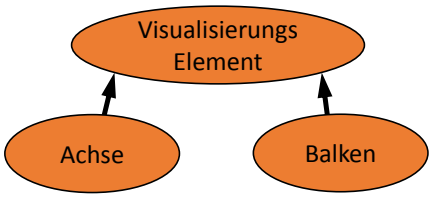

Abbildung 5.6: Beispiel für eine Visualisierungs–Elemente–Ontologie. Die Ontologie enthält die Visualisierungs–Elemente für Balkendiagramme.

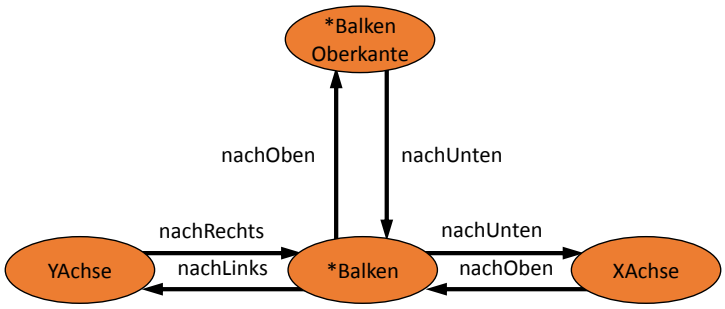

Abbildung 5.7: Beispiel für eine Visualisierungs–Schema–Ontologie. Die Ontologie repräsentiert den Visualisierungstyp Balkendiagramm. Instanzen der mit einem Stern (*) versehenen Klassen können beliebig oft in einem Balkendiagramm vorkommen.

Visualisierungstyps Balkendiagramm. Die Eye–Tracking–Daten beinhalten ebenfalls nur Aufnahmen von Augenbewegungen der Aufgabe für die die Lesevorschrift gelten soll.

5.5.3 Erstellung oder Erweiterung der beiden obersten Ontologien der Hierarchie

Die Visualisierungs–Elemente–Ontologie muss nur ein einziges Mal erstellt werden. Eine Änderung ist nur notwendig, wenn beispielsweise ein neuer Visualisierungstyp untersucht werden soll und dessen visuelle Elemente noch nicht in der Visualisierungs–Elemente–Ontologie enthalten sind. Dann müssen gegebenenfalls die neuen Elemente der Visualisierungs–Elemente–Ontologie hinzugefügt werden. Die Visualisierungs–Schema–Ontologie muss für jeden Visualisierungstyp erstellt werden. Ontologien können gespeichert werden, um nicht für jede Analyse von neuem erstellt werden zu müssen. Ein Beispiel für die Visualisierungs–Elemente–Ontologie, die die visuellen Elemente für Balkendiagramme enthält, ist in Abbildung 5.6 dargestellt. Die Visualisierungs–Schema–Ontologie für den Visualisierungstyp Balkendiagramm ist in Abbildung 5.7 enthalten. Die mit einem Stern (*) versehenen Klassen der Visualisierungs–Schema–Ontologie können beliebig oft in den Instanzen des Visualisierungstyps Balkendiagramm auftreten.

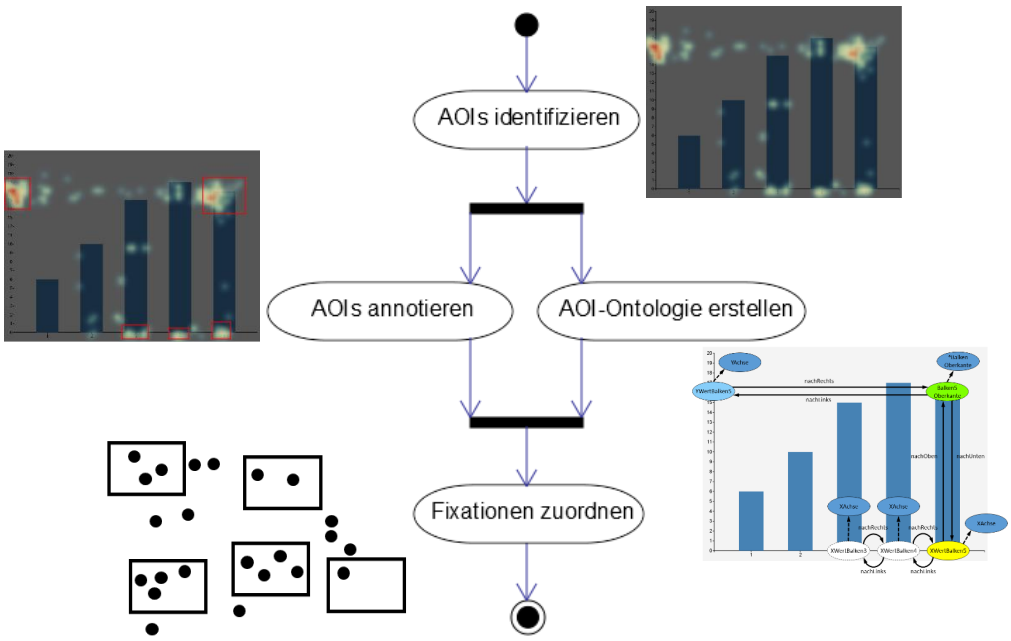

Abbildung 5.8: Schritte der Vorverarbeitung. Auf die Identifikation der AOIs mittels verschiedener automatischer und manueller Verfahren folgt die voneinander abhängige Annotation der AOIs und die Erstellung der AOI–Ontologie. Abschließend werden die Fixationen automatisch ihren entsprechenden AOIs zugeordnet.

5.5.4 Verarbeitung auf AOI–Ebene

Das Ziel dieses Schritts ist die Erzeugung einer repräsentativen AOI Reihenfolge für jeden der N Eingabe–Stimuli. Jeder der beiden im Folgenden beschriebenen Schritte wird pro Stimulus ausgeführt.

Vorverarbeitung

Die Vorverarbeitung dient dazu, die Eye–Tracking–Daten pro Stimulus für die Analyse vorzubereiten. Abbildung 5.8 zeigt die einzelnen Schritte der Vorverarbeitung.

Zuerst werden die AOIs identifiziert. Für die Identifikation stehen zwei automatische Verfahren zur Verfügung. Zusätzlich kann die Identifikation auch manuell durchgeführt werden. Das erste automatische Verfahren arbeitet ohne Eye–Tracking–Daten. Es analysiert den Stimulus mit Hilfe von Computer Vision Techniken und definiert AOIs basierend auf dieser Analyse. Das zweite automatische Verfahren arbeitet mit Eye–Tracking–Daten und ist Heatmap basiert. AOIs werden dort definiert, wo die Heatmap einen vom Benutzer festgelegten Schwellwert übersteigt. Bei der manuellen Identifikation entscheidet der Benutzer welche Bereiche des Stimulus er als AOI definieren möchte. Zur Unterstützung der Entscheidung kann auf die Anzeige von Gazepoints, Fixationen und Scanpfaden sowie einer Heatmap zurückgegriffen werden. Bei der manuellen Identifikation wird der Benutzer

durch Vorschläge unterstützt, die auf einem zum Stimulus passenden Assoziationsnetz basieren. Dies soll an dem im Folgenden beschriebenen Beispiel verdeutlicht werden. Handelt es sich bei dem Stimulus etwa um ein Balkendiagramm und der Benutzer sieht einen Balken, so ergeben sich im Gehirn des Benutzers bestimmte Assoziationen zum Begriff Balken. Die Assoziationen beinhalten Begriffe wie „Höhe des Balkens“, „X–Wert des Balkens“, „Höhe des Balkens im Vergleich zu anderen Balken“. Somit könnte ein assoziationsnetzbasierter Vorschlag etwa lauten „Der Bereich *X–Wert des Balkens mit der Nummer 3* befindet sich in der Nähe der zuletzt identifizierten AOI und weist eine hohe Blickintensität auf. Wollen Sie in diesem Bereich eine AOI identifizieren?“

Nachdem die AOIs identifiziert sind, folgt die Annotation der AOIs und die Erstellung der AOI–Ontologie. Beide Schritte hängen voneinander ab. Die AOI–Ontologie wird durch das Festlegen der AOIs erstellt. Dabei wird mit der Annotation einer AOI eine dieser AOI entsprechende AOI–Klasse zur AOI–Ontologie hinzugefügt. Es werden immer alle AOIs in die AOI–Ontologie mit aufgenommen. Wichtig für die weitere Verarbeitung ist jedoch die Zuweisung eines Gewichts an eine AOI. Ein positives Gewicht erhalten nur AOIs, die zur Lösung der Aufgabenstellung notwendig sind. Die Gewichtung kann einerseits dem Benutzer überlassen werden. Dieser wird bei seiner Entscheidung über die Gewichtung von einer Heatmap unterstützt und bezieht die Aufgabenstellung in die Entscheidung mit ein. Ein positives Gewicht ist für AOIs vorgesehen, die den Bereichen der Heatmap mit hoher Intensität entsprechen, da solche Bereiche mit hoher Wahrscheinlichkeit für die Lösung der Aufgabe relevant sind. Weiterhin ist ein positives Gewicht für AOIs vorgesehen, die für den Benutzer intuitiv zur Lösung der Aufgabe notwendig sind. Andererseits kann die Gewichtung über einen Schwellwert erfolgen. Übersteigt die Blickintensität diesen Schwellwert, so wird die AOI positiv gewichtet. Wird der Schwellwert unterschritten, so wird die AOI negativ gewichtet.

Nachdem alle AOIs identifiziert, annotiert und zur AOI–Ontologie hinzugefügt wurden, werden nun noch automatisch alle Fixationen des Stimulus ihren entsprechenden AOIs zugeordnet.

Analyse der AOI Reihenfolgen zur Erzeugung einer repräsentativen AOI Reihenfolge

In diesem Schritt wird eine für alle Probanden repräsentative Reihenfolge der AOI–Ontologie–Klassen für den Stimulus erzeugt. Dazu werden die Abfolgen der AOI–Besuche der Probanden analysiert. Der Ablauf ist in Abbildung 5.9 dargestellt. Zur Analyse stehen ein automatisches, ein halbautomatisches und ein manuelles Verfahren zur Verfügung.

Das automatische Verfahren erzeugt eine repräsentative Reihenfolge mit Hilfe eines Repräsentanten. Dabei kann aus verschiedenen Möglichkeiten gewählt werden, wie der Repräsentant berechnet werden soll (siehe Abschnitt 5.6). An die Berechnung des Repräsentanten schließt sich die Filterung an. Dabei werden alle AOIs entfernt, die nicht weiterverarbeitet werden dürfen, da sie einer visuellen Suche oder einem Cross–Checking entsprechen (siehe Abschnitt 4.3). Im Fall der visuellen Suche werden daher negativ gewichtete AOIs aus dem Repräsentanten entfernt. AOIs, die dem Cross–Checking entsprechen, werden über einen Wiederholungsfilter entfernt. Der Wiederholungsfilter kürzt beispielsweise die AOI Abfolge AOI1 AOI2 AOI1 AOI2 auf die Abfolge AOI1 AOI2.

Der Vorteil bei der Verwendung des automatischen Verfahrens liegt darin, dass, bei gleichzeitiger Anwendung des automatischen Verfahrens bei der Verarbeitung auf Schema–Klassen–Ebene, die Erzeugung einer Lesevorschrift vollständig automatisiert ablaufen kann. Der Nachteil dabei ist, dass

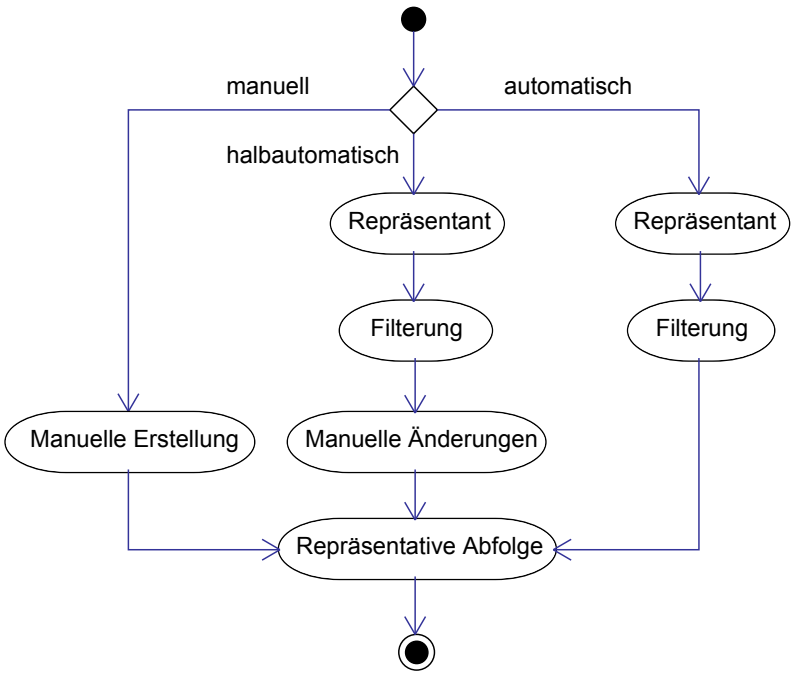

Abbildung 5.9: Ablauf der Analyse der AOI Reihenfolgen der Probanden. Für die Analyse kann zwischen einem manuellen, einem halbautomatischen und einem automatischen Verfahren gewählt werden. Im manuellen Verfahren wird eine repräsentative Abfolge von AOIs mit Hilfe des in Abschnitt 5.7 vorgestellten Visualisierungs– und Analysekonzepts von Hand erstellt. Beim halbautomatischen Verfahren wird die repräsentative AOI Abfolge auf der Basis eines Repräsentanten erzeugt, anschließend gefiltert und kann danach noch abgeändert werden. Beim automatischen Verfahren wird ein Repräsentant berechnet, gefiltert und schließlich als repräsentative Abfolge verwendet.

der Benutzer keine Kontrolle über die Zusammensetzung der jeweiligen repräsentativen Abfolgen hat.

Beim manuellen Verfahren werden anhand des in Abschnitt 5.7 vorgestellten Visualisierungs– und Analysekonzepts die Abfolgen der AOI–Besuche der Probanden untersucht und von Hand eine repräsentative Abfolge von AOIs erzeugt. Das manuelle Verfahren bietet den Vorteil, dass der Benutzer seine Erfahrung und sein Wissen in die Analyse einfließen lassen kann, um von Hand eine auf den Ergebnissen der Analyse basierende repräsentative Abfolge von AOIs zu erstellen. Als Nachteil kann der zeitliche Aufwand für die Analyse gesehen werden.

Der Ablauf des halbautomatischen Verfahrens entspricht dem des automatischen Verfahrens mit dem Unterschied, dass der Repräsentant nach Abschluss seiner Berechnung nach Belieben von Hand geändert werden kann. Der Benutzer kann also AOIs zum Repräsentanten hinzufügen, aus dem Repräsentanten entfernen oder die Reihenfolge der AOIs des Repräsentanten verändern. Somit wird es

dem Benutzer ermöglicht eine auf dem Repräsentanten basierende Reihenfolge von AOIs zu erstellen, die der Vorstellung des Benutzers von der intuitiv richtigen Reihenfolge entspricht. Der Vorteil bei der Verwendung des halbautomatischen Verfahrens liegt in der Kombination aus automatisierter Erzeugung eines aussagekräftigen Repräsentanten und der sich daran anschließenden Veränderung des Repräsentanten basierend auf dem Wissen und der Erfahrung des Benutzers. Als Nachteil kann auch hier der zeitliche Aufwand der manuellen Analyse gesehen werden.

5.5.5 Verarbeitung auf Schema–Klassen–Ebene

Da im Anschluss auf der Ebene der Schema–Klassen weitergearbeitet wird, ändert sich der Untersuchungsgegenstand der Analyse. Dieser umfasste bisher die AOIs und wechselt nun auf die Klassen der Visualisierungs–Schema–Ontologie von denen die AOIs abgeleitet sind. Die Reihenfolge der Schema–Klassen entspricht dabei der der AOI–Besuche des jeweiligen Stimulus.

Das Ziel dieses Schritts ist die Erzeugung einer repräsentativen Schema–Klassen–Reihenfolge. Dazu werden die N Schema–Klassen Reihenfolgen der N Stimuli analysiert, die aus den N repräsentativen AOI Reihenfolgen der Stimuli hervorgehen. Die Analyse erfolgt analog zur Analyse der AOI Reihenfolgen. Es stehen ebenso ein automatisches, ein halbautomatisches und ein manuelles Verfahren zur Erzeugung der repräsentativen Abfolge von Schema–Klassen zur Verfügung. Das automatische und das halbautomatische Verfahren arbeiten mit der Berechnung eines Repräsentanten. Der Repräsentant kann beim halbautomatischen Verfahren nach Abschluss seiner Berechnung von Hand geändert werden. Beim manuellen Verfahren wird die repräsentative Abfolge von Schema–Klassen von Hand durch den Benutzer zusammengestellt. Dabei werden die Änderungen des Repräsentanten beim halbautomatischen Verfahren und die Erstellung der repräsentativen Abfolge beim manuellen Verfahren durch das in Abschnitt 5.7 beschriebene Visualisierungs– und Analysekonzept unterstützt. Der Unterschied zur Verarbeitung auf AOI–Ebene liegt darin, dass beim automatischen und beim halbautomatischen Verfahren keine Filterung notwendig ist. Dies begründet sich dadurch, dass die Abfolgen der Schema–Klassen aus den repräsentativen AOI Abfolgen hervorgehen und jegliche unerlaubte AOIs bereits in der Verarbeitung auf AOI–Ebene aus den repräsentativen AOI Abfolgen herausgefiltert wurden.

5.5.6 Ausgabe der Lesevorschrift

Abschließend wird die Lesevorschrift ausgegeben. Diese enthält die im letzten Schritt erzeugte und für diesen Visualisierungstyp repräsentative Reihenfolge der Schema–Klassen. Die Lesevorschrift kann anschließend zu einem Satz von Produktionsregeln weiterverarbeitet werden, der dann als Eingabe für die Augenbewegungs–Simulation genutzt werden kann, wie in Abbildung 5.1 dargestellt.

5.6 Berechnung eines Repräsentanten

Die Berechnung eines Repräsentanten kommt bei den automatischen und halbautomatischen Verfahren zur Erzeugung einer repräsentativen Abfolge von AOIs oder Schema–Klassen zum Einsatz. Der

Begriff des Repräsentanten und der der repräsentativen Abfolge sowie der Zusammenhang zwischen beiden Begriffen wird in Abschnitt 5.5.1 erklärt. Im Folgenden wird die grundlegende Vorgehensweise zur Berechnung eines Repräsentanten erläutert sowie drei verschiedene Berechnungsverfahren beschrieben. Bei jedem Verfahren kann ausgewählt werden, ob die Berechnung mit oder ohne negativ gewichtete Elemente erfolgen soll und ob Elemente, die einem Cross–Checking entsprechen in die Berechnung miteinbezogen werden sollen oder nicht.

5.6.1 Grundlegende Vorgehensweise

Da bei den folgenden Methoden Abfolgen von AOIs oder Schema–Klassen miteinander verglichen werden, ist es wichtig, dass sich die zu vergleichenden Abfolgen grundsätzlich ähneln. Aufgrund der Tatsache, dass es sich bei den zugrundeliegenden Daten für den Vergleich um die zeitunabhängigen Reihenfolgen von AOI– oder Schema–Klassen–Besuchen handelt und diese sich nur auf eine einzige Aufgabe beziehen, wird eine grundsätzliche Ähnlichkeit vorausgesetzt.

Die Berechnung von Repräsentanten basiert auf dem Vergleich von Zeichenketten. Dazu wird den AOIs oder Schema–Klassen ein eindeutiger Buchstabe zugewiesen. Dies ergibt eine Zeichenkette für jede Abfolge von AOI– und Schema–Klassen–Besuchen. Um die Ähnlichkeit zweier Zeichenketten zu vergleichen, wird deren Levenshtein–Distanz berechnet. Zwei Zeichenketten sind sich umso ähnlicher, je niedriger ihre Levenshtein–Distanz ist. Die Berechnung der Levenshtein–Distanz für zwei AOI Abfolgen ist in Abbildung 5.10 dargestellt. Die Einfüge– und Löschkosten betragen dabei jeweils zwei, die Ersetzungskosten eins. In Bereich (1) der Abbildung sind die beiden AOI Abfolgen dargestellt. Bereich (2) enthält die Matrix, die während der Durchführung des Algorithmus zur Berechnung der Levenshtein–Distanz aufgebaut wird. Dabei steht das Ergebnis der Berechnung, also die Levenshtein–Distanz im rechten unteren Feld der Matrix. Sie berechnet sich für die beiden AOI Abfolgen ABC und DBC zu 1.

5.6.2 Beschreibung von drei Verfahren zur Repräsentanten–Berechnung

Zur Berechnung eines Repräsentanten stehen drei verschiedene Verfahren zur Auswahl. Dazu zählt die Auswahl einer Abfolge aus einer Menge von Abfolgen und Bestimmung dieser Abfolge zum Repräsentanten. Weiterhin kann ein Repräsentant mittels eines Clustering–Algorithmus bestimmt werden sowie ein idealer Repräsentant anhand der Lösung des Closest Substring Problems erstellt werden. Alle drei Verfahren werden in den folgenden Abschnitten näher vorgestellt.

Auswahl einer Abfolge aus einer Menge von Abfolgen

Für die Auswahl einer Abfolge aus einer Menge von Abfolgen und Bestimmung dieser Abfolge zum Repräsentanten, wird folgendermaßen vorgegangen. Es wird die Levenshtein–Distanz einer jeden

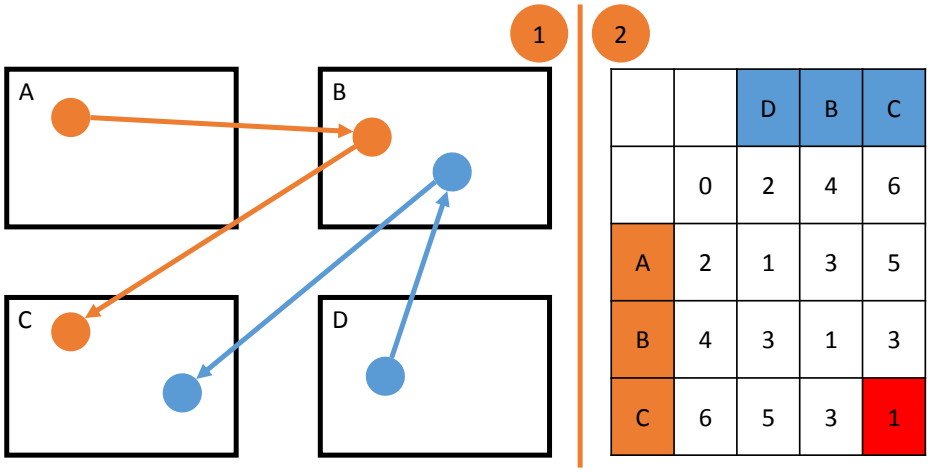

Abbildung 5.10: Ähnlichkeits–Berechnung mit Hilfe der Levenshtein–Distanz. (1) zeigt zwei Abfolgen von AOI–Besuchen, die sich über vier AOIs verteilen. Den AOIs werden eindeutige Buchstaben A, B, C, D zugewiesen. Die Abfolge der AOI–Besuche kann nun durch eine Zeichenkette der zugehörigen AOIs repräsentiert werden. Für zwei AOI Abfolgen kann somit ihrer Levenshtein–Distanz berechnet werden. (2) zeigt die Matrix, die während der Berechnung der Levenshtein–Distanz der AOI–Abfolgen in (1) aufgebaut wird. Das Ergebnis der Berechnung steht im rot markierten Feld der Matrix.

Abfolge mit jeder anderen Abfolge der Menge berechnet und pro Abfolge aufsummiert. Dies entspricht folgender Berechnungsvorschrift:

$$R_i = \sum_{j}^{k} L(X_i, Y_j) \text{ mit } i \neq j \text{ und } 0 \leq i \leq l-1 \text{ und } 0 \leq j \leq k = l-2$$

Dabei bezeichnet l die Anzahl der Abfolgen in der Menge, R_i die aufsummierte Levenshtein–Distanz für die aktuell betrachtete Abfolge X_i und Y_j die jeweils zu vergleichende Abfolge. Die Funktion L berechnet die Levenshtein–Distanz. Als Repräsentant wird die Abfolge X_i mit minimalem Wert für R_i gewählt. Diese Art der Berechnung ist in Abbildung 5.11 dargestellt. In der Abbildung hat die Abfolge ABD die aufsummierte Levenshtein–Distanz von 5, ist somit am ähnlichsten zu den beiden anderen Abfolgen und wird daher als Repräsentant gewählt.

Idealer Repräsentant

Zur Berechnung eines idealen Repräsentanten wird eine ideale Abfolge aus den vorhandenen Abfolgen berechnet. Dazu wird aus allen vorhandenen Abfolgen eine neue erzeugt, deren Levenshtein–Distanz zu allen vorhandenen Abfolgen den geringsten Wert hat. Dies ist nach Kao ein NP–vollständiges Problem mit der Bezeichnung Closest Substring [Kao07]. Der Vorteil dieser Methode liegt darin, das die berechnete Abfolge die minimal möglichste Levenshstein–Distanz und somit die größtmögliche

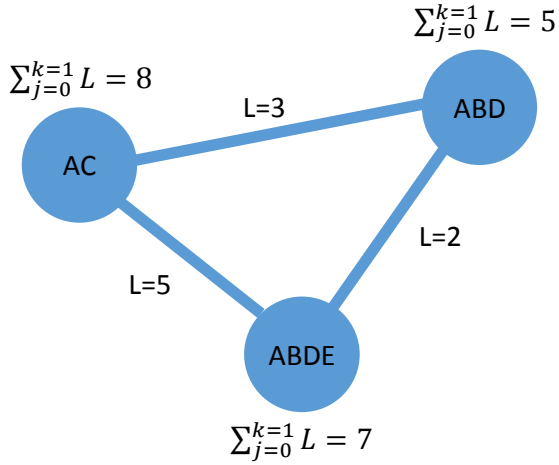

Abbildung 5.11: Bestimmung eines Repräsentanten durch Auswahl einer Abfolge aus einer Menge von Abfolgen. Die ausgewählte Abfolge wird zum Repräsentanten bestimmt. Dies geschieht durch die Berechnung der aufsummierten Levenshtein–Distanz zu allen anderen Abfolgen der Menge. Zum Repräsentanten wird diejenige Abfolge mit der geringsten Summe bestimmt.

Ähnlichkeit zu allen anderen Abfolgen hat. Der Nachteil liegt darin, dass es sich um eine künstlich erzeugte Abfolge handelt, die keiner realen Abfolge entspricht.

Berechnung des Repräsentanten durch Clustering

Bei der Berechnung eines Repräsentanten durch einen Clustering–Algorithmus bildet man eine Rangliste der Häufigkeit von ähnlichen Abfolgen. Dazu werden ähnliche Abfolgen mit Hilfe eines Clustering–Alogithmus in Gruppen zusammengefasst. So entsteht eine Menge von Gruppen, die hinsichtlich der Anzahl ihrer enthaltenen Abfolgen absteigend in der Rangliste platziert werden. Für die Wahl des Repräsentanten wird die Gruppe an der ersten Stelle der Rangliste herangezogen und ihr Repräsentant mit einem der beiden vorgenannten Verfahren berechnet.

5.7 Visualisierungs– und Analysekonzept

Der Ablauf zur Gewinnung einer Lesevorschrift sieht zwei Verarbeitungsschritte vor, die Verarbeitung auf AOI–Ebene und die Verarbeitung auf Schema–Klassen–Ebene. Bei der Verarbeitung auf AOI–Ebene wird eine repräsentative Abfolge von AOIs erzeugt, bei der Verarbeitung auf Schema–Klassen–Ebene eine repräsentative Abfolge von Schema–Klassen. Die repräsentativen Abfolgen sollen mit Hilfe verschiedener Visualisierungs– und Analysetechniken erzeugt werden. Auf der Ebene der Verarbeitung der AOIs werden für N Stimuli die Abfolgen der AOI–Besuche von M Probanden untersucht, um N

repräsentative AOI Abfolgen zu erzeugen. Bei der Verarbeitung auf Schema–Klassen–Ebene werden N Schema–Klassen Abfolgen analysiert, um eine für den Visualisierungstyp repräsentative Abfolge von Schema–Klassen zu bestimmen. Dies legt es nahe, die Visualisierungs– und Analysetechniken, die bei der Verarbeitung auf AOI–Ebene für jeden einzelnen der N Stimuli zur Untersuchung der M AOI Abfolgen zum Einsatz kommen, auch bei der Verarbeitung auf Schema–Klassen–Ebene zur Analyse der N Abfolgen von Schema–Klassen zu verwenden. Da sich die Vorgehensweisen der beiden Verarbeitungsebenen voneinander unterscheiden, beherrschen die Visualisierungs– und Analysetechniken zwei verschiedene Modi, sofern sie in beiden Verarbeitungsschritten zum Einsatz kommen und nicht nur in einem. Die verschiedenen Visualisierungs– und Analysetechniken werden für jeden der beiden Verarbeitungsschritte im zum jeweiligen Verarbeitungsschritt passenden Modus zu einem Visualisierungs– und Analysekonzept kombiniert. Das Visualisierungs– und Analysekonzept beherrscht dabei zwei Modi. Der erste Modus umfasst die Kombination von Visualisierungen im Modus zur Verarbeitung auf AOI–Ebene und dient dazu den Schritt zur Verarbeitung auf AOI–Ebene durchzuführen. Der zweite Modus beinhaltet die Kombination von Visualisierungen im Modus zur Verarbeitung auf Schema–Klassen–Ebene und ist dazu vorgesehen den Schritt zur Verarbeitung auf Schema–Klassen–Ebene auszuführen. Die verschiedenen Visualisierungen sind dabei im jeweiligen Modus per Brushing & Linking miteinander verbunden. Alle Visualisierungen werden als Plugin umgesetzt. Dadurch können auf einfache Art und Weise neue Visualisierungen zum Konzept hinzugefügt werden.

Dieser Abschnitt beschreibt zunächst die Techniken, die an verschiedenen Stellen von den Visualisierungs– und Analysetechniken verwendet werden. Anschließend werden die Visualisierungs– und Analysetechniken im Detail erläutert und näher auf die Besonderheiten der beiden Modi der jeweiligen Technik eingegangen. Abschließend wird jeder der beiden Modi des Visualisierungs– und Analysekonzepts erklärt.

5.7.1 Mehrfach verwendete Techniken

Die in diesem Abschnitt vorgestellten Techniken finden an verschiedenen Stellen in den Visualisierungs– und Analysetechniken ihre Anwendung. Sie werden daher hier an zentraler Stelle beschrieben und beim Einsatz in einer der Visualisierungs– oder Analysetechniken wird auf diesen Abschnitt verwiesen. Zu den Techniken gehört die Darstellungsweise von AOIs und AOI–Besuchen sowie von Schema–Klassen und Schema–Klassen–Besuchen. Des weiteren werden die Informationen beschrieben, die für AOIs, AOI–Besuche, Schema–Klassen und Schema–Klassen–Besuche angezeigt werden können. Weiterhin wird die Möglichkeit zur Einblendung des Stimulus erläutert und schließlich werden die zur Verfügung stehenden Layouttechniken für Graphen erklärt.

Darstellung von AOIs und AOI–Besuchen

Für Visualisierungen im Modus zur Verarbeitung auf AOI–Ebene gibt es eine einheitliche Darstellung von AOIs und AOI–Besuchen. Kommt diese Darstellungsweise zum Einsatz, so wird dies im Abschnitt über die jeweilige Visualisierung erwähnt. AOIs und AOI–Besuche werden als Kreis dargestellt. Bei AOI–Besuchen wird die Aufenthaltsdauer in der AOI auf den Durchmesser des Kreises abgebildet. Um zu vermeiden, dass ein einzelner übermäßig langer AOI–Besuch zur Folge hat, dass alle anderen

Besuche sehr klein dargestellt werden, kann ein Schwellwert festgelegt werden. Dieser bestimmt ab welcher Dauer Besuche in einer anderen Form dargestellt werden, sollte die Dauer den Schwellwert übersteigen. Statt der kreisförmigen Darstellung wird der AOI–Besuch dann sternförmig dargestellt.

Informationen zu AOIs und AOI–Besuchen

Über AOIs und AOI–Besuche können bestimmte Informationen angezeigt werden. Die Informationen werden durch die Auswahl einer AOI oder eines AOI–Besuchs angezeigt. Bei der Auswahl einer AOI werden die Informationen der AOI–Klasse aus der AOI–Ontologie angezeigt. Dazu gehören das Gewicht der AOI und von welchen Vaterklassen in der Ontologie–Hierarchie die Klasse abgeleitet ist. Bei AOI–Besuchen wird zusätzlich zu den Informationen der zugehörigen AOI noch die Aufenthaltsdauer in der AOI in Millisekunden angegeben.

Darstellung von Schema–Klassen und Schema–Klassen–Besuchen

Für Visualisierungen im Modus zur Verarbeitung auf Schema–Klassen–Ebene gibt es eine einheitliche Darstellung der Schema–Klassen und der Schema–Klassen–Besuche. Sofern diese Darstellungsweise zum Einsatz kommt, wird dies im Abschnitt über die jeweilige Visualisierung angemerkt. Die Darstellung von Schema–Klassen und Schema–Klassen–Besuchen erfolgt als Kreis. Bei Schema–Klassen–Besuchen wird die durchschnittliche Aufenthaltsdauer in den von dieser Schema–Klasse abgeleiteten AOIs auf den Durchmesser des Kreises abgebildet. Die Anwendung eines Schwellwerts erfolgt analog zur Anwendung des Schwellwerts bei der Darstellung von AOIs.

Informationen zu Schema–Klassen und Schema–Klassen–Besuchen

Es können bestimmte Informationen über Schema–Klassen und Schema–Klassen–Besuche angezeigt werden. Durch die Auswahl einer Schema–Klasse oder eines Schema–Klassen–Besuchs werden die Informationen angezeigt. Durch die Auswahl einer Schema–Klasse werden die Informationen der Schema–Klasse aus der Visualisierungs–Schema–Ontologie angezeigt. Dabei handelt es sich darum von welchen Vaterklassen in der Ontologie–Hierarchie die Klasse abgeleitet ist. Bei Schema–Klassen–Besuchen wird zusätzlich zu den Informationen der zugehörigen Schema–Klasse noch die durchschnittliche Aufenthaltsdauer in den von dieser Schema–Klasse abgeleiteten AOIs in Millisekunden angegeben.

Stimulus Einblendung

Je nach Visualisierung besteht die Möglichkeit, den zugrundeliegenden Stimulus einzublenden. Bei welcher Visualisierung dies jeweils möglich ist, wird im Abschnitt der jeweiligen Visualisierung erläutert. Die Transparenz des eingeblendeten Stimulus kann vom Benutzer festgelegt werden. Ebenso kann die Transparenz der Elemente der jeweiligen Visualisierung eingestellt werden. Weiterhin kann die Reihenfolge der Einblendungsebenen festgelegt werden. So kann zuerst der Stimulus und in der Ebene darüber die Visualisierung angezeigt werden oder umgekehrt. Dadurch kann die jeweilige

Visualisierung durch den Benutzer so gestaltet werden, wie sie für ihn in Kombination mit der Einblendung des Stimulus die beste Lesbarkeit bietet.

Layouts für Graph–Visualisierungen

Für die Visualisierungen, die die Darstellung eines Graphen beinhalten, stehen verschiedene Techniken für das Layout des Graphen zur Verfügung. Diese werden im Folgenden näher beschrieben.

- Stimulusbasiertes Layout: Diese Option steht nur zur Verfügung, sofern der dargestellte Graph einen Bezug zu einem Stimulus hat. Die Position der Knoten entspricht dabei der Position des dem Knoten entsprechenden Elements auf dem Stimulus. Wird das Aussehen des Graphs geändert, so kann das ursprüngliche stimulusbasierte Layout auf Wunsch des Benutzers wiederhergestellt werden.
- Kräftebasiertes Layout: Für das Layout des Graphen steht ein kräftebasierter Layout–Algorithmus zur Verfügung. Die Ziele sind hierbei eine möglichst übersichtliche Verteilung der Knoten und möglichst wenige Kantenüberschneidungen.
- Manuelles Layout: Es besteht die Möglichkeit sowohl die Position der Knoten als auch den Verlauf der Kanten von Hand festzulegen. Knotenpositionen und Kantenverläufe können gespeichert und zu einem späteren Zeitpunkt wiederhergestellt werden.

5.7.2 Visualisierungs– und Analysetechniken

Die im Folgenden beschriebenen Visualisierungs– und Analysetechniken werden in einem oder in beiden Modi verwendet. Nachfolgend werden die Ontologie Visualisierung, die Visualisierungstechnik des Space–Time–Cube, die Stimulus–Visualisierung, die Pattern Search Lanes, die Suche in den Pattern Search Lanes, der Graph und die Ergebniszusammenstellung beschrieben. Schließlich wird der Brushing & Linking Mechanismus erläutert, der die Visualisierungen miteinander verbindet.

Ontologie Visualisierung

Die Ontologie–Visualisierung dient dazu, die Visualisierungs–Elemente–Ontologie, die Visualisierungs–Schema–Ontologie und die AOI–Ontologie zu visualisieren. Die Darstellung der jeweiligen Ontologie erfolgt als Graph. Für das Layout des Graphen kommen die in Abschnitt 5.7.1 beschriebenen Layouttechniken für Graphen zum Einsatz. Bei der Darstellung des Ontologie Graphen kann ausgewählt werden, ob die Relationen zwischen den Ontologie Klassen ein– oder ausgeblendet werden sollen. Informationen zu den Ontologie Klassen der jeweiligen Ontologie können über die in Abschnitt 5.7.1 beschriebene Technik angezeigt werden. Die Ontologie–Visualisierung ist gleichzeitig ein graphischer Editor für die jeweils enthaltene Ontologie. Dies hat den Vorteil, dass Tätigkeiten wie das Erstellen oder Erweitern von Ontologien ohne den Einsatz von externer Software erledigt werden können.

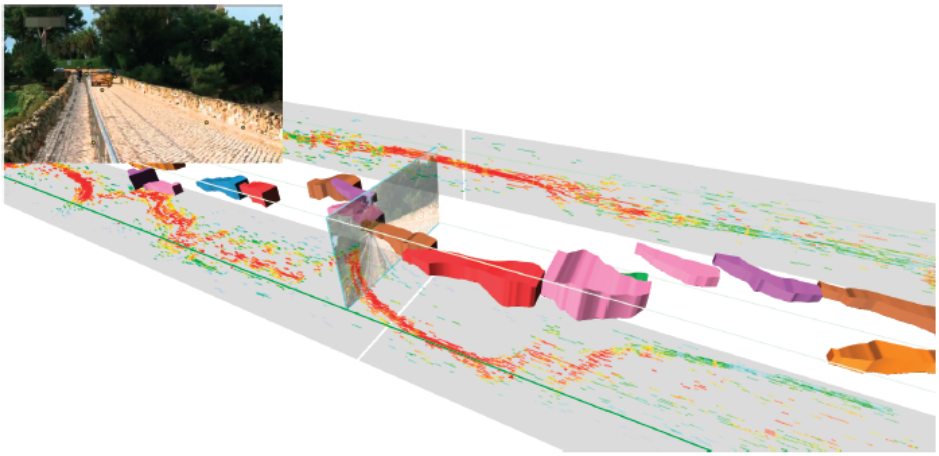

Abbildung 5.12: Die Visualisierungstechnik des Space–Time–Cubes. Diese Visualisierungstechnik zeigt den Verlauf der Scanpfade über die Zeit innerhalb eines drehbaren Würfels an [KW13].

Space–Time–Cube

Die Visualisierungstechnik des Space–Time–Cubes dient dazu, einen Gesamtüberblick über die Scanpfade eines Stimulus zu erhalten. Abbildung 5.12 zeigt die Visualisierungstechnik des Space–Time–Cubes. Diese Visualisierung–Technik verbindet nach Aigner et al. die Visualisierung von räumlichen und zeitlichen Daten [AMST11]. Dabei werden zwei räumliche Dimensionen auf zwei Achsen eines dreidimensionalen Würfels abgebildet. Auf die dritte Achse wird die Zeit abgebildet. Im dreidimensionalen Raum innerhalb des Würfels kann dadurch jeder Punkt als Punkt mit dreidimensionalen Koordinaten betrachtet werden. Im Bezug auf Eye–Tracking–Daten kann somit die räumliche Position einer Fixation sowie deren Zeitpunkt im Innern des Würfels dargestellt werden. Zeigt man mehrere Fixationen an und verbindet diese durch Linien miteinander so entsteht ein dreidimensionaler Scanpfad. Es können auch mehrere Scanpfade von verschiedenen Probanden angezeigt und unterschiedlich eingefärbt werden. Um die Scanpfade aus verschiedenen Perspektiven betrachten zu können, ist der Würfel drehbar. In der von den beiden räumlichen Achsen aufgespannten Ebene kann der Stimulus angezeigt werden, wie in Abschnitt 5.7.1 beschrieben. Der Stimulus kann dabei entlang der Zeitachse des Würfels verschoben werden, um jeden von den Scanpfaden betrachteten Punkt des Stimulus zu jedem Zeitpunkt untersuchen zu können. Die AOIs und die AOI–Ontologie können eingeblendet und auf dieselbe Weise wie der Stimulus verschoben sowie ihre Transparenz geändert werden. Somit kann festgestellt werden, zu welchem Zeitpunkt welche AOI von welchem Probanden fokussiert wurde. Zur Einblendung der AOI–Ontologie kommt die Abschnitt 5.7.2 beschriebene Ontologie–Visualisierung zum Einsatz. Informationen über die AOIs können angezeigt werden wie in Abschnitt 5.7.1 beschrieben.

Stimulus–Visualisierung

Die Stimulus–Visualisierung dient dazu AOIs zu identifizieren und zu annotieren und um parallel zur Analyse im WAS–Raum auch im WO–Raum Untersuchungen anstellen zu können. Weiterhin soll durch die Anzeige von WO–Raum Metriken die Identifikation und Annotation von AOIs unterstützt werden. Dazu beherrscht die Stimulus–Visualisierung verschiedene Ansichten. Es können Gazepoints, Fixationen, Scanpfade und eine Heatmap angezeigt werden. Der Stimulus kann eingeblendet werden, wie in Abschnitt 5.7.1 beschrieben. Zusätzlich integriert die Stimulus–Visualisierung die in Abschnitt 5.7.2 beschriebene Ontologie Visualisierung. Sie wird dazu genutzt, um die AOI–Ontologie auf dem Stimulus anzuzeigen. Es kann jederzeit eine neue AOI hinzugefügt, eine vorhandene bearbeitet oder entfernt werden. Die AOI–Ontologie wird dann automatisch angepasst. Dies hat den Vorteil, dass keine externe Software zur Annotation der AOIs verwendet werden muss. Informationen zu AOIs können über die in Abschnitt 5.7.1 beschriebene Technik angezeigt werden. Durch die parallele Nutzung der Stimulus–Visualisierung zu den im weiteren Verlauf dieser Ausarbeitung vorgestellten Visualisierungen im WAS–Raum kann beispielsweise eine Abfolge von AOI–Besuchen mit dem Verlauf des zugehörigen Scanpfads auf dem Stimulus verglichen werden.

Pattern Search Lanes

Die Pattern Search Lanes sind das zentrale Instrument für die manuelle Untersuchung von AOI– und Schema–Klassen–Abfolgen. Die Visualisierung ist angelehnt an eine Scarf Plot Darstellung (siehe Abschnitt 3.2). Die Pattern Search Lanes sind aus mehreren vertikalen und zueinander parallel verlaufenden Bahnen aufgebaut. Jede Bahn repräsentiert je nach Verarbeitungsmodus einen Probanden oder einen Stimulus. Die gesamte Ansicht ist scrollbar, damit auch sehr lange Abfolgen analysiert werden können. Im Modus zur Verarbeitung auf AOI–Ebene enthält eine Bahn die Abfolge der AOI–Besuche eines Probanden in ihrer zeitlichen Reihenfolge. Beim Modus zur Verarbeitung auf Schema–Klassen–Ebene enthält eine Bahn die Abfolge der Schema–Klassen–Besuche eines Stimulus. Für die Anzeige von Informationen zu AOIs, AOI–Besuchen, Schema–Klassen und Schema–Klassen–Besuchen kommen die in Abschnitt 5.7.1 beschriebenen Techniken zum Einsatz. Negativ gewichtete AOI–Besuche werden, wie in Abschnitt 5.4.2 beschrieben, mit gestricheltem Rand dargestellt.

Die normale Ansicht der Pattern Search Lanes ist in Bereich (1) der Abbildung 5.13 dargestellt. Im Modus zur Verarbeitung auf AOI–Ebene wird dabei die Aufenthaltsdauer in den AOIs auf den Durchmesser der Kreise abgebildet wie in Abschnitt 5.7.1 beschrieben. Die Dauer der Transition zwischen zwei AOI–Besuchen wird auf den Abstand zwischen den AOI–Besuchen abgebildet. Im Modus zur Verarbeitung auf Schema–Klassen–Ebene wird die durchschnittliche Aufenthaltsdauer in den von dieser Schema–Klasse abgeleiteten AOIs auf den Durchmesser des Kreises abgebildet wie in Abschnitt 5.7.1 erklärt. Die durchschnittliche Transitionsdauer zwischen den von den Schema–Klassen abgeleiteten AOIs wird auf den Abstand zwischen den Schema–Klassen–Besuchen abgebildet. Der Name des Probanden beziehungsweise des Stimulus wird im Bereich (5) der Abbildung dargestellt. Reicht der Platz zur Darstellung nicht aus, so wird der Name um 90° gedreht.

Im Folgenden wird erläutert, welche Optionen zur Beeinflussung der Darstellung der Pattern Search Lanes bestehen. Um Besuche besser auswählen zu können und da es ohnehin nur um den zeitunabhängigen Vergleich ihrer Reihenfolge geht, besteht die Möglichkeit sowohl Durchmesser als auch

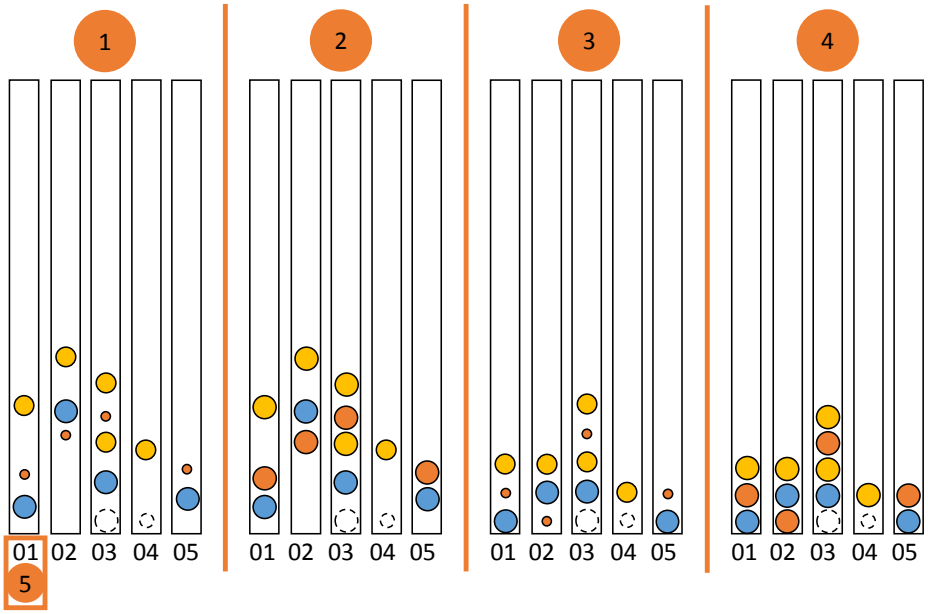

Abbildung 5.13: Verschiedenartige Darstellung von Abfolgen in den Pattern Search Lanes. (1) zeigt die normale Ansicht, mit auf den Kreisdurchmesser des Besuchs abgebildeten Aufenthaltsdauern und auf Abstände abgebildeten Transitionsdauern. Bei (2) erhalten alle Kreise der Besuche einen einheitlichen Durchmesser. In (3) haben alle Kreise der Besuche einen einheitlichen Abstand voneinander und in (4) sind sowohl Abstände als auch Durchmesser der Kreise der Besuche einheitlich. In (5) wird der Name des zur jeweiligen Bahn gehörigen Probanden beziehungsweise Stimulus angezeigt.

Abstände der Besuche zu vereinheitlichen. Der Bereich (2) in Abbildung 5.13 zeigt die Ansicht der Pattern Search Lanes, bei der die Kreise aller AOI beziehungsweise Schema–Klassen–Besuche einen einheitlichen Durchmesser haben. In Bereich (3) der Abbildung sind die Durchmesser nicht mehr einheitlich, aber die Abstände. Schließlich sind in Bereich (4) der Abbildung sowohl die Durchmesser als auch die Abstände einheitlich groß.

Die Pattern Search Lanes sind dazu entwickelt worden, dem Benutzer die manuelle Zusammenstellung einer AOI oder Schema–Klassen Abfolge zu ermöglichen. Zur Unterstützung der menschlichen Fähigkeiten zur Mustererkennung insbesondere auch bei langen und auch vielen Abfolgen stehen verschiedene Suchverfahren zur Verfügung. Der Benutzer kann AOI– oder Schema–Klassen–Besuche in einer Bahn auswählen und deren Vorkommen in den anderen Bahnen suchen und hervorheben lassen. Dies ist in Abbildung 5.14 dargestellt. Bereich (1) der Abbildung zeigt eine markierte Abfolge von Besuchen. Die zu suchenden Besuche werden schwarz umrandet dargestellt. Bereich (2) zeigt das Ergebnis einer Suche. Die gefundenen Besuche werden rot umrandet angezeigt. Nachdem eine Abfolge gesucht wurde steht eine statistische Auswertung zur Verfügung, die Auskunft darüber gibt, wie häufig die gesuchte Abfolge in den anderen Bahnen vorkommt.

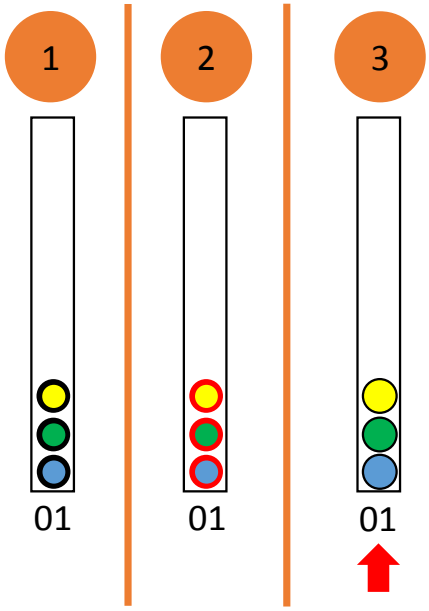

Abbildung 5.14: Hervorhebungen in den Pattern Search Lanes. (1) zeigt mit der Maus markierte Besuche, die schwarz umrandet hervorgehoben werden. (2) zeigt ein Suchergebnis. Die Besuche des Ergebnisses werden rot umrandet hervorgehoben dargestellt. (3) zeigt die Hervorhebung eines Repräsentanten durch einen roten Pfeil.

Der Repräsentant wird automatisch berechnet und durch einen roten Pfeil hervorgehoben wie in Bereich (3) der Abbildung 5.14 dargestellt. Die Methode zur Berechnung des Repräsentanten kann ausgewählt werden sowie die Hervorhebung deaktiviert werden.

Eine weitere Analysemöglichkeit stellt der lokale Scanpfad dar. Dies ist eine Technik, bei der die Pattern Search Lanes per Brushing & Linking mit der Stimulus–Visualisierung verbunden sind. Bei dieser Technik wird in den Pattern Search Lanes im Modus zur Verarbeitung auf AOI–Ebene ein AOI–Besuch ausgewählt und dadurch schwarz umrandet hervorgehoben. Daraufhin wird dessen Vorgänger und Nachfolger rot umrandet hervorgehoben. Dies ist in Abbildung 5.15 in Bereich (1) abgebildet. Nun wird mittels des Brushing & Linking Mechanismus in der Stimulus–Visualisierung in der Scanpfad Ansicht der Bereich des zugehörigen Scanpfads, der dieser Abfolge von AOI–Besuchen entspricht, durch eine rote Darstellung hervorgehoben. Dies ist in der Abbildung in Bereich (2) dargestellt.

Ein großer Vorteil der Pattern Search Lanes soll am Beispiel von AOI Abfolgen erklärt werden. Die Nutzung für Abfolgen von Schema–Klassen erfolgt analog. Wie in Abschnitt 5.5.1 beschrieben, können die AOI Abfolgen mehrerer Probanden durch einen Repräsentanten stellvertretend dargestellt werden. Durch die Zusammenfassung der AOI Abfolgen mehrerer Probanden zu einem Repräsentanten und durch die Nutzung *jeder* Bahn der Pattern Search Lanes zur Darstellung eines solchen Repräsentanten, kann eine beliebige Anzahl von Probanden in den Pattern Search Lanes dargestellt werden. Eine Bahn kann somit beispielsweise zehn Probanden repräsentieren. Bei zehn Bahnen ermöglicht dies die Analyse der AOI Abfolgen von 100 Probanden. Somit können auch Studien mit sehr vielen

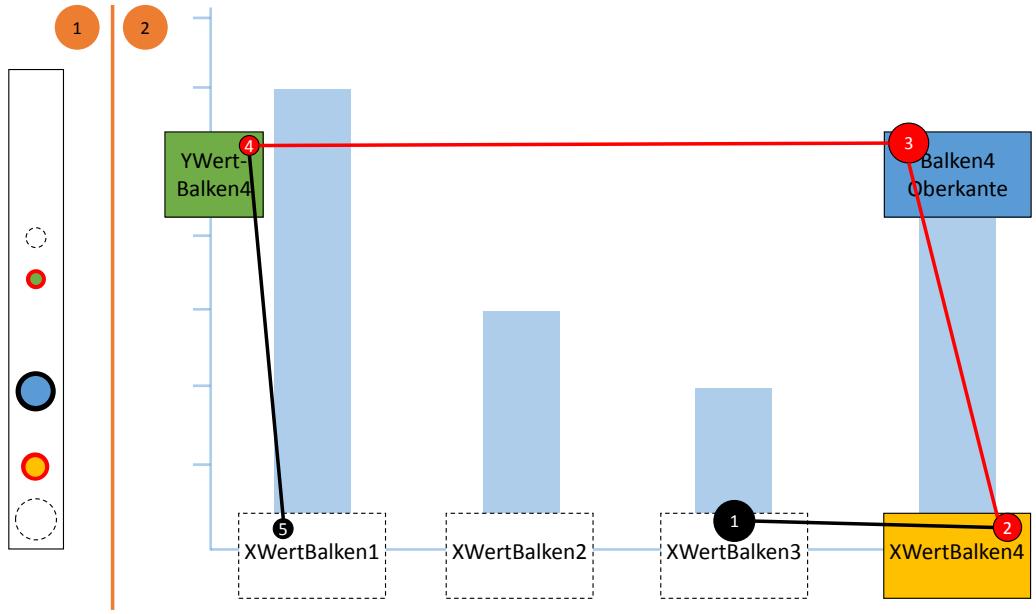

Abbildung 5.15: Der lokale Scanpfad. In (1) wird ein AOI–Besuch ausgewählt und hervorgehoben sowie dessen Vorgänger und Nachfolger rot hervorgehoben. In (2) wird der Bereich des zugehörigen Scanpfads rot hervorgehoben, der der in den Pattern Search Lanes hevorgehobenen Abfolge von AOIs entspricht.

Teilnehmern und entsprechender Aussagekraft analysiert werden, genauso wie die Untersuchung der Schema–Klassen sich auf sehr viele Stimuli stützen kann. Dieses Nutzungsszenario der Pattern Search Lanes ist nicht in das Konzept der vorliegenden Arbeit eingearbeitet. Es eignet sich jedoch gut zur Nutzung bei der Analyse von Eye–Tracking–Daten einer großen Anzahl von Probanden.

Suche

Dieser Abschnitt beschreibt die Suchverfahren, die für die Suche in den Pattern Search Lanes zur Verfügung stehen. Die Suchverfahren können sowohl im Modus zur Verarbeitung auf AOI–Ebene als auch im Modus zur Verarbeitung auf Schema–Klassen–Ebene genutzt werden. Für die Suche im Modus zur Verarbeitung auf AOI–Ebene sind insbesondere AOIs mit positivem Gewicht von großem Interesse, da sie mit hoher Wahrscheinlichkeit diejenigen Bildbereiche repräsentieren, die zur Lösung der gestellten Aufgabe notwendig sind. Nachfolgend werden die verschiedenen Suchverfahren erklärt, die für die Pattern Search Lanes verwendet werden können.

- Elementsuche: Es können mehrere AOI– oder Schema–Klassen–Besuche ausgewählt werden. Diese werden anschließend in den anderen Bahnen gesucht und hervorgehoben. Dies soll dazu dienen die Gestaltung der Verteilung der ausgewählten Besuche in den anderen Bahnen untersuchen zu können. Durch die Elementsuche erhält der Benutzer Aufschluss darüber in

welcher Reihenfolge die ausgewählten Besuche in den anderen Bahnen auftreten. Ein Beispiel dafür findet sich in Abbildung 5.16 in Bereich (1).

- Exakte Suche: Hierbei wird eine Abfolge von AOI– oder Schema–Klassen–Besuchen in einer Bahn ausgewählt. Die ausgewählten Elemente müssen nicht zwingend direkt aufeinanderfolgen. Sie werden im Anschluss in exakt der ausgewählten Reihenfolge in den anderen Bahnen gesucht und hervorgehoben. Im Gegensatz zur Auswahl beinhalten die Ergebnisse nur direkt aufeinanderfolgende Elemente. Abbildung 5.16 zeigt ein Beispiel für die exakte Suche in Bereich (2).
- Unscharfe Suche: Da sich die Abfolgen aufgrund der gegebenen Aufgabenstellung häufig ähneln aber nicht exakt gleich sind, steht das Instrument der unscharfen Suche zur Verfügung. Dies ermöglicht es zum einen in anderen Bahnen Abfolgen zu finden, die zwar dieselben Elemente wie in der Auswahl beinhalten, deren Reihenfolge jedoch nicht derjenigen in der Auswahl entspricht. Zum anderen ermöglicht es das Auffinden von Abfolgen die neben Elementen der Auswahl auch andere Elemente beinhalten. Die unscharfe Suche wird unter Zuhilfenahme der Levenshtein–Distanz realisiert.

Die Durchführung der unscharfen Suche soll am Beispiel des Modus zur Verarbeitung auf AOI–Ebene erklärt werden, die Anwendung und Durchführung im Modus zur Verarbeitung auf Schema–Klassen–Ebene erfolgt analog. Die Kosten für Einfüge– und Löschoperationen betragen im Folgenden zwei, die Ersetzungskosten eins. Um eine Abfolge von AOI–Besuchen unscharf zu suchen, werden zunächst die zu suchenden AOI–Besuche ausgewählt. Pro Proband wird ein Suchfenster über seine Abfolge von AOI–Besuchen gelegt, welches pro Suchschritt um ein Element weitergeschoben wird. Innerhalb dieses Suchfensters wird die markierte Abfolge unscharf gesucht. Dazu wird die Levenshtein–Distanz der markierten Abfolge mit der Abfolge innerhalb des Suchfensters berechnet und der Inhalt des Suchfensters mitsamt der berechneten Levenshtein–Distanz zur Menge der Ergebnisse hinzugefügt. Sobald das Suchfenster über das Ende der Abfolge hinausläuft, wird die Suche beendet. Die Größe des Suchfensters hat den Wert

$$g = i + k$$

Dabei entspricht i der Anzahl der ausgewählten AOI–Besuche. Der Wert für k dient der Verlängerung des Suchfensters und kann frei gewählt werden. Es gilt lediglich die Einschränkung $k \geq 0$.

Abbildung 5.16 zeigt in Bereich (3) ein Beispiel für die unscharfe Suche. Das Suchfenster wird dabei nicht verlängert, es ist also $k = 0$. Es wird zuerst die Abfolge in Bereich (4) gefunden. Die Kosten dieses Ergebnisses betragen 0, da es sich um die gleiche Abfolge handelt, wie die die markiert ist. Bereich (5) enthält ein Ergebnis, dessen Kosten 3 betragen, da drei Ersetzungen vorzunehmen sind. Pro Proband werden die Ergebnisse nach Kosten sortiert angezeigt.

Graph

Die im Folgenden beschriebene Visualisierung zeigt eine Abfolge von AOIs oder Schema–Klassen auf einem Graphen. Abbildung 5.17 zeigt die Visualisierung. Sie enthält Gruppierungsknoten und Besuchsknoten. Besuchsknoten werden um ihren zugehörigen Gruppierungsknoten herum kreisförmig angeordnet. Zur Verdeutlichung sind in der Abbildung gestrichelte Hilfslinien eingezeichnet,

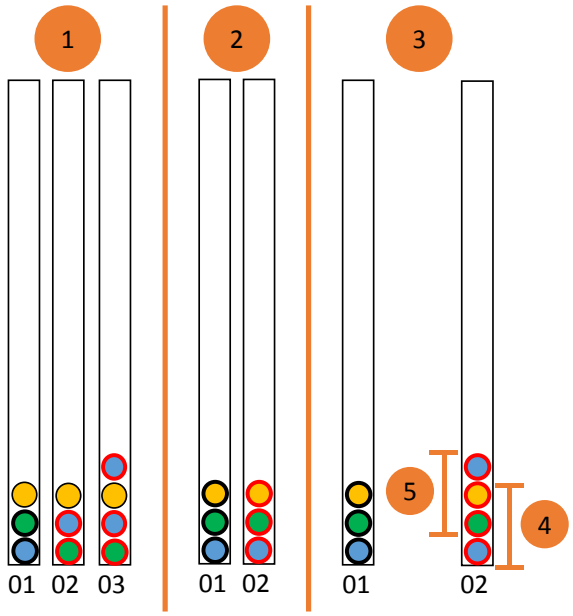

Abbildung 5.16: Suche in den Pattern Search Lanes. (1) zeigt die Elementsuche mit der die Reihenfolge der in Bahn 01 ausgewählten Besuche in den anderen Bahnen untersucht werden kann. Die Suchergebnisse werden rot umrandet dargestellt. (2) zeigt die exakte Suche. Die zu suchenden Besuche sind in Bahn 01 ausgewählt. Das Suchergebnis in Bahn 02 enthält die exakt gleiche Abfolge. (3) zeigt die unscharfe Suche. Die markierte Abfolge in Bahn 01 wird unscharf gesucht. Bahn 02 enthält ein Suchergebnis (4) mit den Kosten 0 und ein Suchergebnis (5) mit den Kosten 3.

auf denen der Mittelpunkt der Besuchsknoten liegt. In der eigentlichen Visualisierung sind diese Hilfslinien nicht vorhanden. Einen ähnlichen Ansatz verwenden Neupert et al. zur Visualisierung von Scanpfaden [NEP13].

Für das Layout des Graphen kommen die in Abschnitt 5.7.1 beschriebenen Layout Verfahren zum Einsatz. Eine Nebenbedingung für diese Verfahren ist, dass die gruppierte Darstellung auch bei einer Änderung der Anordnung der Graphknoten aufrechterhalten werden muss, um die Besuchsknoten ihren zugehörigen Gruppierungsknoten zuordnen zu können.

Im Modus zur Verarbeitung auf AOI–Ebene entsprechen die Gruppierungsknoten den AOIs. Ein Besuchsknoten entspricht einem AOI–Besuch. Eine Kante entspricht der Transition zwischen zwei AOI–Besuchen. Im Modus zur Verarbeitung auf Schema–Klassen–Ebene entspricht ein Gruppierungsknoten einer Schema–Klasse. Besuchsknoten entsprechen einem Schema–Klassen–Besuch. Eine Kante entspricht einer Transition zwischen zwei Schema–Klassen–Besuchen. Für die Darstellung der AOIs, AOI–Besuche, Schema–Klassen und Schema–Klassen–Besuche wird die in Abschnitt 5.7.1 beschriebene Darstellungsform genutzt. Auch die Randdarstellung von negativ gewichteten AOIs und AOI–Besuchen wird umgesetzt wie in Abschnitt 5.4.2 geschildert. Informationen zu AOIs, AOI–Besuchen, Schema–Klassen und Schema–Klassen–Besuchen werden auf die Art und Weise angezeigt,

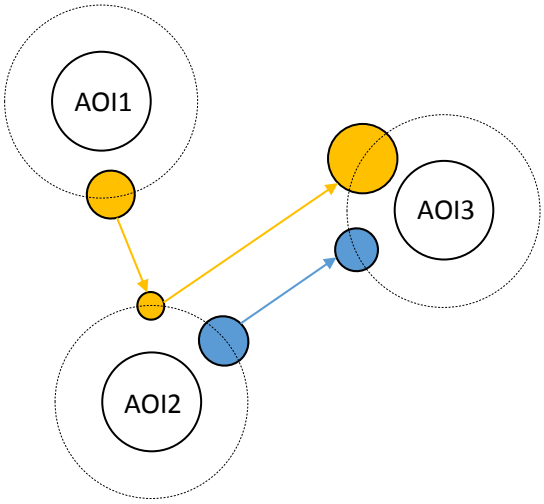

Abbildung 5.17: Der Graph. Diese Visualisierung zeigt eine Abfolge von AOIs oder Schema–Klassen auf einem Graphen. AOI– beziehungsweise Schema–Klassen–Besuche werden als Knoten dargestellt und um den zu den Besuchen gehörenden Knoten, der die AOI oder Schema–Klasse repräsentiert, kreisförmig angeordnet. Die gestrichtelten Hilfslinien dienen nur der Verdeutlichung und sind in der eigentlichen Visualisierung nicht enthalten.

wie in Abschnitt 5.7.1 ausgeführt. Bei Auswahl einer Kante werden die Ontologie Daten dieser Kante angezeigt, sofern diese Transition in der zugehörigen Ontologie eine Kante besitzt. Weiterhin kann der Stimulus eingeblendet werden wie in Abschnitt 5.7.1 beschrieben.

Die Visualisierung kann auf zwei Arten verwendet werden. Zum einen als Detailansicht für die Pattern Search Lanes. Die Auswahl einer Bahn hat dann zur Folge, dass nur dieser Proband beziehungsweise dieser Stimulus in der Graphdarstellung angezeigt wird. Zum anderen kann die Visualisierung von den Pattern Search Lanes entkoppelt als eigenständige Ansicht für mehrere Probanden beziehungsweise Stimuli gleichzeitig benutzt werden. Dabei kann der Repräsentant ein– oder ausgeblendet werden sowie das Verfahren zur Berechnung des Repräsentanten ausgewählt werden. Abbildung 5.17 zeigt ein Beispiel für die entkoppelte Ansicht. Dabei werden die AOI–Besuche zweier Probanden dargestellt.

Befindet man sich im Modus zur Verarbeitung auf AOI–Ebene und ist nur ein Proband ausgewählt, kann der Blickpfad des Probanden in Echtzeit in einer Animation dargestellt werden. Der Ablauf der Animation wird im Folgenden beschrieben. Zu Anfang sind nur die AOI Gruppierungsknoten sichtbar. Mit dem Start der Animation wächst der erste AOI–Besuchknoten in seinem Durchmesser. Nach Beendigung des AOI–Besuchs wächst eine Kante in Richtung des nächsten AOI–Besuchsknotens entsprechend der Dauer der AOI–Transition. Nach Beendigung der Transition wächst der nächste AOI–Besuchsknoten. Dies wiederholt sich bis der Blickpfad des Probanden komplett abgearbeitet ist. Synchron zur Animation der Besuchsknoten und Kanten werden in einer 3D–Darstellung des Gehirns diejenigen Hirnbereiche hervorgehoben, die zum aktuellen Animationszeitpunkt aktiv sind. Dabei handelt es sich um Regionen die dem deklarativen Wissen der AOIs entsprechen und Regionen

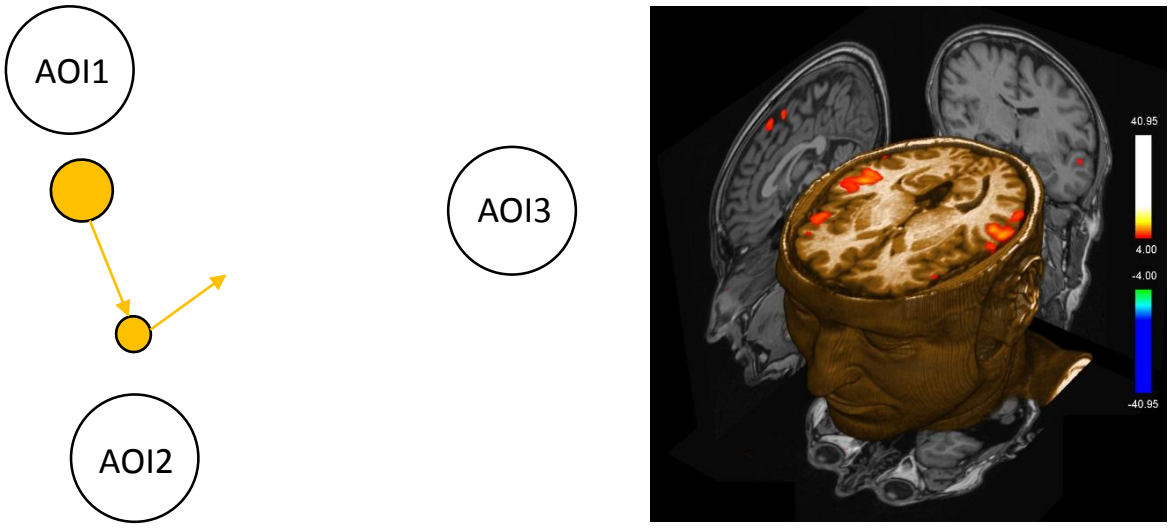

Abbildung 5.18: Visualisierung der aktiven Gehirnregionen. Im Modus zur Verarbeitung auf AOI Ebene kann mittels des Graphen der Blickpfad eines Probanden mit synchroner Darstellung der während des Blickpfadverlaufs aktiven Gehirnregionen visualisiert werden [Ruh12].

zur Steuerung der Augenbewegung. Eine Momentaufnahme der Animation ist in Abbildung 5.18 dargestellt. Die Animation bietet die Möglichkeit die Abspielgeschwindigkeit einzustellen.

Ergebniszusammenstellung

Die Ergebniszusammenstellung ist dafür vorgesehen ein Ergebnis von Hand zusammenzustellen oder ein berechnetes Ergebnis im Anschluss an seine Berechnung von Hand zu ändern. Ausgehend von den Analyseergebnissen können AOIs und Schema–Klassen per Drag & Drop in die gewünschte Reihenfolge gebracht werden. Einzelne Elemente können gelöscht und vertauscht werden sowie das gesamte Ergebnis gelöscht und anschließend wieder von neuem begonnen werden. Anschließend kann die zusammengestellte Abfolge zur Weiterverarbeitung verwendet werden Abbildung 5.19 zeigt die Ergebniszusammenstellung am Beispiel des Modus zur Verarbeitung auf AOI–Ebene. Die Ergebniszusammenstellung bei der Verarbeitung auf Schema–Klassen–Ebene funktioniert analog dazu. AOIs beziehungsweise Schema–Klassen werden aus Bereich (1) nach Bereich (2) gezogen. Bereich (1) ist horizontal scrollbar und Bereich (2) ist vertikal scrollbar, um viele AOIs beziehungsweise Schema–Klassen verarbeiten zu können.

Brushing & Linking

Sowohl der Modus zur Verarbeitung auf AOI–Ebene als auch der Modus zur Verarbeitung auf Schema–Klassen–Ebene stellt zwei Betriebsarten für Brushing & Linking zur Verfügung. Der Modus zur

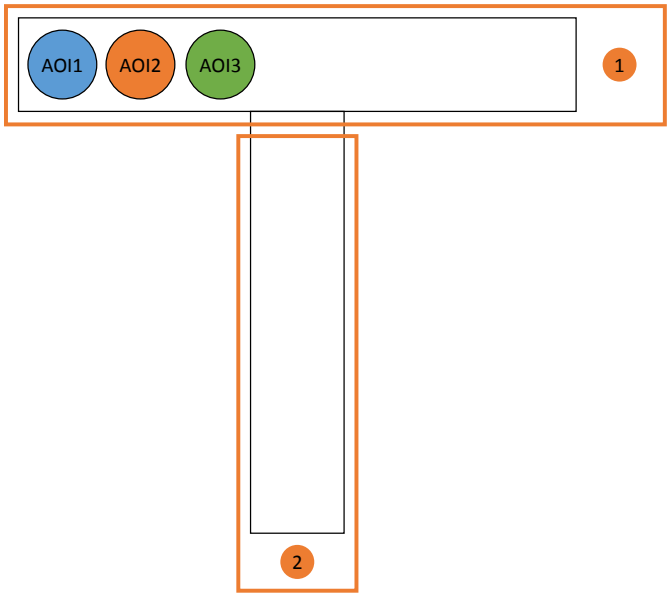

Abbildung 5.19: Die Ergebniszusammenstellung. Hier kann von Hand eine repräsentative Abfolge von AOIs oder Schema–Klassen erstellt werden. Dazu werden die AOIs beziehungsweise Schema–Klassen per Drag & Drop von (1) nach (2) gezogen.

Verarbeitung auf AOI–Ebene kennt die Betriebsarten AOIs und Probanden. Der Modus zur Verarbeitung auf Schema–Klassen–Ebene beinhaltet die Betriebsarten Schema–Klassen und Stimuli.

Im Modus zur Verarbeitung auf AOI–Ebene bewirkt die Betriebsart AOIs, dass die Farben für AOIs in allen Visualisierungen synchronisiert werden. Dies bedeutet, dass dieselbe AOI in verschiedenen Visualisierungen dieselbe Farbe hat. Währenddessen haben alle Probanden dieselbe neutrale Farbe. Dies ist für die Scanpfad Ansicht der Stimulus–Visualisierung (1), die Pattern Search Lanes (2) und den Graphen (3) in Abbildung 5.20 dargestellt. In der Betriebsart Probanden werden die Farben der Probanden in allen Visualisierungen synchronisiert und die AOIs haben alle dieselbe neutrale Farbe. Abbildung 5.21 zeigt dies für die Scanpfad Ansicht der Stimulus–Visualisierung (1), die Pattern Search Lanes (2) und den Graphen (3). Die Betriebsarten des Brushing & Linking Mechanismus für den Modus zur Verarbeitung auf Schema–Klassen–Ebene funktionieren analog.

Das Auswählen eines Probanden, einer AOI oder Schema–Klasse sowie eines AOI– oder Schema–Klassen–Besuchs in einer Visualisierung hat zur Folge, dass das entsprechende Element in den anderen Visualisierungen ebenfalls als ausgewählt dargestellt wird. Die Anzeige von Probanden, AOIs und Schema–Klassen sowie AOI– und Schema–Klassen–Besuchen kann gefiltert werden und die anderen Visualisierungen passen sich entsprechend an.

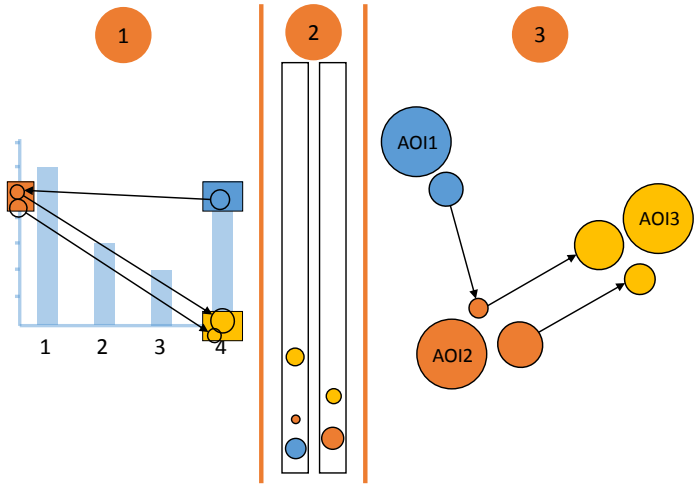

Abbildung 5.20: Brushing & Linking im Modus zur Verarbeitung auf AOI–Ebene in der Betriebsart AOIs. In dieser Betriebsart werden die Farben der AOIs in den verschiedenen Visualisierungen synchronisiert. AOIs haben dieselbe Farbe in der Stimulus–Visualisierung (1), in den Pattern Search Lanes (2) und im Graph (3).

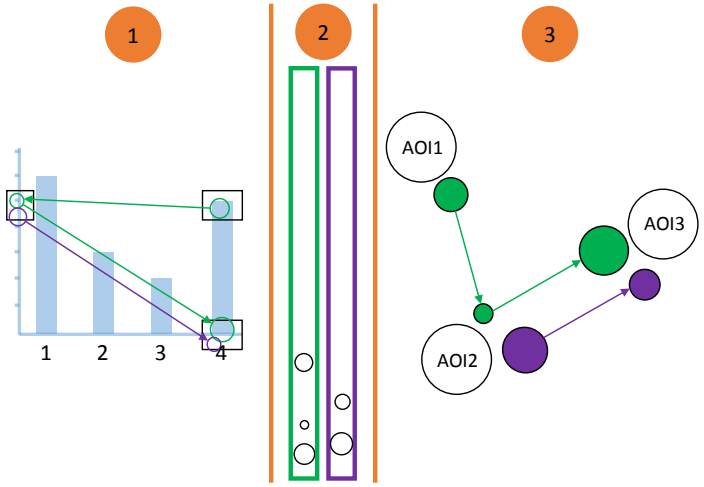

Abbildung 5.21: Brushing & Linking im Modus zur Verarbeitung auf AOI–Ebene in der Betriebsart Probanden. In dieser Betriebsart werden die Farben der Probanden in den verschiedenen Visualisierungen synchronisiert. Die Probanden haben dieselbe Farbe in der Stimulus–Visualisierung (1), in den Pattern Search Lanes (2) und im Graph (3).

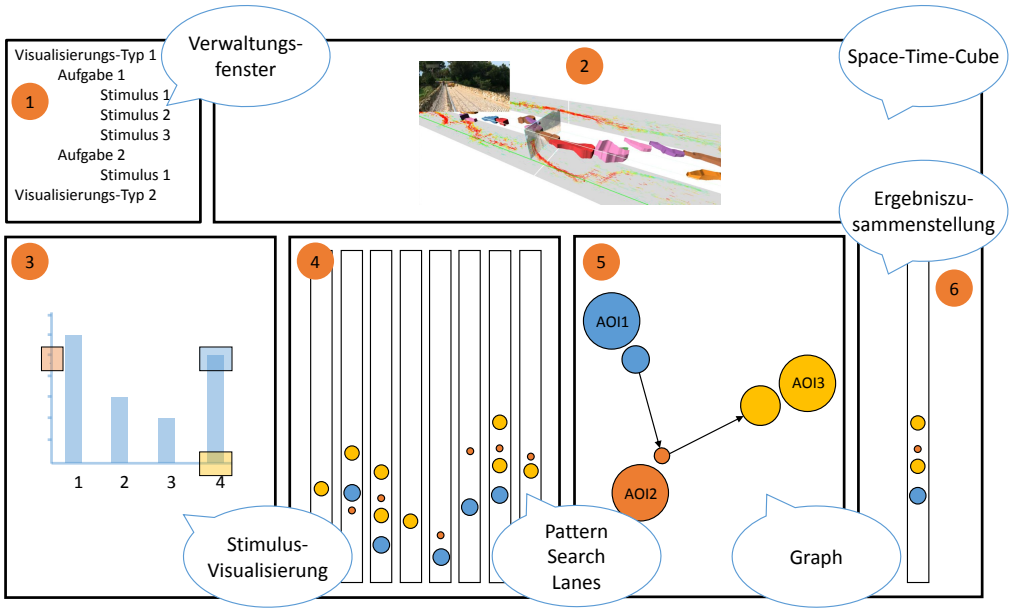

Abbildung 5.22: Kombination von Visualisierungen im Modus zur Verarbeitung auf AOI–Ebene. Dieser Modus kombiniert das Verwaltungsfenster (1), die Visualisierungstechnik des Space–Time–Cubes (2), die Stimulus–Visualisierung (3), die Pattern Search Lanes (4), den Graph (5) und die Ergebniszusammenstellung (6).

5.7.3 Die beiden Modi des Visualisierungs– und Analysekonzepts

Die Visualisierungs– und Analysetechniken werden nun zu einem aus zwei Modi bestehenden Visualisierungs– und Analysekonzept zusammengesetzt. Der erste Modus entspricht der Verarbeitung auf AOI–Ebene des Ablaufs zur Gewinnung einer Lesevorschrift. Beim zweiten Modus handelt es sich um die Verarbeitung auf Schema–Klassen–Ebene des Ablaufs zur Gewinnung einer Lesevorschrift. Für jeden der beiden Modi werden verschiedene Visualisierungs– und Analysetechniken miteinander kombiniert. Die beiden Modi sowie die Durchführung des Wechsels zwischen den beiden Modi werden im Folgenden beschrieben.

Modus zur Verarbeitung auf AOI–Ebene

Abbildung 5.22 zeigt die Kombination von Visualisierungen des Modus zur Verarbeitung auf AOI–Ebene. Die Kombination beinhaltet das Verwaltungsfenster (1), die Visualisierungstechnik des Space–Time–Cubes (2), die Stimulus–Visualisierung (3), die Pattern Search Lanes (4), den Graph (5) und die Ergebniszusammenstellung (6). Jede dieser Visualisierungen läuft im Modus zur Verarbeitung auf AOI–Ebene. Die Visualisierungen beziehen sich auf Probanden und AOIs. Es werden die AOI–Besuche und Transitionen für die Probanden dargestellt. Jede Visualisierung kann einzeln bildschirmfüllend dargestellt werden.

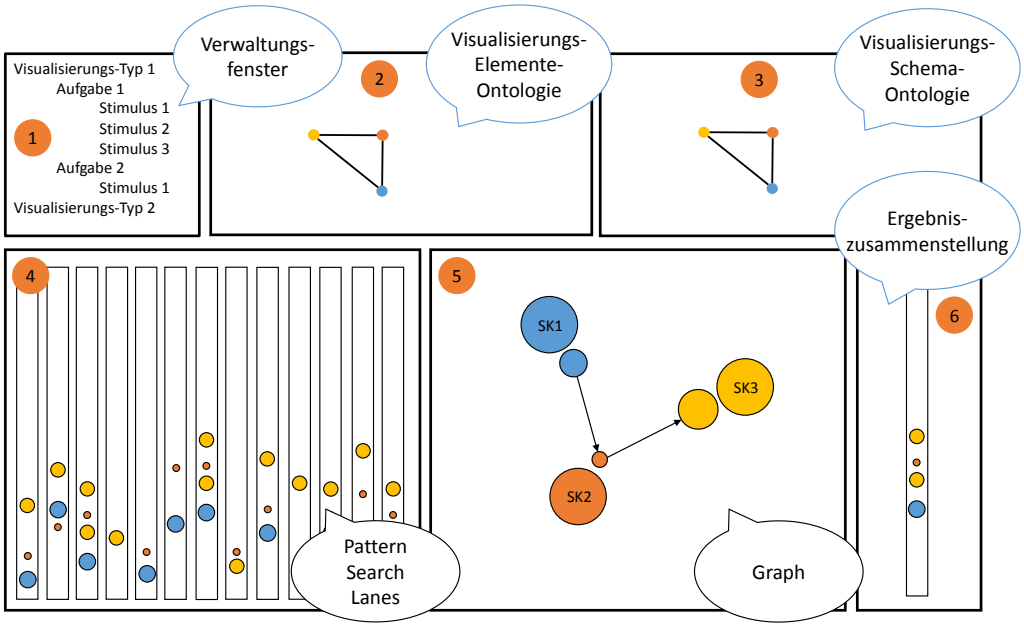

Abbildung 5.23: Kombination von Visualisierungen im Modus zur Verarbeitung auf Schema–Klassen–Ebene. Die Kombination beinhaltet das Verwaltungsfenster (1), die Ontologie–Visualisierung der Visualisierungs–Elemente–Ontologie (2), die Ontologie–Visualisierung der Visualisierungs–Schema–Ontologie (3), die Pattern Search Lanes (4), den Graph (5) und die Ergebniszusammenstellung (6). Die Abkürzung SK steht für Schema–Klasse.

Modus zur Verarbeitung auf Schema–Klassen–Ebene

Abbildung 5.23 zeigt die Kombination von Visualisierungen des Modus zur Verarbeitung auf Schema–Klassen–Ebene. Die Kombination beinhaltet das Verwaltungsfenster (1) und jeweils eine Ontologie–Visualisierung der Visualisierungs–Elemente–Ontologie (2) und der Visualisierungs–Schema–Ontologie (3). Weiterhin sind die Pattern Search Lanes (4), der Graph (5) und die Ergebniszusammenstellung (6) beinhaltet. Jede der Visualisierungen wird im Modus zur Verarbeitung auf Schema–Klassen–Ebene angezeigt. Die Visualisierungen dieses Modus beziehen sich auf Stimuli und Schema–Klassen. Es werden die Abfolgen von Schema–Klassen für die Stimuli dargestellt. Jede Visualisierung kann einzeln bildschirmfüllend dargestellt werden.

Wechsel zwischen Visualisierungstypen, Aufgaben und Stimuli

Das Verwaltungsfenster ist in Abbildung 5.24 dargestellt. Es wird in die Kombination aus Visualisierungen beider Modi mit aufgenommen. Dadurch wird die Gewinnung von Lesevorschriften für mehrere verschiedene Visualisierungstypen und Aufgaben ermöglicht ohne mehrere Instanzen der Software starten zu müssen. Weiterhin wird das Verwaltungsfenster dazu genutzt, um zwischen

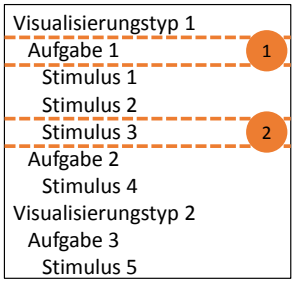

Abbildung 5.24: Wechsel zwischen Visualisierungstypen, Aufgaben und Stimuli. Der Wechsel wird anhand des Verwaltungsfensters vollzogen. Das Auswählen von (1) startet den Modus zur Verarbeitung auf Schema–Klassen–Ebene für Aufgabe 1 des Visualisierungstyps 1. Durch die Auswahl von (2) wird der Modus zur Verarbeitung auf AOI–Ebene für Stimulus 3 von Aufgabe 1 des Visualisierungstyps 1 gestartet.

den beiden Modi des Visualisierungs– und Analysekonzepts zu wechseln, um die Eingabedaten zu importieren und um Visualisierungstypen, Aufgaben und Stimuli hinzuzufügen und zu entfernen. Durch die Auswahl eines Stimulus wird der Modus zur Verarbeitung auf AOI–Ebene für diesen Stimulus gestartet. Sobald die repräsentative AOI Reihenfolge für diesen Stimulus feststeht, wird diese im Hintergrund dem Modus zur Verarbeitung auf Schema–Klassen–Ebene für die Aufgabe des Stimulus und dem Visualisierungstyp des Stimulus hinzugefügt. Der Modus zur Verarbeitung auf Schema–Klassen–Ebene kann durch die Auswahl einer Aufgabe gestartet werden. Die Hierarchie von Visualisierungstypen, Aufgaben und Stimuli im Verwaltungsfenster resultiert aus der Tatsache, dass eine Lesevorschrift für nur eine Kombination aus Visualisierungstyp und Aufgabe gilt. Für eine weitere Aufgabe zum selben Visualisierungstyp muss eine weitere Lesevorschrift erzeugt werden. Daher werden im Verwaltungsfenster mehrere Aufgaben pro Visualisierungstyp angezeigt.

Die folgende Aufzählung bezieht sich auf Abbildung 5.24 und gibt Aufschluss darüber, was bei der Auswahl eines Bereichs im Verwaltungsfenster passiert.

- Bereich (1): Der Modus zur Verarbeitung auf Schema–Klassen–Ebene wird für Aufgabe 1 des Visualisierungstyps 1 gestartet. Das Ergebnis der Verarbeitung ist ein Lesevorschrift für Aufgabe 1 des Visualisierungstyps 1.
- Bereich (2): Der Modus zur Verarbeitung auf AOI–Ebene wird für Stimulus 3 von Aufgabe 1 des Visualisierungstyps 1 gestartet. Der Ergebnis der Verarbeitung ist eine repräsentative AOI Abfolge für Stimulus 3 für Aufgabe 1 des Visualisierungstyps 1.

6 Implementierung

Dieses Kapitel behandelt die Implementierung der wichtigsten Teile des im vorigen Kapitel beschriebenen Konzepts als Prototyp.

Zu Beginn werden in Abschnitt 6.1 die zur Implemetierung des Prototyps verwendeten Technologien vorgestellt. Daran schließt sich in Abschnitt 6.2 eine Beschreibung der Datenbank an, die der Prototyp zur Speicherung von Eye–Tracking–Daten einsetzt. Abschnitt 6.3 erläutert die pluginbasierte Architektur des Prototypen und Abschnitt 6.4 die verwendete Technologie für den Einsatz von Plugins. Die für den Prototyp entwickelten Bibliotheken werden in Abschnitt 6.5 behandelt. Abschnitt 6.6 beschreibt die notwendigen Vorbereitungen um mit dem Prototypen arbeiten zu können. Schließlich wird in Abschnitt 6.7 die Benutzeroberfläche des Prototyps erläutert.

6.1 Verwendete Technologien

In diesem Abschnitt werden die Software und die Bibliotheken vorgestellt, die bei der Entwicklung zum Einsatz kamen.

6.1.1 Software

Zur Erstellung des Prototyps wurde die Programmiersprache C# in Kombination mit der Windows Presentation Foundation auf der Software–Plattform .NET 4.5 von Microsoft verwendet. Die Programmierung fand unter Windows 8 mit der Entwicklungsumgebung Visual Studio 2012 statt.

Als Datenbank kommt die localDB zum Einsatz. Dabei handelt es sich um eine leichtgewichtige Variante des Microsoft SQL Servers. Um die localDB nutzen zu können reicht eine Installation des Microsoft SQL Server 2012 Express aus. Zur Ausführung von administrativen Tätigkeiten auf der Datenbank wurde das SQL Server Management Studio von Microsoft verwendet.

6.1.2 Bibliotheken

Als externe Bibliotheken kommen das Entity Framework in der Version 6.1.0[1], dotNetRDF in der Version 1.0.4[2], das Extended WPF Toolkit in der Version 2.1.0[3] sowie das Windows 7 API Code

[1] http://msdn.microsoft.com/en-US/data/ef
[2] http://www.dotnetrdf.org
[3] http://wpftoolkit.codeplex.com

Pack – Shell in der Version 1.1.0.0[4] zum Einsatz. Jede Bibliothek kann unter Umständen weitere Abhängigkeiten besitzen auf die an dieser Stelle aber nicht näher eingegangen werden soll. Alle Bibliotheken lassen sich komfortabel über den Paket Manager von Visual Studio installieren. Sollte eine Bibliothek weitere Abhängikeiten besitzen, so installiert der Paket Manager diese automatisch mit.

Das Entity Framework wird zur Datenbankanbindung benötigt. Die Bibliothek dotNetRDF dient zur Verwaltung der Ontologien. Das Extended WPF Toolkit sowie das Windows 7 API Code Pack – Shell beinhalten Oberflächenelemente die benötigt werden, jedoch im Funktionsumfang der Windows Presentation Foundation nicht enthalten sind.

6.2 Datenbank

Zur Speicherung der Eye–Tracking–Daten wird eine Datenbank verwendet. Das Datenbankschema basiert auf dem Datenbankschema von Strohmaier und wurde für die Nutzung im Prototyp angepasst [Str14]. Der relevante Ausschnitt des Schemas ist in Abbildung 6.1 dargestellt. Die Eye–Tracking–Daten werden über einen Importer in die Datenbank importiert. Beim Importer wurde auf die Implementierung von Herr zurückgegriffen und diese für die Nutzung im Prototyp angepasst [Her13]. Der Importer kann mit Dateien des Eyetrackers Tobii T60 XL im Tab–Separated–Values–Format umgehen. Er wurde bei der Implementierung des Prototyps um die Möglichkeit des Imports von Fixationen erweitert, die vom Fixationsfilter der Software des Eyetrackers in die Ausgabedatei eingetragen werden. Im Folgenden befindet sich eine Beschreibung der einzelnen Datenbank Tabellen.

Fixation: In diese Tabelle werden die vom Fixationsfilter des Eyetrackers berechneten Fixationen und ihre zugehörigen Informationen eingetragen. Die Angabe der Position erfolgt in Stimulus–Koordinaten. Dies bedeutet, dass über den Stimulus ein zweidimensionales Koordinatensystem gelegt. Die linke obere Ecke entspricht dabei dem Punkt (0,0) des Koordinatensystems.

AOI_Fixation: Hierbei handelt es sich um die Tabelle, in die diejenigen Fixationen eingetragen werden, die in einer AOI liegen.

GazePoint: Die Tabelle, die die Gazepoints beinhaltet. Ein Gazepoint ist der Blickpunkt, der vom Eyetracker ohne weitere Filterung ausgegeben wird. Seine Positionsangabe erfolgt in Bildschirm–Koordinaten.

Participant: Diese Tabelle enthält die Probanden mit denen das Eye–Tracking durchgeführt wird.

Stimulus: Die Stimuli werden in diese Tabelle eingetragen. Dabei werden Angaben über Höhe und Breite des Stimulus mit aufgenommen, nicht jedoch der Stimulus selbst.

AreaOfInterest: In diese Tabelle werden die AOIs gespeichert. Die zusätzliche Speicherung der Farbe der AOIs ermöglicht es, beim erneuten Start des Prototypen auf die gespeicherten Farben zurückgreifen zu können.

[4] https://www.nuget.org/packages/Microsoft.WindowsAPICodePack-Shell/1.1.0

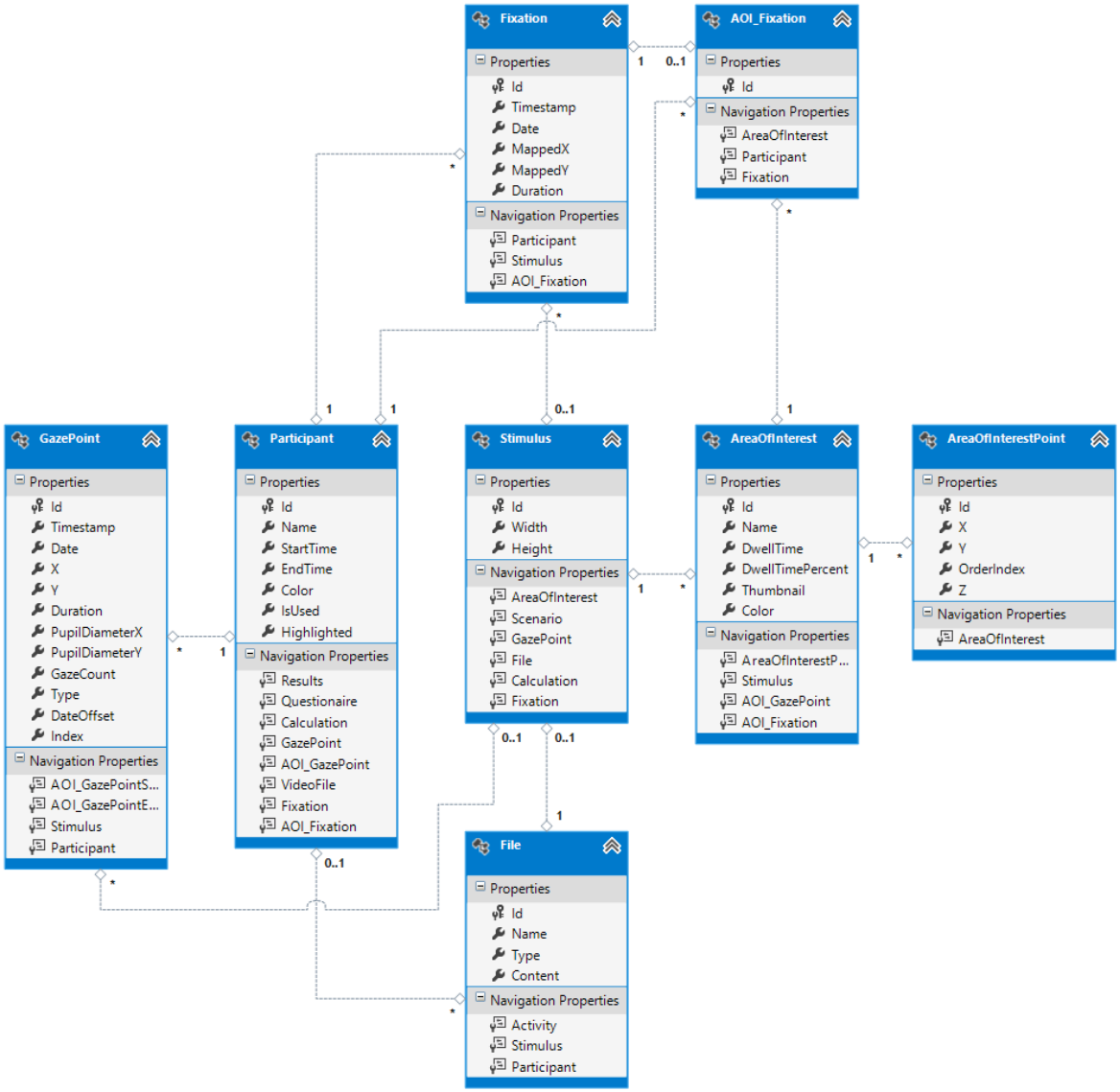

Abbildung 6.1: Ausschnitt des Datenbankschemas des Prototyps. Der relevante Ausschnitt des Datenbankschemas des Prototyps beinhaltet die Klassen, die notwendig sind, um die Eye–Tracking–Daten in der Datenbank speichern zu können.

AreaOfInterestPoint: Da im Prototyp ausschließlich rechteckige AOIs verwendet werden, enthält diese Tabelle pro AOI jeweils zwei Punkte, da cEdit ein AOI Rechteck durch zwei Punkte definiert. Beim ersten Punkt handelt es sich um den oberen linken Punkt des AOI Rechtecks. Der zweite Punkt ist der Punkt unten rechts.

File: Die Tabelle enthält die Informationen über die Bilddatei eines Stimulus. Es werden der Dateiname und das eigentliche Bild in Form eines Bytearrays gespeichert.

6.3 Architektur

Abbildung 6.2 beinhaltet die Architektur des Prototyps als Klassendiagramm mit den wichtigsten Namensräumen und deren Unterteilung in weitere Namensräume. Die Abbildung zeigt die Verteilung der Klassen über die Namensräume und die Abhängigkeiten der Namensräume und Klassen. Es sind die vier wichtigsten Namensräume `Libraries`, `Plugins`, `vaocp` und `Model` dargestellt.

Die Architektur folgt dem MVVM Entwurfsmuster. MVVM bedeutet *Model–View–ViewModel*. Beim *Model* des MVVM Entwurfsmusters handelt es sich um die Datenverwaltungsschicht. Die *View* des MVVM–Entwurfsmusters beinhaltet die grafische Benutzeroberfläche. Das *ViewModel* stellt die Verbindung zwischen *View* und *Model* her. Im Prototyp entspricht der Namensraum `Model` dem *Model* im MVVM Entwurfsmuster. Die Klassen `AOIViewModel`, `AOIOntologyViewModel`, `SchemeViewModel` und `SchemeOntologyViewModel` übernehmen die Funktion des *ViewModels* des MVVM Entwurfsmusters. Die Klassen der Namensräume der Plugins sowie die Klasse `StimulusContainer` entsprechen der *View* des Entwurfsmusters.

Der Namensraum `Libraries` unterteilt sich in zwei weitere Namensräume, `PatternSearch` und `GraphRenderer`. In den Klassen des Namensraums `PatternSearch` sind alle Funktionen ausgelagert, die die Pattern Search Lanes in beiden Modi der Benutzeroberfläche benutzen. Der Namensraum `GraphRenderer` enthält Klassen zur Anzeige von Graphen. Sie wurde implementiert, da frei verfügbare Bibliotheken für die Graphvisualisierung wie etwa GraphSharp[5] nicht den für den Prototyp erforderlichen Funktionsumfang bieten. Beispielsweise ist es für den Prototyp notwendig Knoten automatisch kreisförmig um einen anderen Knoten herum zu positionieren, was mit GraphSharp nicht möglich ist.

Der Namensraum `Plugins` umfasst die Namensräume aller Visualisierungs–Plugins sowie den Namensraum `PluginCommunication`. Bei den Visualisierungs–Plugins handelt es sich um die Elemente, die auf der Benutzeroberfläche zu sehen sind, wie die Pattern Search Lanes, den Graphen und die Ontologie Visualisierung. Der Namensraum `PluginCommunication` beinhaltet Interfaces und Klassen für die ViewModels und die Visualisierungs–Plugins. Die Interfaces des Namensraums `PluginCommunication` müssen implementiert werden, um die Kommunikation des Hauptprogramms über die ViewModels mit den Plugins zu ermöglichen. Für die Kommunikation sind die Klassen des Namensraums `PluginCommunication` vorgesehen. Es gibt Klassen, die AOIs, AOI–Besuche, AOI–Transitionen, Schema–Klassen, Schema–Klassen–Besuche, Schema–Klassen–Besuchs–Transitionen,

[5] http://graphsharp.codeplex.com

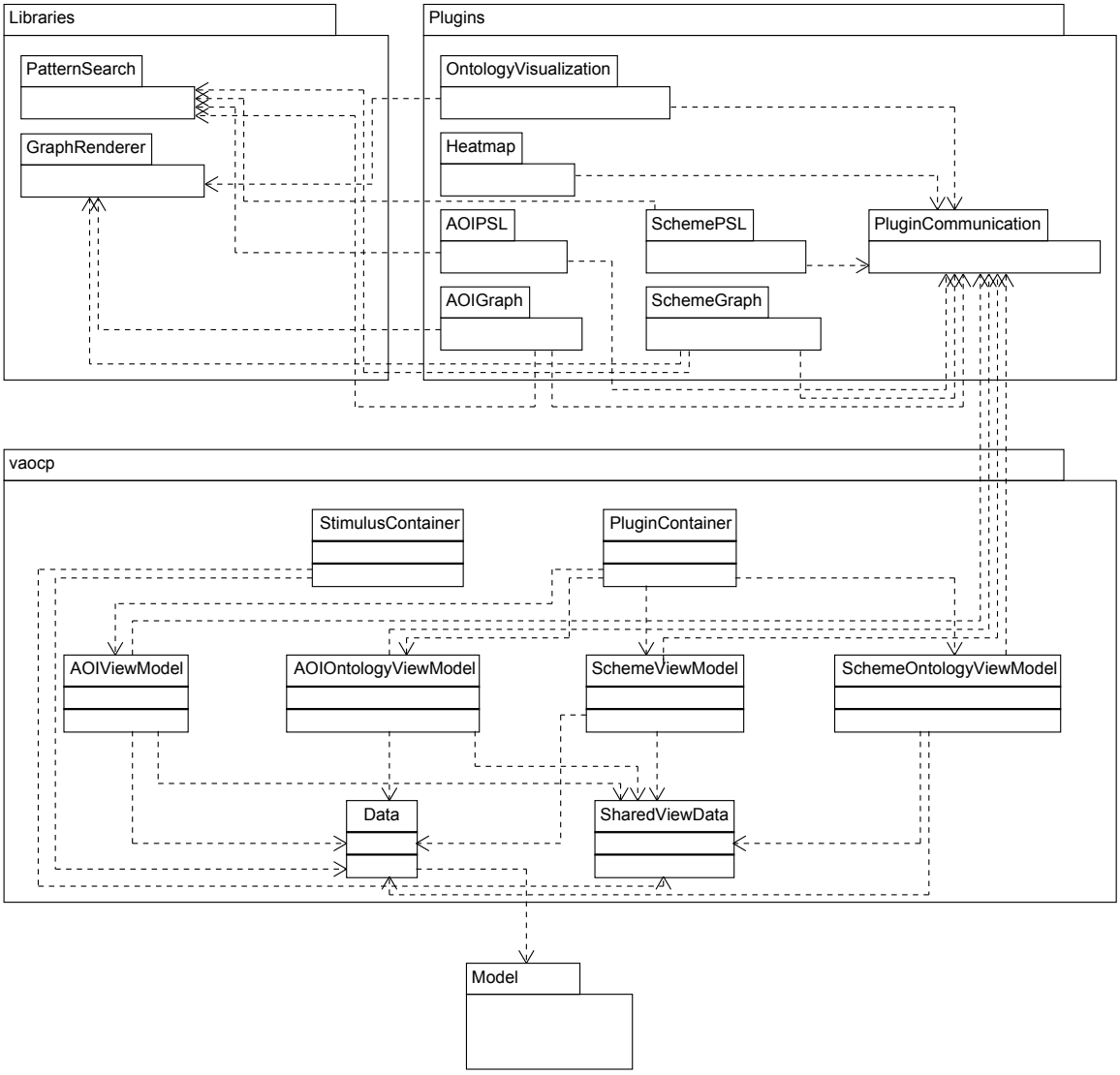

Abbildung 6.2: Die Architektur des Prototyps. Das Klassendiagramm zeigt die Architektur des Prototyps mit den wichtigsten Namensräumen und deren Unterteilung in weitere Namensräume. Weiterhin ist die Verteilung der Klassen über die Namensräume und die Abhängigkeiten der Namensräume und Klassen zu sehen. Die Abkürzung PSL steht für Pattern Search Lanes.

Probanden, Stimuli, Ontologie–Knoten und Ontologie–Kanten repräsentieren. Die Klassen des Datenbankschemas sind den Plugins nicht bekannt. Dadurch bleiben Plugins und Hauptprogramm voneinander unabhängig. Beide hängen nur vom Namensraum `PluginCommunication` ab. Der Namensraum `PluginCommunication` enthält alle Klassen und Interfaces, die erforderlich sind um neue Visualisierungs–Plugins für das Hauptprogramm zu entwickeln. Dadurch kann dieses einfach um neue Plugins erweitert werden.

Beim Namensraum `vaocp` handelt es sich um das Hauptprogramm. Das Akronym `vaocp` steht für den englischen Titel der vorliegenden Ausarbeitung: Visual Analysis of Cognitive Processes. Die Klasse `Data` lädt die Daten aus der Datenbank und stellt sie über Properties bereit. Die Klasse `SharedViewData` implementiert den Brushing & Linking Mechanismus. Die vier ViewModels greifen auf die Daten der `Data` Klasse zu, bereiten sie für die jeweilige Anzeige vor und stellen sie über Properties bereit. Die Klasse `StimulusContainer` enthält Methoden, um den Stimulus und seine zugehörigen Visualisierungen anzuzeigen. Die Klasse `PluginContainer` kann ein Visualisierungs–Plugin aufnehmen und anzeigen. Sie ermöglicht dem Plugin den Zugriff auf die Daten seines zugehörigen ViewModels.

Der Namensraum `Model` beinhaltet Klassen zur Kommunikation mit der Datenbank und die Klassen des Datenbankschemas. Mit Hilfe des Entity Frameworks werden die Instanzen der Klassen des Datenbankschemas verwaltet.

6.4 Plugins

Bei den Plugins handelt es sich um MEF Plugins. MEF steht für Managed Extensibility Framework und wird von .NET zur komfortablen Entwicklung und Einbindung von Plugins bereitgestellt. Die Verwendung von Plugins bietet den großen Vorteil, dass durch sie neue oder geänderte Funktionalität zum Hauptprogramm hinzugefügt werden kann, ohne dieses neu kompilieren zu müssen. Im Folgenden wird zunächst die Funktionsweise der MEF Plugins erläutert und anschließend die Klasse erklärt, die die Plugins aufnimmt sowie anzeigt. Abschließend werden die Klassen und Interfaces der `PluginCommunication` Komponente näher beschrieben.

6.4.1 Funktionsweise der Plugins

Die MEF Plugin Architektur erlaubt es, Klassen als Plugin zu definieren und zu exportieren. Diese Klassen können dann an anderer Stelle importiert und dort verwendet werden. Listing 6.1 zeigt einen Ausschnitt aus einer Klasse für ein Plugin. Im Listing sieht man, dass ein Plugin vom Typ `IAOIVisualizationPlugin` exportiert wird. Es handelt sich dabei um eine Klasse, die von UserControl ableitet. Dies bedeutet, dass sie in der Benutzeroberfläche verwendet werden kann. Die `PartCreationPolicy` gibt in diesem Fall an, dass jedes Mal, wenn dieses Plugin importiert wird, eine neue Instanz der Klasse erzeugt wird. In Listing 6.2 ist ein Ausschnitt einer Property zu sehen, die eine Liste von Plugins beinhaltet, die das Interface `IAOIVisualizationPlugin` implementieren. Durch die Annotation `ImportMany` wird angegeben, dass die Property mehrere Plugins aufnimmt. Die `RequiredCreationPolicy` gibt hier an, dass hier nur Plugins erlaubt sind, die bei ihrem Import eine neue Instanz erzeugen. Die Liste wird befüllt, in dem ein Befehl für den Import der Plugins ausgeführt

Listing 6.1 Ausschnitt aus einer Klasse, die ein Plugin vom Typ `IAOIVisualizationPlugin` exportiert. Es handelt sich dabei um eine von UserControl abgeleitete Klasse, die in der Benutzeroberfläche angezeigt werden kann.

```
...
[Export(typeof(IAOIVisualizationPlugin)), PartCreationPolicy(CreationPolicy.NonShared)]
public partial class AOIPatternSearchLanes : UserControl, IAOIVisualizationPlugin
{
   ...
```

Listing 6.2 Ausschnitt aus einer Property, die mehrere Plugins in einer Liste aufnimmt. Die Liste kann Plugins enthalten, die das Interface `IAOIVisualizationPlugin` implementieren.

```
...
[ImportMany(typeof(IAOIVisualizationPlugin), RequiredCreationPolicy = CreationPolicy.NonShared)]
public List<IAOIVisualizationPlugin> Plugins
{
   ...
```

wird. Dabei kann angegeben werden, aus welcher Quelle die Plugins importiert werden sollen. Als Quelle kann unter anderem ein Verzeichnis dienen.

6.4.2 Die Klasse `PluginContainer`

Diese Klasse verwaltet die Plugins. Sie enthält eine Property wie in Listing 6.2 beschrieben. Als Quelle für Plugins wird im Prototyp ein separates Verzeichnis verwendet, das ausschließlich Plugins beinhaltet. Der `PluginContainer` lädt automatisch alle Plugins, die in diesem Verzeichnis enthalten sind. Es kann angegeben werden, dass nur Plugins geladen werden, die ein bestimmtes Interface implementieren. Der `PluginContainer` selbst ist von UserControl abgeleitet, kann also in der Benutzeroberfläche verwendet werden. Er enthält eine ComboBox, die alle geladenen Plugins zur Auswahl bereitstellt. Durch die Auswahl eines Plugins in der ComboBox wird dieses Plugin dann angezeigt.

6.4.3 Interfaces und Klassen des Namensraums `PluginCommunication`

Die Visualisierungs–Plugins sind dazu vorgesehen mit ihnen die Analyse in den beiden Modi durchzuführen und ein Ergebnis zu erzeugen. Für den Prototyp bedeutet dies im Einzelnen, dass zuerst im Modus zur Verarbeitung auf AOI–Ebene pro Stimulus eine repräsentative Abfolge von AOIs im `AOIPSL` Visualisierungs–Plugin erstellt wird. Diese wird anschließend an das Hauptprogramm zurückgegeben. Dieses vollzieht den Wechsel auf die Schema–Klassen, von denen die AOIs der repräsentativen AOI Abfolge abgeleitet sind. Die Abfolge der Schema–Klassen wird dann an den Modus zur Verarbeitung auf Schema–Klassen–Ebene übergeben. Dort wird mit dem `SchemePSL` Visualisierungs–Plugin eine repräsentative Abfolge von Schema–Klassen erzeugt und letztlich ausgegeben. Um diese Vorgänge zu ermöglichen, müssen bestimmte Schnittstellen eingehalten werden, was durch entsprechende Interfaces sichergestellt wird.

Listing 6.3 Interface für ViewModels zur Anzeige von AOI Abfolgen. Dieses Interface muss von einem ViewModel implementiert werden, um ein Plugin mit den notwendigen Daten zur Visualisierung von AOI Abfolgen zu versorgen. Dazu gehören vier Listen und die Methode `SetAOIResultOrder`, die die repräsentative AOI Abfolge als Parameter entgegennimmt.

```
public interface IAOIVisualizationHost : IHost
{
   ...
   List<PluginAOI> PluginAOIs { get; }
   List<PluginAOITransition> PluginAOITransitions { get; }
   List<PluginAOIVisit> PluginAOIVisits { get; }
   List<PluginParticipant> PluginParticipants { get; }
   void SetAOIResultOrder(List<PluginAOI> pluginAOIs);
   ...
}
```

Listing 6.4 Interface für Plugins. Dieses Interface muss von einem Plugin implementiert werden, um vom Prototypen eingebunden werden zu können. Es enthält die Property `PluginName` für den Namen des Plugins, eine Methode `SetHost`, um dem Plugin eine Referenz auf sein ViewModel zu übergeben und die `Refresh` Methode zur Aktualisierung der grafischen Ansicht des Plugins.

```
public interface IPlugin
{
   string PluginName { get; }
   void SetHost(IHost host);
   void Refresh();
}
```

Die Interfaces und Klassen des Namensraums `PluginCommunication` stellen die Unabhängigkeit von Plugins und Hauptprogramm sicher, da sie deren einzige gegenseitige Abhängigkeit darstellen. Die Klassen sind dafür vorgesehen, die notwendigen Daten zwischen dem Hauptprogramm und den Plugins auszutauschen. Die Interfaces stellen die Einhaltung der Schnittstellen für den Datenaustausch sicher. Listing 6.3 zeigt einen Teil des Interfaces, dass die Klasse `AOIViewModel` implementiert. Es beinhaltet vier Listen, die ein Plugin benötigt, um AOI Abfolgen für verschiedene Probanden anzuzeigen. Weiterhin ist die Methode `SetAOIResultOrder` vorhanden, die die repräsentative AOI Abfolge entgegennimmt. Wie man im Listing sieht wird mit den Klassen des Namensraums `PluginCommunication` gearbeitet. Listing 6.4 zeigt das Interface für die Plugins. Es enthält eine Property `PluginName` die den Namen des Plugins aufnimmt. Dieser wird beispielsweise in der ComboBox des `PluginContainers` angezeigt. Über die `SetHost` Methode wird dem Plugin eine Referenz auf das entsprechende ViewModel übergeben. Die `Refresh` Methode dient der Aktualisierung der grafischen Ansicht des Plugins, etwa beim erneuten Laden durch das Auswählen in der ComboBox.

6.5 Bibliotheken

Die beiden Namensräume `PatternSearch` und `GraphRenderer` enthalten für den Prototyp entwickelte Bibliotheken, die zwar die speziellen Belange des Prototyps berücksichtigen, aber keine Abhängigkeit vom Prototyp aufweisen. Sie werden im Folgenden näher beschrieben.

6.5.1 PatternSearch

Die `PatternSearch` Bibliothek implementiert die exakte und die unscharfe Suche sowie die Berechnung der Levenshtein–Distanz zweier Zeichenketten. Für die Durchführung der exakten Suche wird ein Suchfenster wie im Konzept beschrieben über die Abfolge gelegt in der gesucht werden soll. Die unscharfe Suche wird ebenfalls mittels eines Suchfensters durchgeführt. Dabei wird pro Suchschritt die Levenshtein–Distanz der unscharf zu suchenden Abfolge mit dem Inhalt des Suchfensters berechnet. Die Einfüge– und Löschkosten betragen jeweils zwei, die Ersetzungskosten eins. Der Code zur Berechnung der Ergebnisse der unscharfen Suche für eine Bahn der Pattern Search Lanes ist in Listing 6.5 dargestellt. Im Code wird die Klasse `Node` verwendet, um AOIs oder Schema–Klassen zu repräsentieren und somit die Unabhängigkeit vom Hauptprogramm zu sichern.

6.5.2 GraphRenderer

Der `GraphRenderer` Bibliothek wird eine Menge von Knoten und Kanten übergeben. Daraufhin gibt sie ein UserControl zurück, dass den gerenderten Graphen enthält. Für den Prototyp war es erforderlich Knoten um einen anderen Knoten herum anordnen zu können. Dies ist mit der `GraphRenderer` Bibliothek möglich. Das initiale Layout für den anzuzeigenden Graphen ist die Kreisform. Dabei werden die Knoten des Graphen auf einem Kreis platziert. Die Implementierung ist so angelegt, dass die Klasse zur Berechnung des initialen Layouts ausgetauscht oder erweitert werden kann, um weitere Layoutverfahren hinzuzufügen. Sollte es durch das initiale Layout zu Verdeckungen kommen, so können die Knoten mit der Maus verschoben werden. Die neuen Knotenpositionen können gespeichert und geladen sowie das initiale Layout wiederhergestellt werden. Abbildung 6.3 zeigt einen Screenshot der in Abschnitt 6.7.2 beschriebenen Ontologie Visualisierung. Sie benutzt die `GraphRenderer` Bibliothek zur Anzeige eines Ontologie Graphen.

6.6 Vorbereitungen

Zur Identifikation und Annotation von AOIs wird das am Institut für Visualisierung und Interaktive Systeme entwickelte Programm cEdit verwendet. Um AOIs anhand einer Heatmap identifizieren zu können, wurde cEdit um die Funktionalität zur Einbindung eines Heatmap–Plugins erweitert. Für das Heatmap–Plugin wurde eine Heatmap Implementierung von Strohmaier verwendet [Str14]. Diese Implementierung wurde für die pluginbasierte Verwendung angepasst. Somit kann die Heatmap sowohl in cEdit als auch im Prototyp der vorliegenden Ausarbeitung als Plugin verwendet werden. Abbildung 6.4 zeigt cEdit mit der eingeblendeten Heatmap. Für die Erstellung und Bearbeitung der Ontologien kommt die Software Protégé[6] zum Einsatz.

Die Identifikation und Annotation von AOIs sowie das Erstellen der AOI–Ontologie sind ein zusammenhängender Prozess. Die AOIs werden in cEdit anhand der Heatmap identifiziert und annotiert. Bevor die Ausgabe von cEdit in Form einer sogenannten Motif–Datei erstellt werden kann, muss

[6] http://protege.stanford.edu

Listing 6.5 Code zur Berechnung der Ergebnisse der unscharfen Suche. Der Code benutzt ein Suchfenster, dass über die Abfolge gelegt wird, in der gesucht werden soll und berechnet die Levenshtein–Distanz zwischen der unscharf zu suchenden Abfolge und dem Suchfensterinhalt.

```
// Fuehre fuer jeden Knoten in der Bahn aus
for (int i = 0; i < laneNodes.Count(); i++)
{
   // Initialisiere eine Liste von Knoten im Suchfenster
   List<Node> windowNodes = new List<Node>();
   // Initialisiere die Zeichenkette innerhalb des Suchfensters
   string windowNodesString = String.Empty;
   // Initialisiere eine Variable fuer die Suchfensterlaenge
   int windowLength = 0;
   // Erzeuge ein Suchfenster von der Laenge der Anzahl der markierten Knoten
   // und der zusaetzlichen Suchfensterlaenge.
   // Fuehre fuer jeden Knoten im Suchfenster aus
   for (int k = 0; k < markedNodes.Count + additionalWindowLength; k++)
   {
      // Fahre mit naechster Iteration fort, wenn das Suchfenster am Ende der Bahn angelangt ist
      if (laneNodes.Count() <= i + k) { continue; }
      windowLength++;
      // Fuege den Buchstaben des aktuellen Knotens zur Zeichenkette des Suchfensters hinzu
      windowNodesString += laneNodes.ElementAt(i + k).LevenshteinChar;
      // Fuege den aktuellen Knoten zur Liste von Knoten im Suchfenster hinzu
      windowNodes.Add(laneNodes.ElementAt(i + k));
   }
   // Wenn die Suchfensterlaenge der Anzahl der markierten Knoten und der
   // zusaetzlichen Suchfensterlaenge entspricht, dann fuege zum Ergebnis der Suche
   // die Liste der Knoten im Suchfenster (mitsamt ihrer Levenshtein-Distanz zu den
   // markierten Knoten) hinzu.
   if (windowLength == markedNodes.Count + additionalWindowLength)
   {
      result.Add(new LevenshteinNodes()
      {
         Costs = LevenshteinCalculator.calculateDistance(windowNodesString, markedNodesString),
         Nodes = windowNodes.ToList(),
         ...
      });
   }
}
```

allen AOIs jeweils die URI ihrer Ontologie Klasse in cEdit zugewiesen werden. Dazu wird parallel zur Arbeit mit cEdit die AOI–Ontologie mit Protégé erstellt.

Die Anreicherung der AOI–Klassen mit Informationen geschieht mit Hilfe von Annotation Properties. Listing 6.6 zeigt einen Auschnitt aus einer Datei im RDF/XML Format, die die AOI–Ontologie zu einem Balkendiagramm enthält. Man sieht die Definition einer Klasse für die AOI, die dem Bereich des X–Werts des Balkens mit der Nummer 5 zugeordnet ist. Weiterhin ist zu erkennen, dass diese Klasse von der Schema–Klasse `XAchse` abgeleitet ist. Das `Gewicht` wird mit `+1` definiert und die Orientierungsrelationen `nachOben` und `nachLinks` sind angegeben.

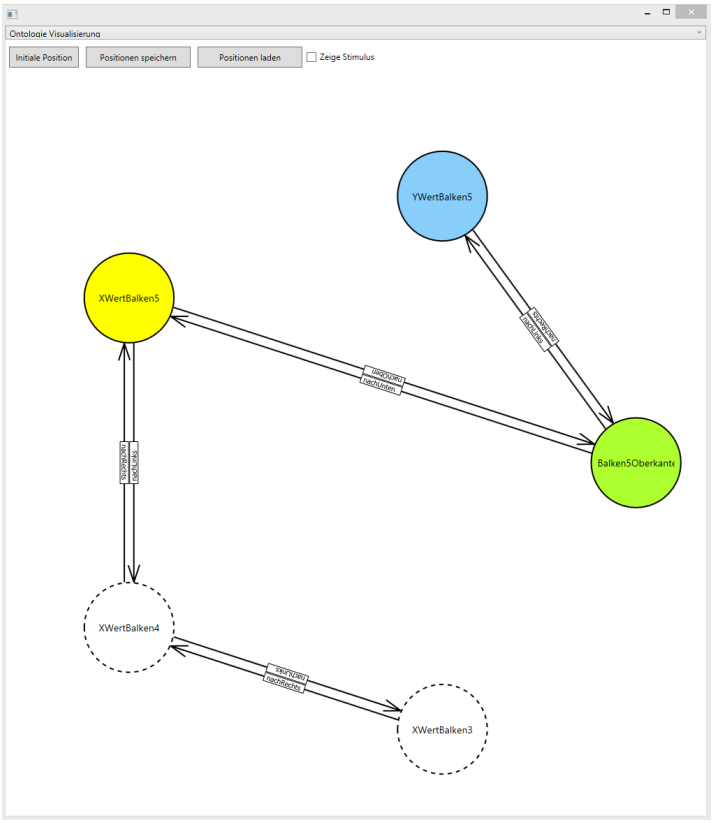

Abbildung 6.3: Screenshot der GraphRenderer Bibliothek. Der Screenshot zeigt den Graphen einer Ontologie der Ontologie–Visualisierung. Die Ontologie–Visualisierung benutzt die GraphRenderer Bibliothek zur Anzeige eines Graphen.

6.7 Benutzeroberfläche

Dieser Abschnitt beschreibt die Benutzeroberfläche des Daten Managements, des Modus zur Verarbeitung auf AOI–Ebene und des Modus zur Verarbeitung auf Schema–Klassen–Ebene. Dabei werden die einzelnen Bestandteile der Benutzeroberfläche eines jeden Modus detailliert erläutert. Beide Modi sind aus mehreren Fenstern aufgebaut, die jeweils ausgeklinkt und bildschirmfüllend dargestellt werden können. Der Inhalt eines Fensters passt sich dabei automatisch an die Größe des Fensters an. Erreicht wird dies durch den Einsatz der vom .NET Framework bereitgestellten Klasse `Viewbox`, die den zu skalierenden Fensterinhalt enthält.

6.7.1 Daten Management

Abbildung 6.5 zeigt die Oberfläche des Daten Managements. Sie besteht im Wesentlichen aus zwei Regionen. Die obere Region (R1) *Neue Daten* dient dazu, neue Daten in eine leere Datenbank zu importieren. In der unteren Region (R2) *Vorhandene Daten* können Daten aus einer bereits bestehenden

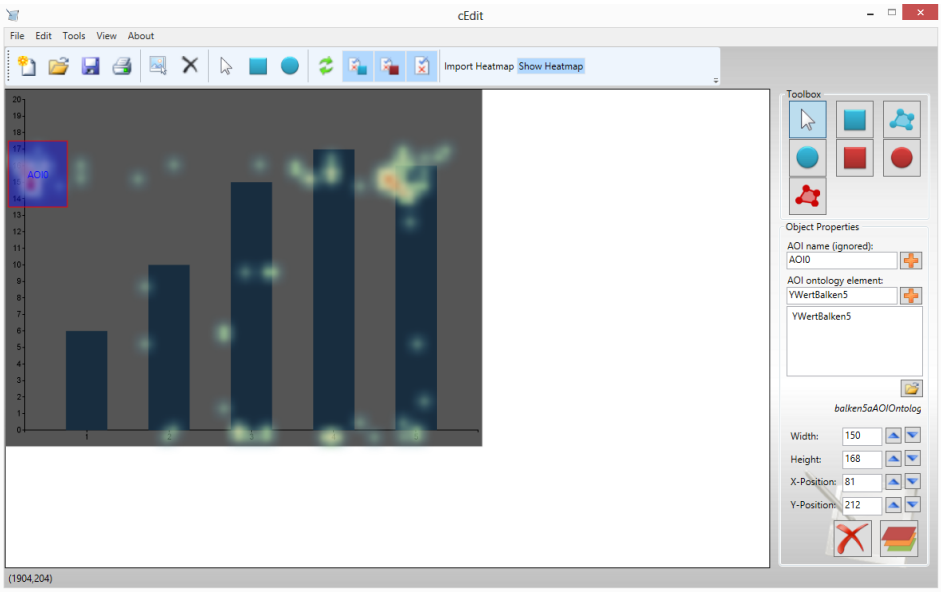

Abbildung 6.4: cEdit mit Heatmap Plugin. Zur Identifikation und Annotation von AOIs wurde das Programm cEdit um die Anzeige eines Heatmap Plugins erweitert.

Listing 6.6 Ausschnitt aus einer Datei im RDF/XML Format. Die Datei enthält die AOI–Ontologie zu einem Balkendiagramm. Es wird eine Klasse für die AOI `XWertBalken5` definiert. Die AOI ist von der Schema–Klasse `XAchse` abgeleitet. Sie erhält das Gewicht +1 und zwei Orientierungsrelationen `nachOben` und `nachLinks`.

```
...
<owl:Class rdf:about="http://www.semanticweb.org/vaocp/balken5a.pngAOIOntologie#XWertBalken5">
   <rdfs:subClassOf rdf:resource=
    "http://www.semanticweb.org/vaocp/BalkendiagrammVisualisierungsSchemaOntologie#XAchse"/>
   <VisualisierungsElementeOntologie:Gewicht>+1</VisualisierungsElementeOntologie:Gewicht>
   <VisualisierungsElementeOntologie:nachOben rdf:resource=
    "http://www.semanticweb.org/vaocp/balken5a.pngAOIOntologie#Balken5Oberkante"/>
   <VisualisierungsElementeOntologie:nachLinks rdf:resource=
    "http://www.semanticweb.org/vaocp/balken5a.pngAOIOntologie#XWertBalken4"/>
</owl:Class>

<owl:Class rdf:about="http://www.semanticweb.org/vaocp/balken5a.pngAOIOntologie#XWertBalken4">
   ...
   ...
</owl:Class>
...
```

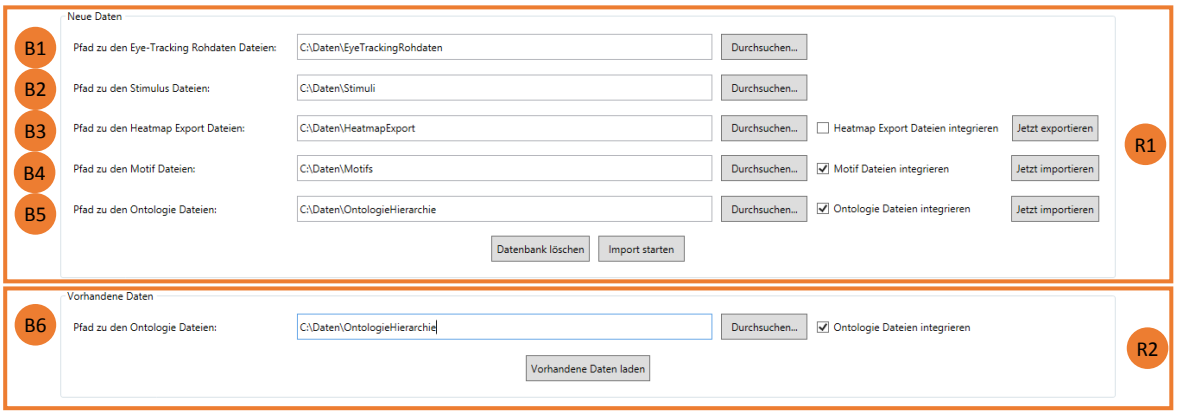

Abbildung 6.5: Benutzeroberfläche des Daten Managements. Die Benutzeroberfläche teilt sich in zwei Regionen. (R1) bietet Funktionen zum Import von Daten in eine leere Datenbank. In Region (R2) können Daten aus einer bestehenden Datenbank geladen werden. (B1) erlaubt die Festlegung des Pfads zu den Eye–Tracking–Daten. (B2) legt den Pfad zu den Stimuli fest. Über (B3) werden die Daten für die Heatmap für das Programm cEdit exportiert. (B4) enthält den Pfad zu den Motif Dateien. Mit (B5) und (B6) wird der Pfad zur Ontologie–Hierarchie eingegeben, einmal für den Import in eine leere Datenbank (B5) und einmal zum Laden von Daten aus einer bestehenden Datenbank.

Datenbank geladen werden. Dadurch muss nicht bei jedem Start des Prototyps ein Import durchgeführt werden.

Es folgt nun die Erläuterung der einzelnen Bereiche. Im Bereich (B1) wird der Pfad zu den Rohdaten des Eyetrackers festgelegt. Bereich (B2) erlaubt die Definition des Pfades zu den Stimulus Dateien. Über Bereich (B3) wird der Export der Heatmap ermöglicht. Dabei wird eine XML–Datei erzeugt, die die Positionen der Fixationen enthält. Diese Datei wird von dem Programm cEdit eingelesen, um die Heatmap anzeigen zu können. Über die Schaltfläche *Jetzt exportieren* kann der Export sofort angestoßen werden, unabhängig vom Import der Rohdaten. Durch Bereich (B4) wird der Pfad zu den von cEdit erzeugten Motif Dateien angegeben. Durch das Drücken der Schaltfläche *Jetzt importieren* kann die Motif Datei sofort importiert werden ohne die Rohdaten zu importieren. Schließlich wird im Bereich (B5) der Pfad zu den Dateien der Ontologie Hierarchie festgelegt. Die Betätigung der Schaltfläche *Jetzt importieren* führt dazu, dass die Ontologie Hierarchie unabhängig vom Import der Rohdaten sofort zu den bestehenden Daten hinzugefügt wird. Durch das Setzen eines Hakens in den Checkboxen in den Bereichen (B3), (B4) und (B5) wird bestimmt, dass die jeweiligen Dateien beim Rohdaten Import miteinbezogen werden. In Bereich (B6) wird der Pfad zu den Dateien der Ontologie–Hierarchie bestimmt. Durch das Setzen eines Hakens in der Checkbox wird die Ontologie–Hierarchie beim Laden von Daten aus einer bestehenden Datenbank miteinbezogen.

6.7.2 Modus zur Verarbeitung auf AOI–Ebene

Der folgende Abschnitt erklärt die Benutzeroberfläche des Modus zur Verarbeitung auf AOI–Ebene. Es wird beschrieben auf welche Weise Informationen zu AOIs angezeigt werden, stellt die Stimulus– und Ontologie–Visualisierung vor und erläutert schließlich wichtige Funktionen der Pattern Search Lanes und des Graphs.

Oberfläche

Abbildung 6.6 zeigt die Oberfläche des Modus zur Verarbeitung auf AOI–Ebene. Durch Betätigen der Schaltfläche (S1) wird das Daten Management geöffnet. Die Betätigung der Schaltfläche (S2) führt dazu, dass in den Modus zur Verarbeitung auf Schema–Klassen–Ebene gewechselt wird. In Bereich (B1) kann anhand der Tabs zwischen den verschiedenen Stimuli gewechselt werden. Bereich (B2) beinhaltet die Stimulus–Visualisierung. In Bereich (B3) sind die Pattern Search Lanes enthalten. In der Abbildung sind die Haken in den beiden Checkboxen *Gleiche Größe* und *Gleicher Abstand* gesetzt, wodurch alle AOI–Besuche in den Bahnen denselben Durchmesser und einen einheitlichen Abstand voneinander haben. Bereich (B4) beinhaltet die Ergebniszusammenstellung, mit der AOI Quelle in Bereich (B5) und der AOI Ablage in Bereich (B6). Die Ontologie Visualisierung ist in Bereich (B7) zu sehen. Der Inhalt von Bereich (B8) ist der Graph.

Informationen zu AOIs

Bei Aufenthalt des Mauszeigers innerhalb einer AOI oder eines AOI–Besuchs wird das Gewicht der AOI sowie die URIs ihrer Vaterknoten in der Ontologie–Hierarchie angezeigt. Die für die Anzeige der Informationen verantwortliche Klasse ist einmal an zentraler Stelle implementiert und steht allen Visualisierungs–Plugins zur Verfügung. Dadurch wird sichergestellt, dass die angezeigten Informationen für jedes Plugin denselben Inhalt und dasselbe Aussehen haben.

Stimulus–Visualisierung

Die Stimulus–Visualisierung beherrscht vier Ansichten und wird durch die Klasse `StimulusContainer` implementiert. Die vier Ansichten sind in Abbildung 6.7 dargestellt. Es können Gazepoints (1), Fixationen (2), eine Heatmap (3) und Scanpfade (4) angezeigt werden. Bei der Heatmap handelt es sich um das Heatmap Plugin, das auch im Programm cEdit verwendet wird. Es kann ausgewählt werden von welchen Probanden Daten angezeigt werden sollen und ob AOIs eingeblendet werden sollen oder nicht. Durch die Auswahl einer AOI kann ihre Farbe festgelegt und in der Datenbank gespeichert werden. Die Farbe der AOI wird per Brushing & Linking mit den Visualisierungen in den anderen Fenstern dieses Modus synchronisiert. Durch die Speicherung der Farbe in der Datenbank erhält die AOI beim nächsten Start des Prototyps wieder die ihr zugewiesene Farbe.

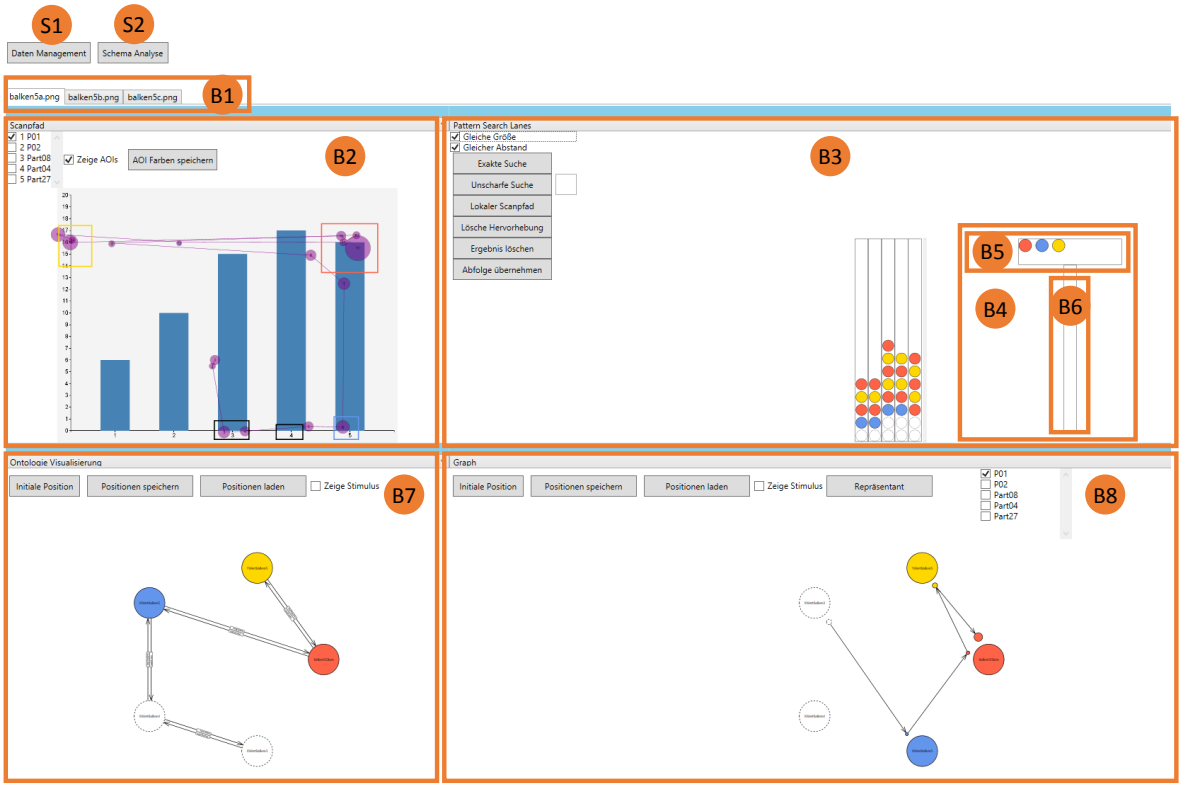

Abbildung 6.6: Benutzeroberfläche zur Verarbeitung auf AOI–Ebene. (S1) Schaltfläche zum Öffnen des Daten Managements, (S2) Wechsel in den Modus zur Verarbeitung auf Schema–Klassen–Ebene, (B1) Wechsel zwischen den Stimuli, (B2) Stimulus–Visualisierung, (B3) Pattern Search Lanes, (B4) Ergebniszusammenstellung, (B5) AOI Quelle der Ergebniszusammenstellung, (B6) AOI Ablage der Ergebniszusammenstellung, (B7) Ontologie–Visualisierung der AOI–Ontologie und (B8) Graph.

Ontologie–Visualisierung

Bereich (B7) in Abbildung 6.6 zeigt die Ontologie Visualisierung. Sie wird als Plugin im Namensraum `OntologyVisualization` implementiert. Sie benutzt die `GraphRenderer` Bibliothek zur Darstellung des Graphen. Im Hintergrund kann der Stimulus eingeblendet werden.

Pattern Search Lanes

Die Pattern Search Lanes sind als Plugin im Namensraum `AOIPSL` implementiert und nutzen die `PatternSearch` Bibliothek zur Durchführung der exakten und der unscharfen Suche. Die exakte und die unscharfe Suche sind insbesondere zur Suche von Mustern in langen und auch vielen AOI Abfolgen vorgesehen. Dazu sind die Bahnen mit einer vertikalen Scrollbar ausgestattet, damit auch lange AOI Abfolgen untersucht werden können. Die Pattern Search Lanes beinhalten im Prototyp die

Abbildung 6.7: Die Stimulus–Visualisierung. Die Stimulus–Visualisierung in der Ansicht Gazepoints (1), Fixationen (2), Heatmap (3) und Scanpfad (4). Die Ansicht kann mittels einer Combobox gewechselt werden.

Ergebniszusammenstellung, durch die von Hand per Drag & Drop eine repräsentative AOI Abfolge zusammengestellt werden kann. Dazu werden in Abbildung 6.6 AOIs mit der Maus von Bereich (B5) nach Bereich (B6) gezogen. Dabei stehen negativ gewichtete AOIs nicht zur Zusammenstellung zur Verfügung. Die Betätigung einer Schaltfläche führt dazu, dass die Abfolge im Bereich (B6) in den Modus zur Verarbeitung auf Schema–Ebene übernommen wird. Abbildung 6.8 zeigt die Benutzeroberfläche der Pattern Search Lanes bei der Verwendung der exakten Suche. In Bereich (1) werden die zu suchenden AOIs mit der Maus markiert und dadurch schwarz umrandet hervorgehoben. Nach Durchführung der Suche sind in Bereich (2) die exakt gleichen Abfolgen von AOI–Besuchen rot umrandet hervorgehoben. Die Implementierung des lokalen Scanpfads ist in Abbildung 6.9 dargestellt. In den Pattern Search Lanes (1) wird in Bereich (2) ein AOI–Besuch mit der Maus markiert und dadurch schwarz umrandet hervorgehoben. Sein Vorgänger wie auch sein Nachfolger werden rot umrandet hervorgehoben. In der Stimulus–Visualisierung in Bereich (3) der Abbildung wird der

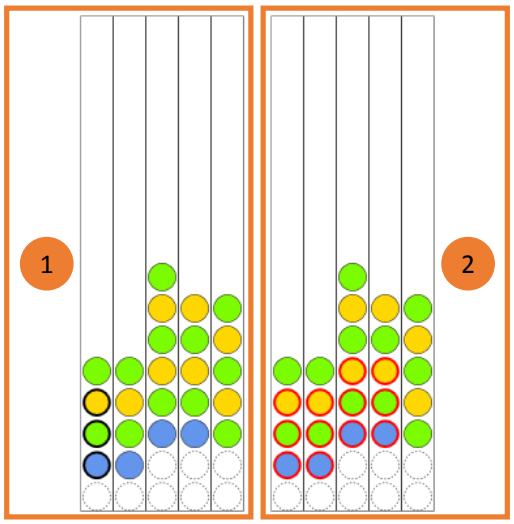

Abbildung 6.8: Pattern Search Lanes mit exakter Suche. (1) enthält die exakt zu suchende Abfolge. Die Suchergebnisse sind in (2) rot umrandet hervorgehoben.

zugehörige Abschnitt des Scanpfads mit allen Fixationen beginnend mit der ersten Fixation in der blauen AOI (A1) über die grüne AOI (A2) bis hin zur letzten Fixation in der gelben AOI (A3) in der Farbe rot angezeigt. Der Rest des Scanpfads ist violett.

Die Abbildung 6.10 zeigt die Benutzeroberfläche der Implementierung der unscharfen Suche. Für die folgende Beschreibung betragen die Einfüge– und Löschkosten jeweils zwei sowie die Ersetzungskosten eins. In der Abbildung werden in Bahn 01 im Bereich (1) drei AOI–Besuche mit der Maus ausgewählt und dadurch schwarz umrandet hervorgehoben. Es wird eine Verlängerung des Suchfensters um 1 festgelegt. Somit ergibt sich eine Suchfensterlänge von 4. In Bereich (2) sind die Ergebnisse pro Bahn mit ihren jeweiligen Kosten aufgelistet. Für Bahn 02 ist ein Ergebnis mit den Kosten 2 ausgewählt. Die zugehörigen AOI–Besuche werden in Bahn 02 rot umrandet hervorgehoben. Die Ergebnis Abfolge hat also die Levenshtein–Distanz 2 zur markierten Abfolge. Für Bahn 05 ist ein Ergebnis mit den Kosten 3 ausgewählt und die zugehörigen AOI–Besuche werden in der zugehörigen Bahn rot umrandet hervorgehoben. In diesem Fall hat die markierte AOI Abfolge die Levenshtein–Distanz 3 zur Ergebnis Abfolge.

Graph

Der Graph ist als Plugin im Namensraum `AOIGraph` implementiert und nutzt die `GraphRenderer` Bibliothek zur Darstellung des enthaltenen Graphen. Er ist in Bereich (B8) der Abbildung 6.6 dargestellt. Es kann ein Proband ausgewählt werden dessen AOI Abfolge dann dargestellt wird. Im Hintergrund kann der Stimulus angezeigt werden. Der Repräsentant wird anhand der Methode *Auswahl einer Abfolge aus einer Menge von Abfolgen* (siehe Abschnitt 5.6.2) berechnet. Durch die Betätigung einer Schaltfläche wird der Repräsentant berechnet und der Name des Probanden angezeigt, dessen AOI Abfolge dem Repräsentanten entspricht.

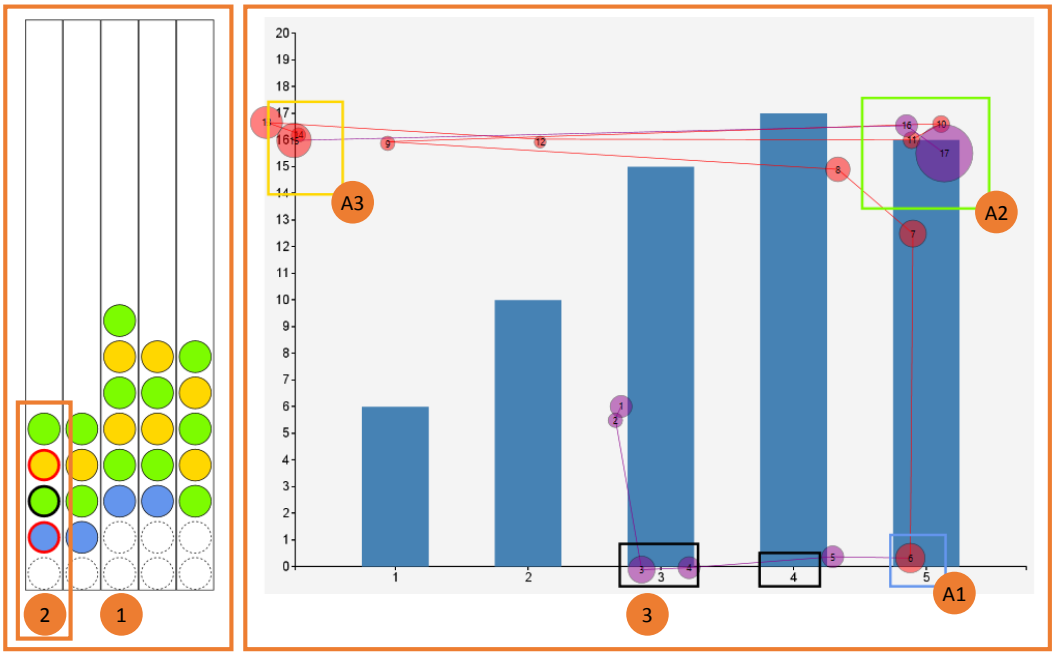

Abbildung 6.9: Der lokale Scanpfad. In den Pattern Search Lanes (1) wird ein AOI–Besuch ausgewählt. Dadurch werden der AOI–Besuch selbst und sein Vorgänger und sein Nachfolger hervorgehoben (2). Durch Brushing & Linking wird in der Scanpfad Ansicht der Stimulus–Visualisierung der zur hervorgehobenen AOI Abfolge gehörende Abschnitt des Scanpfads in der Farbe Rot dargestellt.

6.7.3 Modus zur Verarbeitung auf Schema–Klassen–Ebene

Abbildung 6.11 zeigt die Oberfläche des Modus zur Verarbeitung auf Schema–Klassen–Ebene. Bereich (B1) enthält die Visualisierung der Visualisierungs–Elemente–Ontologie. Bereich (B2) umfasst die Pattern Search Lanes. Sie enthalten die Ergebniszusammenstellung in Bereich (B3). Bereich (B4) beinhaltet die Darstellung der Visualisierungs–Schema–Ontologie. Der Graph ist in Bereich (B5) enthalten. Durch Betätigung der Schaltfläche (S1) wird in den Modus zur Verarbeitung auf AOI–Ebene gewechselt. Die Visualisierungen und Mechanismen im Modus zur Verarbeitung auf Schema–Klassen–Ebene funktionieren weitgehend analog zu den Visualisierungen und Mechanismen im Modus zur Verarbeitung auf AOI–Ebene. Vorhandene Besonderheiten und Ausnahmen werden im Folgenden erläutert.

Für die Anzeige von Informationen zu den Schema–Klassen und Schema–Klassen–Besuchen wird wie im Modus zur Verarbeitung auf AOI–Ebene eine Klasse genutzt, die an zentraler Stelle implementiert ist und allen Plugins zur Verfügung steht.

Die beiden Ontologie–Visualisierungen werden jeweils mit Hilfe des `OntologyVisualization` Plugins dargestellt. Das Plugin nutzt die `GraphRenderer` Bibliothek zur Anzeige des Ontologie–Graphen. Bei den beiden Ontologie–Visualisierungen gilt die Ausnahme, dass, im Gegensatz zur Ontologie–

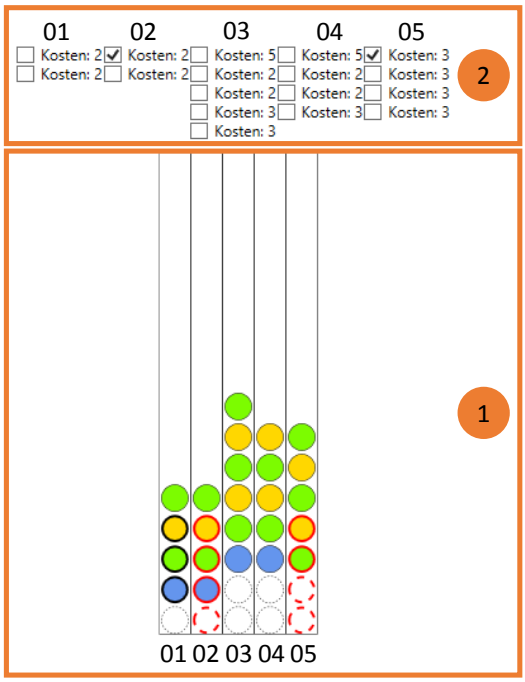

Abbildung 6.10: Pattern Search Lanes mit unscharfer Suche. Die Abfolge von AOI Besuchen in Bahn 01 der Pattern Search Lanes (1) wird in den anderen Bahnen unscharf gesucht. Die Suchergebnisse werden mit ihren Kosten für jede Bahn zur Anzeige bereitgestellt (2). In Bahn 02 im Bereich (1) wird ein Suchergebnis mit den Kosten 2 hervorgehoben und in Bahn 05 im Bereich (1) wird ein Suchergebnis mit den Kosten 3 hervorgehoben.

Visualisierung im Modus zur Verarbeitung auf AOI–Ebene, kein Stimulus im Hintergrund angezeigt werden kann. Die Zuweisung von Farben an die Schema–Klassen erfolgt durch die Ontologie–Visualisierung der Visualisierungs–Schema–Ontologie.

Der Graph ist als Plugin im Namensraum `SchemeGraph` implementiert und nutzt die `GraphRenderer` Bibliothek. Es kann kein Stimulus im Hintergrund eingeblendet werden. Die Berechnung eines Repräsentanten erfolgt durch die Methode *Auswahl einer Abfolge aus einer Menge von Abfolgen* (siehe Abschnitt 5.6.2) und gibt den Dateinamen eines Stimulus Bilds als Ergebnis aus.

Die Implementierung der Pattern Search Lanes erfolgt als Plugin im Namensraum `SchemePSL`. Das Plugin nutzt die `Pattern Search` Bibliothek zur Umsetzung der exakten und der unscharfen Suche. Die Funktionalität zur Anzeige des lokalen Scanpfads ist in diesem Modus nicht vorhanden, genauso wie die Funktionalität zur Übernahme einer repräsentativen Abfolge in die weitere Verarbeitung fehlt. Die URIs der repräsentativen Schema–Klassen Abfolge, also die Lesevorschrift, können nach Betätigung einer Schaltfläche in einer Listbox dargestellt werden. Durch Betätigung einer weiteren Schaltfläche wird eine Textdatei mit den URIs erstellt. Über eine Checkbox kann ausgewählt werden, ob die URIs in ihrer langen oder in ihrer kurzen Form in die Textdatei eingetragen werden sollen.

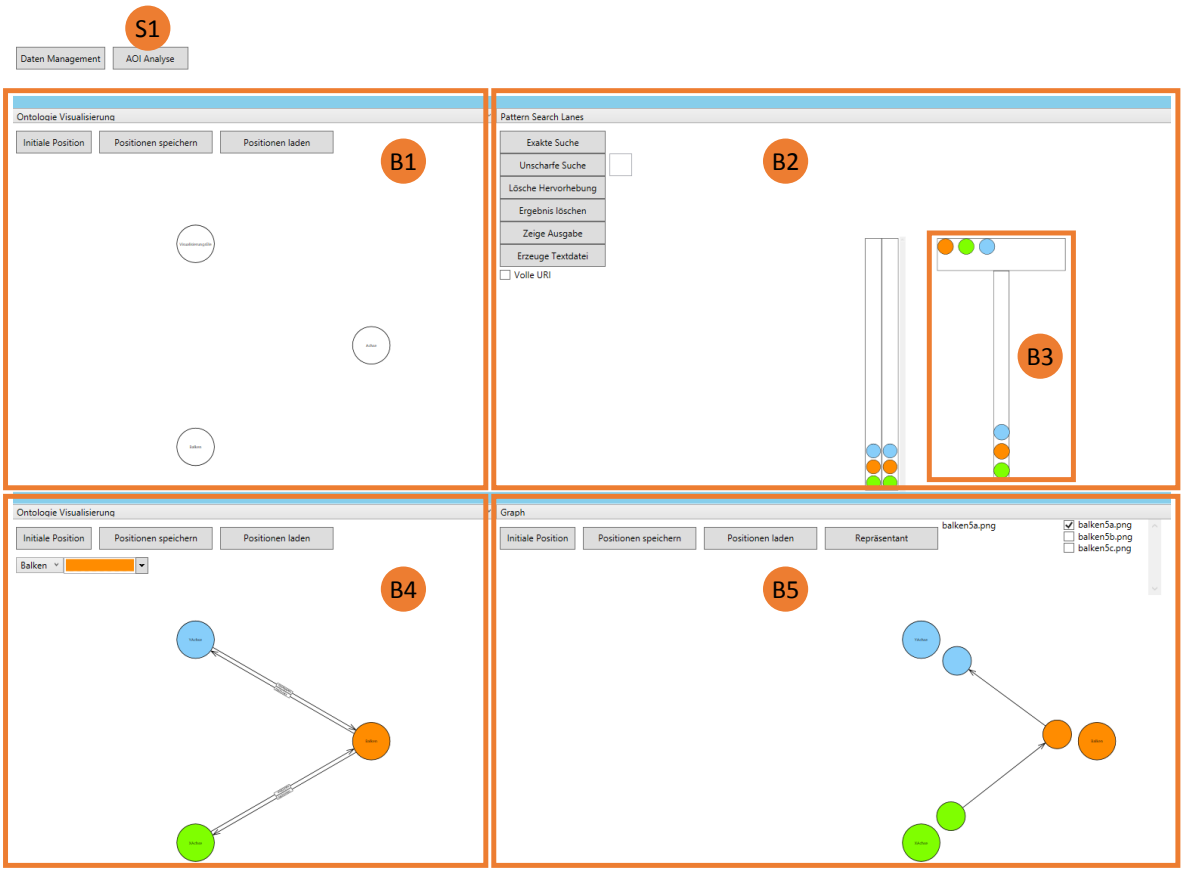

Abbildung 6.11: Benutzeroberfläche des Modus zur Verarbeitung auf Schema–Klassen–Ebene. (S1) Schaltfläche zum Wechsel in den Modus zur Verarbeitung auf AOI–Ebene, (B1) Ontologie–Visualisierung der Visualisierungs–Elemente–Ontologie, (B2) Pattern Search Lanes, (B3) Ergebniszusammenstellung, (B4) Ontologie–Visualisierung der Visualisierungs–Schema–Ontologie und (B5) Graph.

Kurze Form bedeutet, dass beispielweise im Fall der Lesevorschrift aus Listing 5.1 jeweils nur die Zeichenkette nach dem „#“ in die Textdatei geschrieben würde.

7 Demonstration

Die Funktionsweise des Prototyps wird in diesem Kapitel anhand zweier Szenarien demonstriert. Dabei werden aus den Eye–Tracking–Daten zweier Studien zwei Lesevorschriften für zwei Kombinationen aus Visualisierungstyp und Aufgabe erstellt. Als Datenquellen dienen dafür zwei am Institut für Visualisierung und Interaktive Systeme durchgeführte Eye–Tracking–Studien.

Abschnitt 7.1 schildert die Demonstration des Prototypen anhand der Gewinnung einer Lesevorschrift für den Visualisierungstyp Balkendiagramm. In Abschnitt 7.2 wird die Erzeugung einer Lesevorschrift für Flächendiagramme erläutert. Abschnitt 7.3 beschreibt offene Fragestellungen zum Visualisierungstyp Sudoku und Kuchendiagramm.

7.1 Erzeugung einer Lesevorschrift für Balkendiagramme

Der folgende Abschnitt beschreibt die Erzeugung einer Lesevorschrift für Balkendiagramme mit Hilfe des implementierten Prototyps. Die Lesevorschrift gilt für die Aufgabe, die Höhe eines bestimmten Balkens abzulesen. Für die Erzeugung der Lesevorschrift werden die Daten der am Institut für Visualisierung und Interaktive Systeme durchgeführten Eye–Tracking Studie „Visual Elements III“ verwendet. Es werden zunächst die Studiendaten näher erläutert sowie die notwendigen Vorbereitungen geschildert. Schließlich wird die Erzeugung der Lesevorschrift durch den Prototyp beschrieben.

7.1.1 Verwendete Daten und Stimuli

Bei der Studie „Visual Elements III“ wurden die kognitiven Prozesse bei der Durchführung von Aufgaben mit Visualisierungen untersucht. Es haben zehn Probanden an der Studie teilgenommen, deren Durchschnittsalter bei 29,2 Jahren lag. Vier der Teilnehmer waren weiblich und sechs Teilnehmer waren männlich. Zwei Probanden gaben an Vorkenntnisse im Bereich der Visualisierung zu besitzen. In der Studie wurden verschiedene Arten von Aufgaben gestellt. Die Aufgabenarten waren im Einzelnen:

1. Das Lösen einfacher Rechenaufgaben
2. Das Bearbeiten von Fragestellungen zu Mengendiagrammen
3. Das Lösen von Rätseln der Art Sudoku
4. Das Ablesen von Werten aus Balkendiagrammen
5. Das Ablesen von Werten aus verschiedenartigen Diagrammen

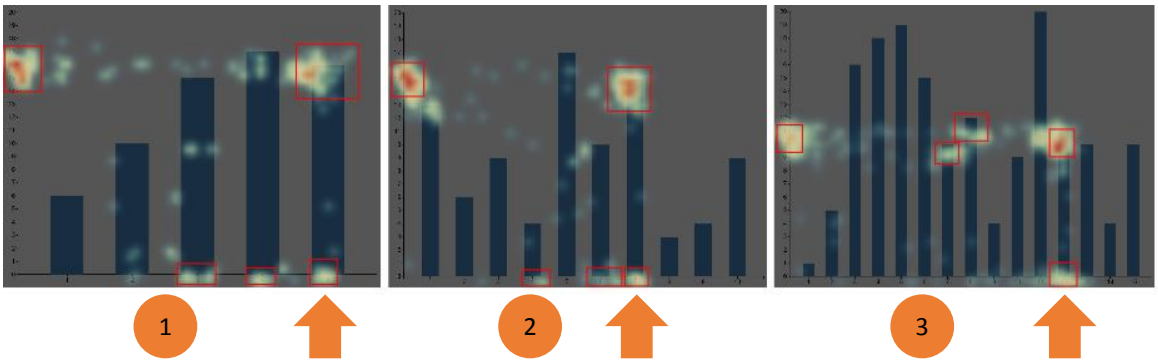

Abbildung 7.1: Die drei Balkendiagramm Stimuli. Auf den Stimuli soll jeweils der Wert des mit einem Pfeil markierten Balkens abgelesen werden. Auf den Stimuli werden mittels einer Heatmap mit cEdit die AOIs identifiziert und annotiert.

6. Das Vergleichen von Bildern

Die Studie wurde mit einem Eyetracker des Typs Tobii T60 XL durchgeführt. Der Bildschirm des Eyetrackers hat eine Auflösung von 1920x1200 Pixeln und eine Diagonale von 24 Zoll. Der Eyetracker nimmt die Blickpunkte mit einer Frequenz von 60 Hz auf.

Für die Erzeugung der Lesevorschrift wurden drei Stimuli ausgewählt. Alle drei zeigen jeweils ein Balkendiagramm, eines mit fünf, eines mit zehn und eines mit 15 Balken. Die Aufgabe lautet jeweils den Wert eines bestimmten Balkens abzulesen. Die Diagramme sind mit Heatmap und annotierten AOIs in Abbildung 7.1 dargestellt. Die Beschriftung der Balken wird Kategorie genannt und ist durch eine mit eins beginnende aufsteigende Nummerierung entlang der X–Achse in das Diagramm eingetragen. In der Abbildung ist der Balken, dessen Wert abgelesen werden soll, durch einen Pfeil markiert. In der eigentlichen Studie ist dieser Pfeil nicht vorhanden. Die Aufgaben lauten im Einzelnen:

- Für Stimulus (1): Bitte geben Sie den Wert der Kategorie fünf an.
- Für Stimulus (2): Bitte geben Sie den Wert der Kategorie sieben an.
- Für Stimulus (3): Bitte geben Sie den Wert der Kategorie zwölf an.

7.1.2 Vorbereitung

Zu Beginn werden mit Protégé die Visualisierungs–Elemente–Ontologie und die Visualisierungs–Schema–Ontologie für Balkendiagramme erstellt. Da die Visualisierungs–Elemente–Ontologie neu erstellt wird, enthält sie ausschließlich die in Balkendiagrammen vorkommenden visuellen Elemente. Diese umfassen die Klasse *Achse* und die Klasse *Balken*. Die Visualisierungs–Schema–Ontologie ist in Abbildung 7.2 dargestellt. Nachdem die Ontologien erstellt wurden, werden nun die Eye–Tracking–Daten und Stimuli in den Prototyp importiert. Dies geschieht zunächst nur, um die Ausgabe zu erzeugen, die für die Anzeige der Heatmap im Programm cEdit nötig ist. Nach Erzeugung der Daten für die Heatmap werden diese in cEdit importiert. Danach werden anhand der Heatmap die AOIs

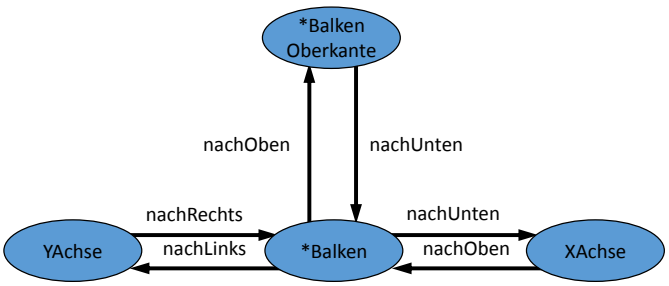

Abbildung 7.2: Visualisierungs–Schema–Ontologie für Balkendiagramme. Die Ontologie definiert die in Balkendiagrammen vorkommenden visuellen Elemente. Klassen, die mit einem (*) markiert sind, können beliebig oft in einem Balkendiagramm auftreten.

identifiziert und annotiert. Abbildung 7.1 zeigt alle drei Stimuli mit eingeblendeter Heatmap und annotierten AOIs. Nachdem alle AOIs annotiert sind, wird mit Protégé die AOI–Ontologie erstellt. In Abbildung 7.3 ist die zu Stimulus (1) in Abbildung 7.1 gehörende AOI–Ontologie abgebildet. Die gestrichelten Pfeile zeigen die Schema–Klasse von der die jeweilige AOI Klasse ableitet. Zur besseren Veranschaulichung haben die AOIs in der Abbildung bereits die Farbe, die ihnen später zugewiesen wird. Die AOI–Ontologie wird nach ihrer Fertigstellung in cEdit importiert und den AOIs werden die zugehörigen Ontologie Klassen zugewiesen. Schließlich wird die Ausgabe von cEdit, die Motif Datei, gespeichert.

7.1.3 Durchführung

Nun wird der Prototyp gestartet und über die Funktion zum Sofortimport werden die Motif Dateien und die Ontologien der drei Stimuli importiert. Es folgt die Verarbeitung auf AOI–Ebene für alle drei Stimuli. Sie wird im Folgenden exemplarisch an Stimulus (1) aus Abbildung 7.1 gezeigt. Über die Stimulus–Visualisierung werden den AOIs verschiedene Farben zugewiesen. Nach der Zuweisung der Farben ergibt sich für die Pattern Search Lanes die in Bereich (1) in Abbildung 7.4 dargestellte Ansicht. Zur komfortableren Analyse sind die Haken in den Checkboxen *Gleiche Größe* und *Gleicher Abstand* gesetzt. Dadurch haben alle AOI–Besuche denselben Durchmesser und einen einheitlichen Abstand voneinander. Zum Vergleich mit den Pattern Search Lanes ist die Stimulus–Visualisierung in der Scanpfad Ansicht in Bereich (2) von Abbildung 7.4 abgebildet. Die Berechnung des Repräsentanten ergibt als Ergebnis den Probanden mit dem Namen „Part04“. In Abbildung 7.4 enthält die Bahn in Bereich (3) den Probanden mit dem Namen „Part04“. Betrachtet man für alle AOI–Besuche von Proband „Part04“ die in Abbildung 7.5 abgebildeten Mouseover Informationen so ergibt sich die folgende durch AOI–Besuche beschriebene Lösungsstrategie von Proband „Part04“:

1. XWertBalken3 – XWertBalken4 – XWertBalken5: Dies entspricht der visuellen Suche nach dem Wert 5 auf der X–Achse.
2. Balken5Oberkante: Der Proband fokussiert die Oberkante des Balkens mit der Nummer 5.

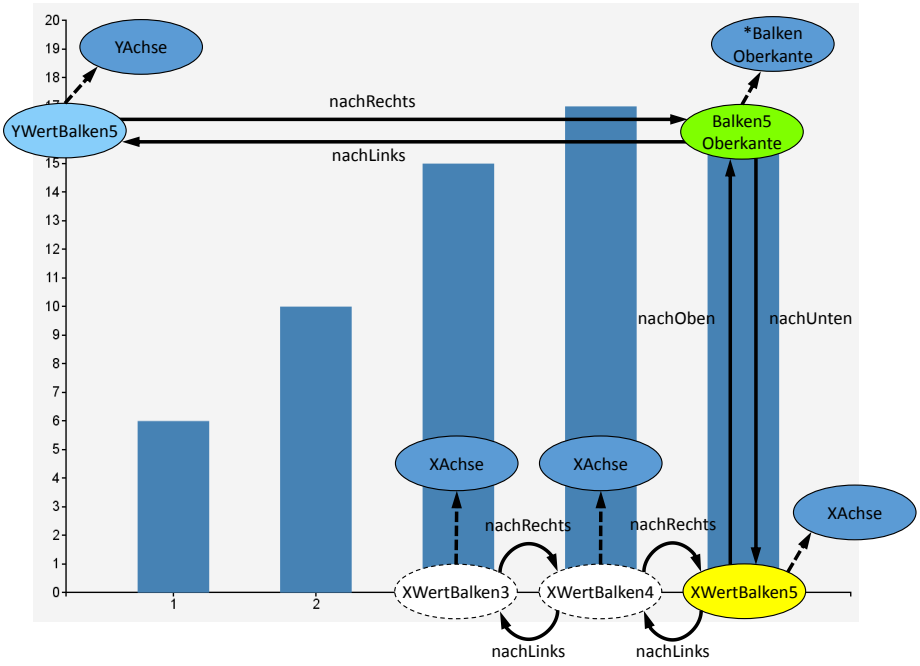

Abbildung 7.3: AOI–Ontologie für Stimulus (1) aus Abbildung 7.1. Zur besseren Veranschaulichung entsprechen die Farben der AOIs den Farben die ihnen später zugewiesenen werden. Es sind die Orientierungsrelationen und die Schema–Klassen dargestellt, von denen die AOIs abgeleitet sind. AOIs mit gestrichelter Randdarstellung haben ein negatives Gewicht.

3. YWertBalken5: Der Wert auf der Y–Achse, der der Höhe von Balken 5 entspricht, wird vom Probanden fokussiert.
4. Balken5Oberkante – YWertBalken5: Der Proband führt ein Cross–Checking durch und vergleicht nochmals die Oberkante von Balken 5 mit dem zugehörigen Wert auf der Y–Achse.

Zur Erzeugung einer repräsentativen Abfolge von AOIs sieht das Konzept die Entfernung von AOIs vor, die während der visuellen Suche oder während des Cross–Checkings besucht werden. Die repräsentative Abfolge, die in den Modus zur Verarbeitung auf Schema–Klassen–Ebene übernommen werden soll, wird per Drag & Drop in der Ergebniszusammenstellung zusammengestellt. Dabei stehen negativ gewichtete AOIs nicht zur Zusammenstellung zur Verfügung. Dadurch wird verhindert, dass AOIs, die während der visuellen Suche besucht werden, in die repräsentative Abfolge gelangen. AOIs, die während des Cross–Checkings besucht werden, sind vom Benutzer von Hand herauszufiltern. Lässt der Benutzer die unerlaubten AOIs bei der Zusammenstellung weg, so reduziert sich die Abfolge von Proband „Part04“ auf:

1. XWertBalken5
2. Balken5Oberkante

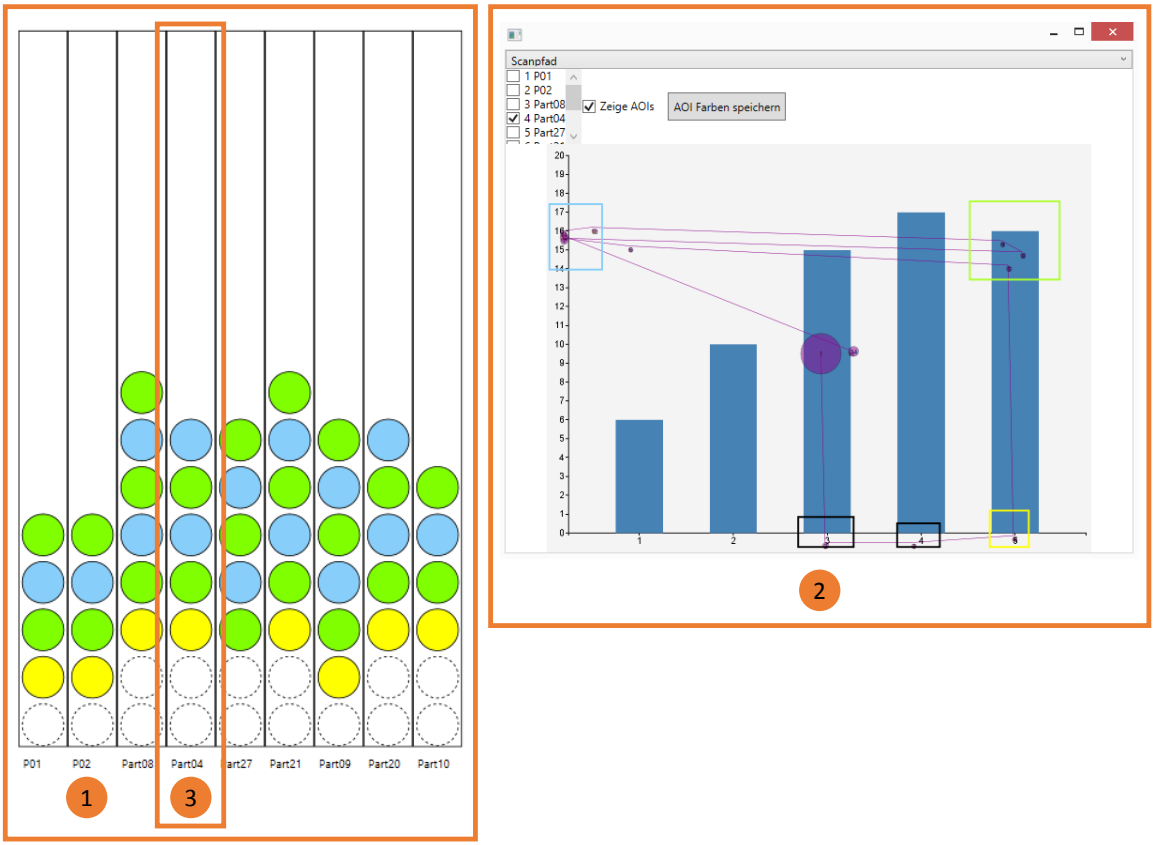

Abbildung 7.4: Pattern Search Lanes und Scanpfad Ansicht für Stimulus (1) aus Abbildung 7.1. Bereich (1) zeigt die Pattern Search Lanes mit den AOI Abfolgen der verschiedenen Probanden. Bereich (2) zeigt den Scanpfad des Repräsentanten. In den Pattern Search Lanes enthält die Bahn von Proband „Part04“ (3) die AOI Abfolge des Repräsentanten.

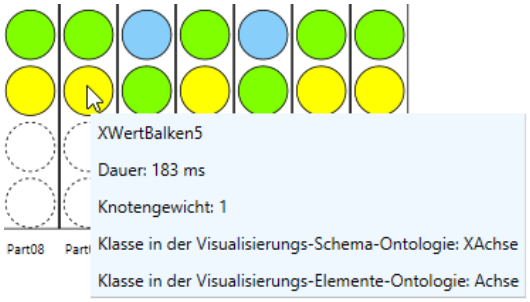

Abbildung 7.5: Mouseover Informationen. Durch den Aufenthalt des Mauszeigers innerhalb eines AOI–Besuchs werden die Daten des AOI–Besuchs angezeigt. Dazu gehört der Name der AOI, die Dauer des AOI–Besuchs, das Gewicht der AOI und die Klassen der Vererbungs–Hierarchie in der Ontologie–Hierarchie.

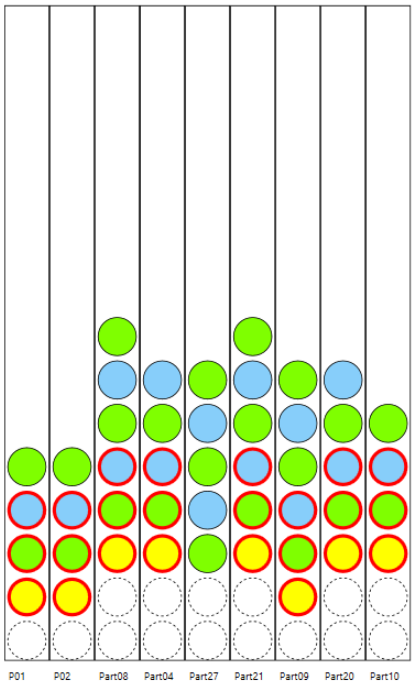

Abbildung 7.6: Exakte Suche in den Pattern Search Lanes. Die Suche ergibt, dass neben dem Probanden mit dem Namen „Part04“ noch sieben weitere Probanden grundsätzlich dieselbe Lösungsstrategie verwenden.

3. YWertBalken5

Dabei handelt es sich nun um die repräsentative Abfolge von AOIs für diesen Stimulus. Diese Abfolge wird vom Benutzer in der Ergebniszusammenstellung zusammengestellt und in den Modus zur Verarbeitung auf Schema–Klassen–Ebene übernommen.

Möchte der Benutzer untersuchen, ob und wenn ja, wie häufig diese Abfolge von den anderen Probanden benutzt wurde, so kann er eine exakte Suche nach dieser Abfolge ausführen. Dazu werden die entsprechenden AOI–Besuche in der Bahn von Proband „Part04“ markiert und die exakte Suche ausgeführt. Das Ergebnis der Suche ist in Abbildung 7.6 dargestellt. Man sieht, dass von den acht weiteren Probanden neben Proband „Part04“ genau sieben Probanden grundsätzlich dieselbe Lösungsstrategie wie Proband „Part04“ verwenden. Sie unterscheiden sich lediglich in der Anzahl der während der visuellen Suche auf der X–Achse fokussierten Bereiche und der Anzahl Wiederholungen beim cross-checking.

Dieses Vorgehen, welches bisher exemplarisch an Stimulus (1) aus Abbildung 7.1 gezeigt wurde, wiederholt man nun für jeden der beiden anderen Stimuli. So erhält man für jeden der drei Stimuli eine repräsentative Abfolge von AOIs. Nachdem alle drei repräsentativen Abfolgen feststehen, wird in den Modus zur Verarbeitung auf Schema–Klassen–Ebene gewechselt. Über die Ontologie Visualisierung der Visualisierungs–Schema–Ontologie werden den einzelnen Schema–Klassen Farben zugewiesen.

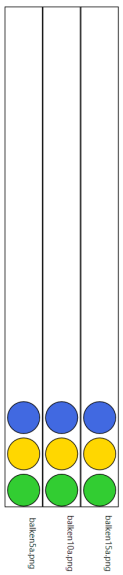

Abbildung 7.7: Pattern Search Lanes im Modus zur Verarbeitung auf Schema–Klassen–Ebene. Alle drei Bahnen enthalten die gleiche Abfolge von Schema–Klassen. Daher ist die Berechnung eines Repräsentanten nicht notwendig.

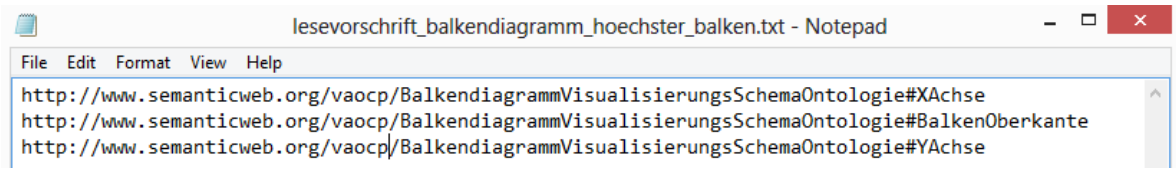

Abbildung 7.8: Die Lesevorschrift als Textdatei exportiert. Die Textdatei enthält die lange Variante der URIs der Lesevorschrift.

Damit ergibt sich für die Pattern Search Lanes des Modus zur Verarbeitung auf Schema–Klassen–Ebene das in Abbildung 7.7 dargestellte Aussehen. Man erkennt, dass alle drei Bahnen die gleiche Abfolge von Schema–Klassen enthalten. Daher kann auf die Berechnung des Repräsentanten in diesem Fall verzichtet werden. Durch das Ablesen der Schema–Klassen in den Mouseover Informationen ergibt sich folgende Abfolge:

1. XAchse
2. BalkenOberkante
3. YAchse

Sie wird per Drag & Drop in der Ergebniszusammenstellung zusammengestellt und im Anschluss als Textdatei exportiert. Abbildung 7.8 zeigt die Textdatei. Die Abbildung zeigt den Export der langen Variante der URIs der repräsentativen Schema–Klassen.

7.2 Erzeugung einer Lesevorschrift für Flächendiagramme

Der folgende Abschnitt beschreibt die Gewinnung einer Lesevorschrift für Flächendiagramme mit Hilfe des Prototyps. Die Aufgabe der Lesevorschrift ist die Ermittlung des höheren Y–Werts zweier vorgegebenen Werte auf der X–Achse und die Angabe des X–Werts. Für die Erzeugung der Lesevorschrift werden die Daten der Studie „Visual Elements VI“ verwendet, die am Institut für Visualisierung und Interaktive Systeme durchgeführt wurde. Im folgenden werden zunächst die Daten der Studie beschrieben, anschließend die Vorbereitungen erläutert und dann die Erzeugung der Lesevorschrift mit Hilfe des Prototypen erklärt.

7.2.1 Verwendete Datenquelle

Die Studie „Visual Elements VI“ untersucht die kognitiven Prozesse bei Visualisierungen. An der Studie nahmen 21 Probanden mit einem Durchschnittsalter von 25,8 Jahren teil. Vier der Probanden gaben an über Vorkenntnisse im Bereich der Visualisierung zu verfügen. 13 Teilnehmer waren weiblich und acht männlich. Es wurden verschiedene Aufgaben gestellt, diese waren im Einzelnen:

1. Das Ablesen von Werten aus Balkendiagrammen
2. Das Ablesen von Werten aus Flächendiagrammen
3. Das Ablesen von Werten aus gestapelten Balkendiagrammen
4. Das Ablesen von Werten aus Liniendiagrammen

Die Studie wurde mit demselben Eyetracker wie die Studie „Visual Elements III“ durchgeführt.

Um die Lesevorschrift zu erzeugen wurden drei Stimuli ausgewählt. Jeder zeigt jeweils ein Flächendiagramm. Der erste Stimulus beinhaltet fünf Werte auf der X–Achse, der zweite zehn und der dritte 15. Die Aufgabe lautet jeweils den höheren Y–Wert zweier auf der X–Achse vorgegebenen Werte zu ermitteln und den Y–Wert sowie den X–Wert anzugeben. Die Diagramme sind mit Heatmap und annotierten AOIs in Abbildung 7.9 dargestellt. In der Abbildung sind die X–Werte, deren Y–Werte verglichen werden sollen, durch einen Pfeil markiert. In der eigentlichen Studie ist dieser Pfeil nicht vorhanden. Die Aufgaben lauten im Einzelnen:

1. Für Stimulus (1): Vergleichen Sie die Produktion 1991 und 1992. Wann ist die Produktion höher? Geben Sie den dazugehörigen Wert an.
2. Für Stimulus (2): Vergleichen Sie die Anzahl der Autounfällen im Februar und September. In welchem der beiden Monate ist die Anzahl höher? Geben Sie den dazugehörigen Wert an.
3. Für Stimulus (3): Vergleichen Sie den Marktanteil 2008 und 2012. Wann ist der Marktanteil höher? Geben Sie den dazugehörigen Wert an.

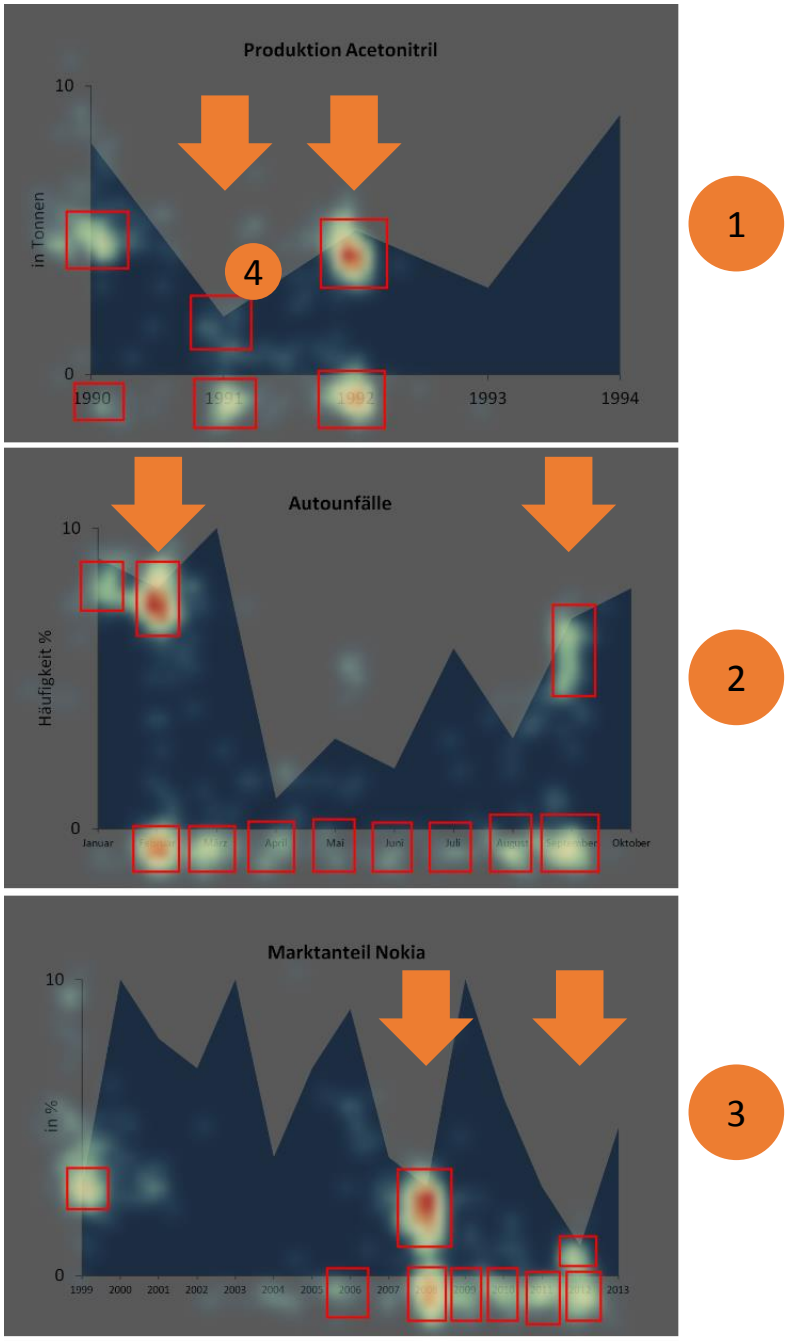

Abbildung 7.9: Die drei Flächendiagramm Stimuli. Auf den Stimuli soll jeweils der höhere Y–Wert zweier auf der X–Achse vorgegebenen Werte ermittelt werden und sowohl der Y–Wert als auch der X–Wert angegeben werden. Die auf der X–Achse vorgegebenen Werte sind jeweils durch einen Pfeil gekennzeichnet. Auf den Stimuli werden mittels einer Heatmap mit cEdit die AOIs identifiziert und annotiert.

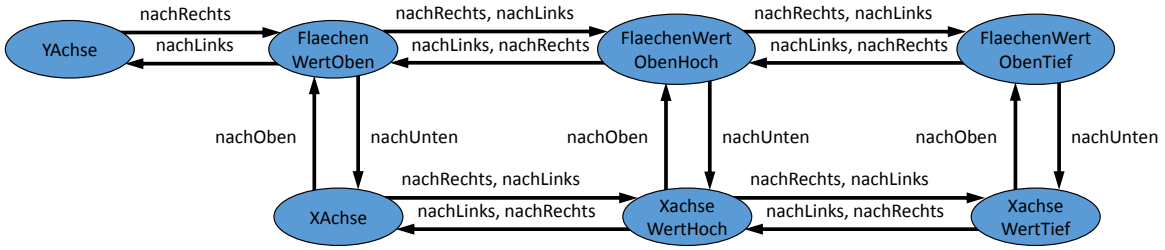

Abbildung 7.10: Die Visualisierungs–Schema–Ontologie für Flächendiagramme.

7.2.2 Vorbereitung

In der Vorbereitung wird die Visualisierungs–Elemente–Ontologie um die Klasse *Flaeche* erweitert. Da bisher nur Balkendiagramme untersucht wurden, wird eine neue Visualisierungs–Schema–Ontologie für Flächendiagramme erstellt. Sie ist in Abbildung 7.10 enthalten. Als nächstes werden wie in der Vorbereitung zur Analyse der Balkendiagramme die Daten für die Anzeige der Heatmap erzeugt und in cEdit importiert. Es folgt die Identifikation und Annotation der AOIs anhand der Heatmap. Abbildung 7.9 zeigt die drei Flächendiagramme mit ihren AOIs und eingeblendeter Heatmap. Bei Stimulus (1) ist die AOI (4) nicht aufgrund der Heatmap, sondern durch die Kenntnis der zu lösenden Aufgabe annotiert worden. Nachdem alle AOIs eingetragen sind folgt die Erstellung der AOI–Ontologie. Für Stimulus (1) aus Abbildung 7.9 ist die AOI–Ontologie in Abbildung 7.11 dargestellt. Die gestrichelten Pfeile zeigen die Schema–Klasse von der die jeweilige AOI Klasse ableitet. Zur besseren Veranschaulichung haben die AOIs in der Abbildung bereits die Farbe, die ihnen später zugewiesen wird. Die AOI–Ontologie wird nach ihrer Fertigstellung in cEdit importiert und den AOIs werden ihre entsprechenden Klassen der Ontologie zugewiesen. Abschließend wird die Motif Datei erzeugt.

7.2.3 Durchführung

Zur Erzeugung einer Lesevorschrift für Flächendiagramme wird der Prototyp gestartet und die Motif Dateien und Ontologien der drei Stimuli per Sofortimport importiert. Daran schließt sich die Verarbeitung auf AOI–Ebene für alle drei Stimuli an. Dies wird nun an Stimulus (1) aus Abbildung 7.9 exemplarisch gezeigt. Den AOIs werden mittels der Stimulus–Visualisierung verschiedene Farben zugewiesen. Nach der Farbzuweisung ergibt sich für die Pattern Search Lanes die in Abbildung 7.12 enthaltene Ansicht. Die AOI–Besuche haben in der Abbildung alle denselben Durchmesser und einen einheitlichen Abstand voneinander, da die Haken in den Checkboxen *Gleiche Größe* und *Gleicher Abstand* gesetzt sind. Die Berechnung des Repräsentanten ergibt den Probanden mit dem Namen „ID14“, seine Bahn ist in der Abbildung mit Bereich (1) markiert. Die Aufgabe für den Stimulus lautet „Vergleichen Sie die Produktion 1991 und 1992. Wann ist die Produktion höher? Geben Sie den dazugehörigen Wert an.“ Im Folgenden wird die Analyse anhand einer intuitiven Abfolge von AOIs und anhand der AOI Abfolge des Repräsentanten gezeigt. Schließlich wird eine Abfolge zur Weiterverarbeitung ausgewählt und die Lesevorschrift erzeugt.

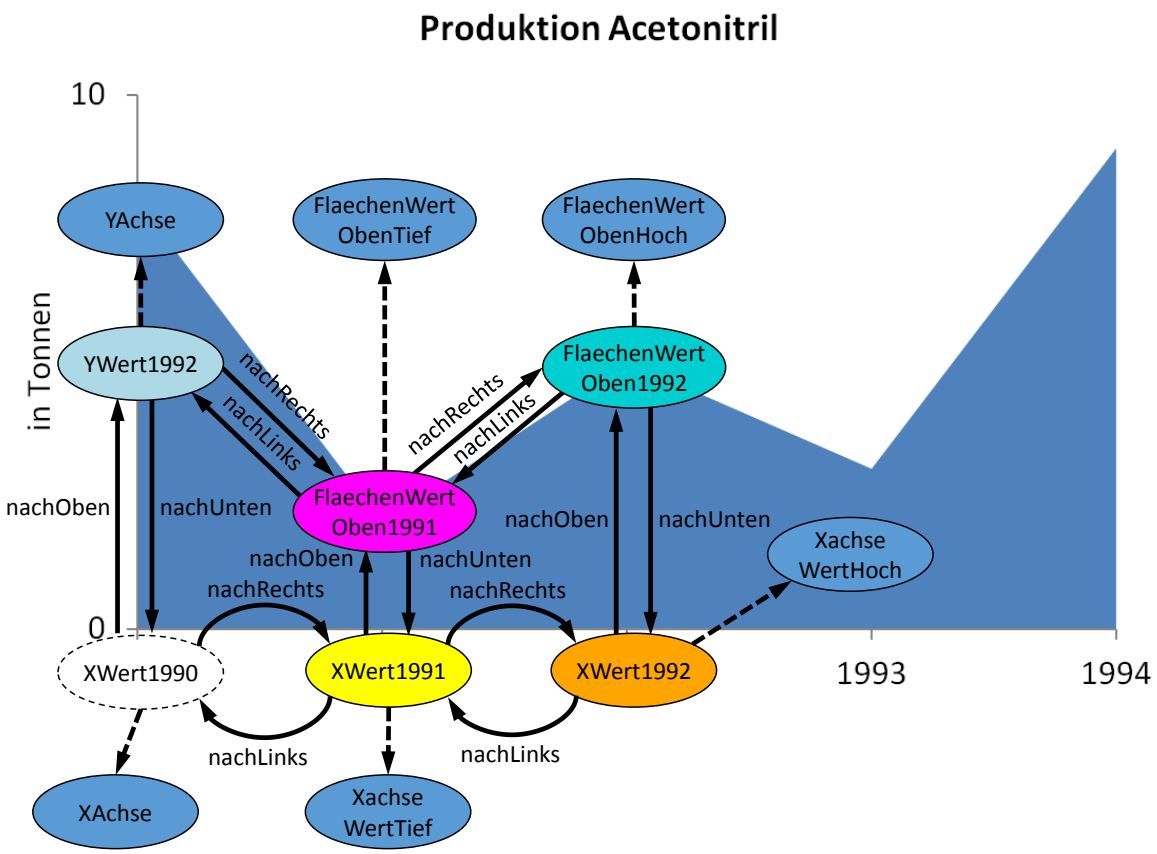

Abbildung 7.11: Die AOI–Ontologie für Stimulus (1) aus Abbildung 7.9. Zur besseren Veranschaulichung entsprechen die Farben der AOIs den Farben die ihnen später zugewiesenen werden. Es sind die Orientierungsrelationen und die Schema–Klassen dargestellt, von denen die AOIs abgeleitet sind. AOIs mit gestrichelter Randdarstellung haben ein negatives Gewicht.

Analyse der intuitiven Abfolge von AOIs

Eine intuitive Abfolge von AOI–Besuchen zur Lösung der Aufgabe wäre zum Beispiel:

1. XWert1991
2. FlaechenWertOben1991
3. XWert1992
4. FlaechenWertOben1992
5. YWert1992

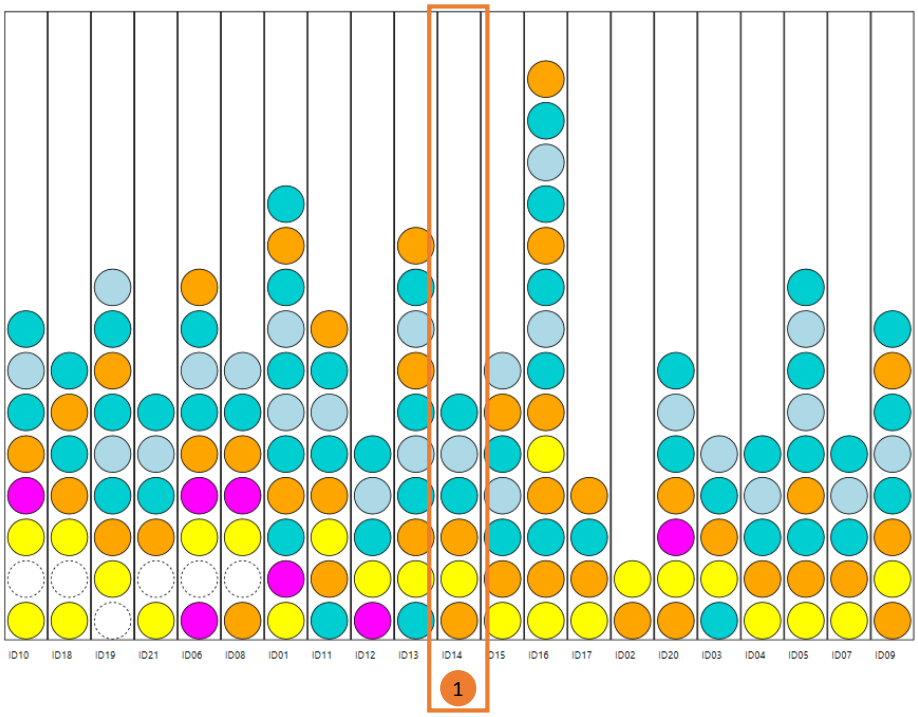

Abbildung 7.12: Ansicht der Pattern Search Lanes für Stimulus (1) aus Abbildung 7.9. Der Bereich (1) enthält den Probanden mit dem Namen „ID14“, der als Repräsentant ermittelt wurde.

Diese Abfolge wird nun unscharf gesucht. Das Suchfenster wird dabei nicht verlängert. Die Einfüge– und Löschkosten betragen jeweils zwei, die Ersetzungskosten eins. Zur Suche werden die entsprechenden AOI–Besuche in Abbildung 7.13 wie in Bereich (1) markiert und die unscharfe Suche gestartet. Pro Bahn werden mehrere Ergebnisse berechnet und können zur Anzeige ausgewählt werden. In der Abbildung werden diejenigen Ergebnisse rot umrandet angezeigt, die die geringste Levenshtein–Distanz zur markierten Abfolge haben. Haben mehrere Abfolgen dieselbe Levenshtein–Distanz, so wird die Abfolge ausgewählt, deren erster AOI–Besuch vor dem ersten AOI–Besuch der Abfolgen mit derselben Levenshtein–Distanz liegt.

Die Kosten der ausgewählten Abfolgen sind in Tabelle Tabelle 7.1 zusammen mit der Häufigkeit ihres Vorkommens enthalten. Anhand der Tabelle lässt sich feststellen, dass 13 der 20 anderen Probanden eine Abfolge von AOI–Besuchen haben, die mit Kosten von zwei oder weniger in die intuitive Abfolge überführt werden können.

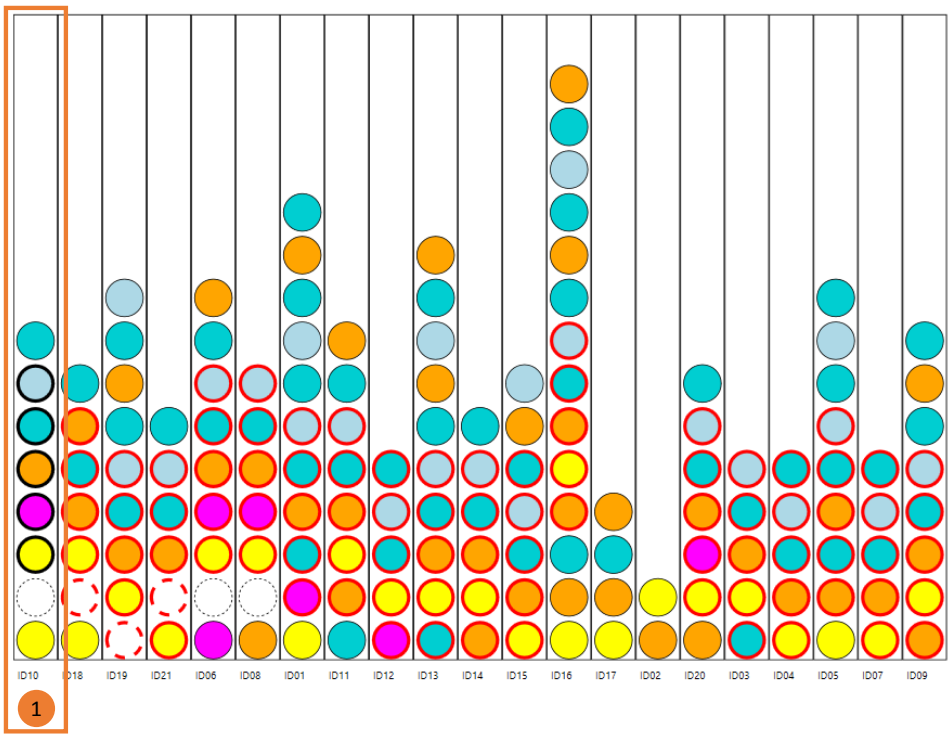

Abbildung 7.13: Unscharfe Suche der intuitiven AOI Abfolge. Die AOI–Besuche der unscharf zu suchenden Abfolge werden in Bereich (1) markiert und anschließend unscharf gesucht. Die Suchergebnisse werden in den anderen Bahnen rot umrandet hervorgehoben.

Kosten	Häufigkeit
0	3
1	1
2	9
3	1
4	3
5	1
kein Vorkommen	2

Tabelle 7.1: Ergebnisse der unscharfen Suche. Die Suchergebnisse werden zusammen mit der Häufigkeit ihres Auftretens aufgelistet. Aus der Auflistung ergibt sich, dass 13 der 20 anderen Probanden eine Abfolge von AOI–Besuchen nutzten, die eine Levenshtein–Distanz von zwei oder weniger zur intuitiven Abfolge haben.

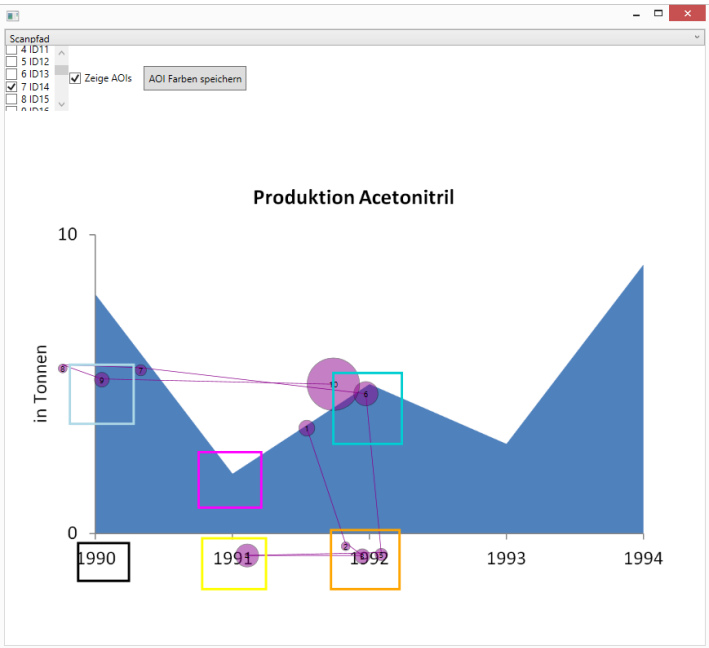

Abbildung 7.14: Scanpfad des Probanden mit dem Namen „ID14“. Der Scanpfad weist keine Fokussierung der AOI „FlächenWertOben1991“ auf. Es wird vermutet, dass der Proband mit Hilfe des peripheren Sehens erkannt hat, dass der Y–Wert von 1992 höher ist als der von 1991.

Analyse der AOI Abfolge des Repräsentanten

Die Berechnung des Repräsentanten hat die AOI Abfolge des Probanden mit dem Namen „ID14“ als Ergebnis. Die Abfolge ist in Abbildung 7.12 in der mit Bereich (1) markierten Bahn enthalten. Die Abfolge lautet:

1. XWert1992
2. XWert1991
3. XWert1992
4. FlächenWertOben1992
5. YWert1992
6. FlächenWertOben1992

Auffällig ist, dass die AOI „FlächenWertOben1991“ nicht fokussiert wurde. Bei Betrachtung des in Abbildung 7.14 abgebildeten Scanpfads von Proband „ID14“ liegt die Vermutung nahe, dass der Proband mit Hilfe des peripheren Sehens erkannt hat, dass der Y–Wert von 1992 höher ist als der von 1991.

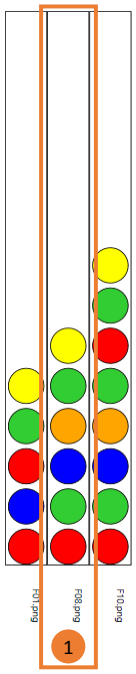

Abbildung 7.15: Pattern Search Lanes im Modus zur Verarbeitung auf Schema–Klassen–Ebene. Als Repräsentanten wird die Abfolge des Stimulus (2) aus Abbildung 7.9 berechnet. Sie ist in Bereich (1) enthalten.

Erzeugung der Lesevorschrift

Es liegt im Ermessen des Benutzers welche Abfolge von AOIs er für die Weiterverarbeitung auswählt. Der Autor entscheidet sich an dieser Stelle für die Aussagekraft des Repräsentanten und wählt ihn zur Weiterverarbeitung aus. Dazu werden die AOIs des Repräsentanten in der Ergebniszusammenstellung zusammengestellt. Der Repräsentant enthält keine AOI–Besuche, die während der visuellen Suche fokussiert wurden, aber ein Cross–Checking. Dieses wird nicht in die Ergebniszusammenstellung übernommen. Somit ergibt sich die repräsentative AOI Abfolge dieses Stimulus zu:

1. XWert1992
2. XWert1991
3. XWert1992
4. FlächenWertOben1992
5. YWert1992

Für die anderen beiden Stimuli wird ebenfalls der Repräsentant zur Weiterverarbeitung ausgewählt. Nach dem Wechsel in den Modus zur Verarbeitung auf Schema–Klassen–Ebene werden den Schema–Klassen anhand der Ontologie Visualisierung Farben zugewiesen. Dadurch ergibt sich die in Abbil-

dung 7.15 enthaltene Ansicht der Pattern Search Lanes. Die Berechnung des Repräsentanten hat die Abfolge von Stimulus (2) in Abbildung 7.9 zum Ergebnis. Sie ist in Abbildung 7.15 in der Bahn im Bereich (1) enthalten und ergibt sich durch das Ablesen der Schema–Klassen aus den Mouseover Informationen zu:

1. XAchseWertHoch
2. FlaechenWertObenHoch
3. XAchseWertTief
4. FlaechenWertObenTief
5. FlaechenWertObenHoch
6. YAchse

Diese Abfolge stellt die repräsentative Abfolge von Schema–Klassen und somit die Lesevorschrift dar. Sie kann in der Ergebniszusammenstellung zusammengestellt und zur weiteren Verwendung in eine Textdatei exportiert werden.

7.3 Offene Fragestellungen bei bestimmten Visualisierungstypen

Eine Herausforderung bei der Erzeugung einer Lesevorschrift ist die Erstellung einer geeigneten Visualisierungs–Schema–Ontologie zu dem zu untersuchenden Visualisierungstyp. Weiterhin muss die Visualisierungs–Schema–Ontologie zur Aufgabe passen, zu der die Lesevorschrift erzeugt werden soll. Dies hat sich insbesondere in Abschnitt 7.2 bei der Erzeugung der Lesevorschrift für Flächendiagramme gezeigt. Dabei musste anhand der Schema–Klassen erkennbar sein, ob es sich um den größeren oder den kleineren Y–Wert handelt. Im Folgenden werden zwei offene Fragestellungen hinsichtlich zweier bestimmter Visualisierungstypen beschrieben.

7.3.1 Sudokus

Im Rahmen der Demonstration des Prototyps wurde versucht eine Lesevorschrift zu erzeugen, um den Werts einer Zelle eines Sudokus zu bestimmen. Eine offene Frage dabei blieb, wie eine Visualisierungs–Schema–Ontologie für Sudokus aussehen könnte. Jede Zelle des Sudokus ist für die Gesamtlösung eines Sudokus von Bedeutung. Annotiert man die Zellen eines Sudokus mit AOIs so ist es, bedingt durch den Aufbau eines Sudokus, eine offene Frage, wie die Klassen der Visualisierungs–Schema–Ontologie gestaltet werden sollen, von denen die AOIs abgeleitet werden sollen. Das Problem liegt hierbei in der Überlappung der Bereiche auf den Sudokus, die den Klassen der Visualisierungs–Schema–Ontologie zugeordnet sind. Dies soll am nachfolgenden Beispiel verdeutlicht werden.

Um den Wert der Zelle des Sudokus im Beispiel zu bestimmen, wird die Methode des Ausschließens von Ziffern verwendet. Dabei werden direkt die bereits vorhandenen Ziffern der Gruppe ausgeschlossen, zu denen die gesuchte Zelle gehört. Eine Gruppe umfasst eine Zeile, eine Spalte und einen Block. Unter Anwendung des beschriebenen Lösungsverfahrens soll herausgefunden werden, welche Ziffer in Bereich (1) in Abbildung 7.16 einzutragen ist. Dazu muss laut Lösungsverfahren die Ziffer im

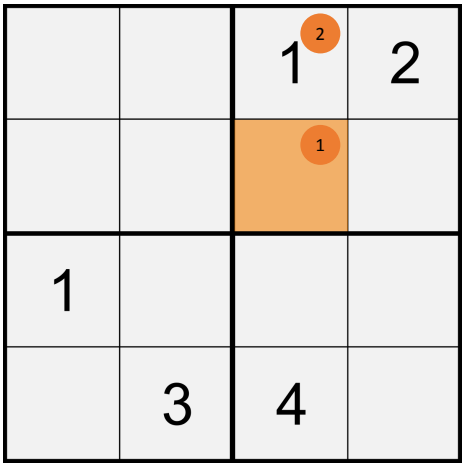

Abbildung 7.16: Ein 4x4 Sudoku. Für das Sudoku wurde versucht eine Lesevorschrift zu ermitteln mit der es möglich ist, den Wert von Zelle (1) zu bestimmen. Eine offene Frage ist hierbei wie eine Visualisierungs–Schema–Ontologie aussehen könnte, die die Zelle (2) so repräsentiert, dass eine Lesevorschrift abgeleitet werden kann.

Bereich (2) aus zwei Gründen fokussiert werden. Zum einen liegt die Ziffer im Bereich (2) in derselben Spalte wie die gesuchte Ziffer. Und zum anderen liegt die Ziffer im Bereich (2) im selben Block wie die gesuchte Ziffer. Die Frage ist nun, wie soll eine Klasse der Visualisierungs–Schema–Ontologie aussehen, von der die AOI für Bereich (2) ableitet. An der Schema–Klasse muss erkennbar sein, ob die zur Schema–Klasse gehörende AOI aus dem Grund fokussiert wurde, weil die Spalte oder weil der Block betrachtet wurde.

Dazu kommt die Frage wie in Anbetracht der Menge von verschiedenen Lösungsstrategien für Sudokus eine allgemeingültige Visualisierungs–Schema–Ontologie erstellt werden kann.

7.3.2 Kuchendiagramme in Verbindung mit einer Anordnungsaufgabe

Bei der Demonstration des Prototyps wurde auch versucht eine Lesevorschrift für den Visualisierungstyp Kuchendiagramm zu erzeugen. Die Aufgabe dabei ist es, die Ausschnitte eines Kuchendiagramms der Größe nach absteigend sortiert zu nennen. Ein Beispiel dafür ist in Abbildung 7.17 abgebildet. Dabei sollen die Ausschnitte A bis D des Kuchendiagramms aus der Abbildung ihrer Größe nach sortiert werden.

Damit die Simulation die Diagramm Ausschnitte eines Kuchendiagramms mit beispielsweise vier Ausschnitten der Reihe nach per visueller Suche suchen kann, müssten die Schema–Klassen folgendermaßen aussehen: „Größter Ausschnitt“, „Zweitgrößter Ausschnitt“, „Drittgrößter Ausschnitt“, „Viertgrößter Ausschnitt“. Dies würde eine Visualisierungs–Schema–Ontologie ergeben, die für den Visualisierungstyp „Kuchendiagramm mit vier Ausschnitten“ gültig ist. Dadurch wird jedoch eine Abhängigkeit der Visualisierungs–Schema–Ontologie vom jeweiligen Stimulus erzeugt. Dies ist zur

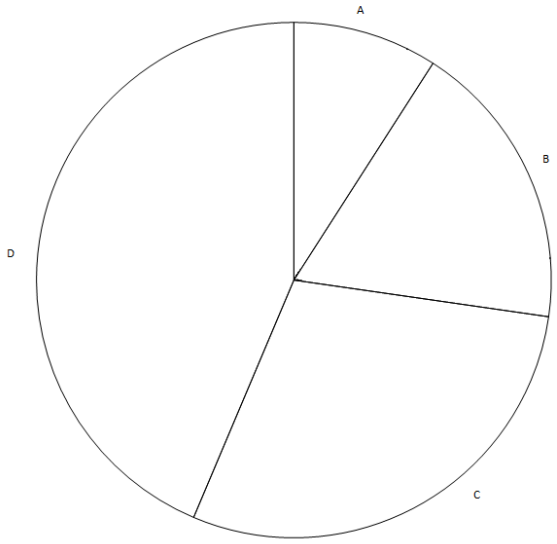

Abbildung 7.17: Ein Kuchendiagramm. Für das Kuchendiagramm wurde versucht eine Lesevorschrift zu gewinnen. Die Aufgabe lautet dabei, die Ausschnitte des Kuchendiagramms der Größe nach absteigend zu sortieren. Eine offene Frage hierbei ist, wie eine Visualisierungs–Schema–Ontologie aussehen könnte, um die Ableitung einer Lesevorschrift zu ermöglichen.

allgemeingültigen Beschreibung eines Visualisierungstyps aber ungeeignet. Somit ergibt sich eine weitere offen Frage und zwar, wie eine Visualisierungs–Schema–Ontologie für Kuchendiagramme in Verbindung mit einer Anordnungsaufgabe aussehen könnte.

8 Zusammenfassung und Ausblick

Ziel dieser Arbeit war es ein Konzept zu entwickeln, mit dem durch die visuelle Analyse von AOI Abfolgen, unter Ausnützung der Eye–Mind–Hypothese, kognitive Prozesse visuell analysierbar gemacht werden. Basierend auf dieser visuellen Analyse soll eine Lesevorschrift für einen Visualisierungstyp erzeugt werden können. Eine Lesevorschrift ist die Vorstufe zu einem Satz von Produktionsregeln, der für eine Simulation von Augenbewegungen benötigt wird.

Ausgehend von der Recherche zur Visualisierung dynamischer Prozesse und zur Darstellung von Wissen wurde ein Konzept entwickelt, das die Gewinnung von Lesevorschriften ermöglicht. Das Konzept sieht vor, eine für eine bestimmte Aufgabe gültige Lesevorschrift für einen Visualisierungstyp aus Eye–Tracking–Daten mehrerer Probanden abzuleiten. Die Eye–Tracking–Daten wurden auf verschiedenen Stimuli des Visualisierungstyps während der Lösung der Aufgabe erfasst. Das Konzept sieht ein mehrstufiges Vorgehen zur Ableitung der Lesevorschrift vor. In einem ersten Schritt werden die Stimuli mit AOIs annotiert. Im nächsten Schritt werden für jeden der Stimuli die Abfolgen der AOIs der Probanden analysiert. Daraufhin wird für jeden der Stimuli eine Abfolge von AOIs bestimmt, die die Lösungsstrategie repräsentiert, die die meisten Probanden zur Lösung der Aufgabe auf dem Stimulus wählten. Aus diesen Abfolgen wird im Anschluss diejenige ausgewählt, die die Lösungsstrategie repräsentiert, die trotz der Verschiedenheit der Stimuli eine Lösungsstrategie für beliebige Stimuli des Visualisierungstyps darstellt. Zur Bestimmung der repräsentativen Lösungsstrategien wurde ein Visualisierungs– und Analysekonzept entwickelt, das mit einem automatischen, einem halbautomatischen und einem manuellen Verfahren arbeitet. Die Verfahren werden durch Algorithmen zur Ähnlichkeitsbestimmung von Lösungsstrategien und zur Mustersuche in Lösungsstrategien unterstützt.

Im Detail bedeutet dies, dass die bereits in der Arbeit über die Augenbewegungs–Simulation verwendete Ontologie–Hierarchie für das Lösungskonzept angepasst und weiterverwendet wurde. Die Ontologie–Hierarchie besteht aus der Visualisierungs–Elemente–Ontologie, der Visualisierungs–Schema–Ontologie und der AOI–Ontologie. Das Konzept sieht zur Ableitung einer Lesevorschrift einen Ablauf in mehreren Schritten vor, die sich auf die Schichten der Ontologie–Hierarchie beziehen. Zunächst werden die benötigten Eingabedaten importiert. Dazu zählen die Eye–Tracking–Daten und die Stimuli. Im weiteren Verlauf werden die beiden obersten Ontologien der Ontologie–Hierarchie neu erstellt beziehungsweise erweitert. Daraufhin folgt die Verarbeitung der AOI Abfolgen der M Probanden der N Stimuli der Eingabe bei der für jeden der Stimuli eine repräsentative Abfolge von AOIs erzeugt wird. Nach dem Wechsel auf die Ebene zur Verarbeitung auf Schema–Klassen Ebene wird aus den N Abfolgen der Schema–Klassen, von denen die AOIs der N repräsentativen AOI Abfolgen abgeleitet sind, eine ausgewählt, die als repräsentative Schema–Klassen Abfolge für den untersuchten Visualisierungstyp gilt. Zur Bestimmung repräsentativer Abfolgen wurde ein Visualisierungs– und Analysekonzept entwickelt, das die automatische, halbautomatische oder manuelle Bestimmung der repräsentativen Abfolgen ermöglicht. Für die automatische Bestimmung kommt die Berechnung

eines Repräsentanten zum Einsatz. Diese Berechnung basiert auf Ähnlichkeitsvergleichen mittels der Levenshtein–Distanz. Für dein manuelle Bestimmung der repräsentativen Abfolgen wurden mehrere Visualisierungen entwickelt. Dazu gehören die Ontologie–Visualisierung, die Visualisierungstechnik des Space–Time–Cubes, die Stimulus–Visualisierung, die Pattern Search Lanes und der Graph. Zur Unterstützung der manuellen Analyse mittels der Pattern Search Lanes wurden die Verfahren der Elementsuche, der exakten Suche, der unscharfen Suche und des lokalen Scanpfads entwickelt.

Die wichtigsten Teile des Konzepts wurden anhand eines Prototypen implementiert. Der Prototyp beherrscht die automatische, die halbautomatische und die manuelle Bestimmung der repräsentativen Abfolgen. Zur automatischen beziehungsweise zur Umsetzung des automatisierten Teils der halbautomatischen Erzeugung der repräsentativen Abfolgen wurde die Berechnung eines Repräsentanten mittels der Levenshtein–Distanz implemeniert. Für den manuellen Teil der Umsetzung der halbautomatischen beziehungsweise der manuellen Erzeugung der repräsentativen Abfolgen wurden mehrere Visualisierungen implementiert. Dazu zählen die Ontologie–Visualisierung, die Stimulus–Visualisierung, die Pattern Search Lanes und der Graph. Für die manuelle Analyse anhand der Pattern Search Lanes wurden die Verfahren der exakten Suche, der unscharfen Suche und des lokalen Scanpfads implementiert. Die Visualisierungen wurden auf Basis eines Plugin–Systems entwickelt so dass der Prototyp einfach erweitert kann und die Visualisierungen einfach ausgetauscht werden können.

Die Funktionsweise des Prototyps wurde anhand zweier Szenarien demonstriert. Im ersten Szenario wurde eine Lesevorschrift für den Visualisierungstyp Balkendiagramm für das Ablesen der Höhe eines bestimmten Balkens erzeugt. Im zweiten Szenario entstand eine Lesevorschrift für den Visualisierungstyp Flächendiagramm für das Angeben des höheren Y–Werts zweier X–Werte sowie die Nennung des zum höheren Y–Werts gehörenden X–Werts.

8.1 Diskussion

Wie in der Demonstration des Prototypen gezeigt wurde, konnte für Balken– und Flächendiagramme eine Lesevorschrift erzeugt werden. Die Demonstration hat aber auch aufgezeigt, dass noch offene Fragen hinsichtlich der Erstellung passender Visualisierungs–Schema–Ontologien für kompliziertere Visualisierungstypen, wie etwa Sudokus, bestehen. Eine offene Frage besteht insbesondere darin, wie Klassen der Visualisierungs–Schema–Ontologie aussehen können, deren abgeleitete AOIs sich auf den Stimuli überlappen.

Weiterhin wurde bei der Demonstration des Prototypen bei der Erzeugung einer Lesevorschrift für Flächendiagramme festgestellt, dass die dafür erstellte Visualisierungs–Schema–Ontologie von der gestellten Aufgabe abhängig ist. Dies liegt daran, dass die zur Lösung der Aufgabe notwendigen Klassen der Visualisierungs–Schema–Ontologie unterscheidbar sein sollen, da zu zwei X–Werten die zugehörigen Y–Werte verglichen werden sollen.

Durch die Nutzung von Farben zur Darstellung von AOIs und Schema–Klassen in den Pattern Search Lanes als zentralem Instrument der manuellen Analyse besteht aufgrund der begrenzten Unterscheidbarkeit von Farben eine Limitierung hinsichtlich der verarbeitbaren Menge von AOIs pro Stimulus beziehungsweise Schema–Klassen in der Visualisierung–Schema–Ontologie. Dies kann

durch die Nutzung von Repräsentanten zur Erzeugung von repräsentativen Abfolgen umgangen werden. Jedoch ist gerade im Falle einer großen Anzahl von AOIs zu vermuten, dass die zur Berechnung von Repräsentanten vorausgesetzte grundsätzliche Ähnlichkeit der AOI Abfolgen nicht gegeben ist.

Für die Filterung der visuellen Suche (siehe Abschnitt 4.3) müssten umfangreiche Filterverfahren entwickelt werden. Allein durch die Gewichtung von AOIs kann insbesondere bei komplizierteren Visualisierungstypen und Aufgaben nicht garantiert werden, dass alle Schema–Klassen aus der Lesevorschrift herausgefiltert werden, deren abgeleitete AOIs während der visuellen Suche fokussiert werden. Beispielsweise existiert der Fall, dass eine Abfolge von AOIs einen positiv gewichteten AOI–Besuch enthält, dessen AOI während einer visuellen Suche fokussiert wurde. Derzeit muss dies vom Benutzer erkannt werden und falls die betreffende Abfolge zu einer repräsentativen Abfolge weiterverarbeitet werden soll, muss der Benutzer die AOI an dieser Position aus der Abfolge entfernen.

Deaktiviert man in der Kognitions–Simulation die Simulation des Cross–Checkings durch künstliches Rauschen, so kann Cross–Checking durch Produktionsregelsätze simuliert werden, die Cross–Checking beinhalten. Somit kann Cross–Checking auch in Lesevorschriften enthalten sein und muss nicht aufwendig gefiltert werden. Kommt man dann für eine Lesevorschrift zu dem Ergebnis, dass sie Cross–Checking enthalten soll, da die überwiegende Anzahl von Probanden Cross–Checking durchführt, so basiert dies auf der Analyse der Eye–Tracking–Daten und kann auch so simuliert werden. Überlässt man hingegen weiterhin der Simulation das Cross–Checking, so müssten auch für das Cross–Checking umfangreiche Filterverfahren entwickelt werden.

Der Einfluss der peripheren Sehens ist bei der Analyse von mit AOIs annotierten Stimuli ein grundsätzliches Problem, wie auch während der Demonstration zur Erzeugung einer Lesevorschrift für Flächendiagramme festgestellt wurde. Dem kann nur durch eine gleichzeitige Analyse des zur AOI Abfolge zugehörigen Scanpfads entgegengewirkt werden.

8.2 Ausblick

Es besteht Bedarf in der Klärung der Frage, ob eine für einen Visualisierungstyp allgemeingültige Visualisierungs–Schema–Ontologie entwickelt werden kann und falls ja, wie diese aufgebaut ist. Die Entwicklung und Implementierung der in Abschnitt 8.1 erwähnten Filterverfahren zur Filterung der visuellen Suche bietet ebenfalls Anknüpfungspotenzial.

Eine Erweiterungsmöglichkeit ist die Umsetzung der in Abschnitt 5.7.2 beschriebenen Nutzung einzelner Bahnen der Pattern Search Lanes als Repräsentant für mehrere Probanden. Durch die Kombination mehrerer auf diese Art genutzter Bahnen wird die Analyse von Eye–Tracking–Daten vieler Probanden ermöglicht.

Eine große Erweiterungsmöglichkeit besteht in der Ausweitung der Vorverarbeitung der Eye–Tracking–Daten. Da es letztendlich darum geht wiederkehrende Muster in den Daten zu erkennen, wäre hier eine automatisierte Vorverarbeitung mit Verfahren zur Mustererkennung unterstützt durch Machine Learning und Computer Vision denkbar.

Die Vision besteht darin, in Zukunft einen umfangreichen Katalog von Lesevorschriften für die verschiedensten Visualisierungstypen und Aufgaben aufzubauen, um dadurch die Simulation von Augenbewegungen auf einem breit gefächerten Spektrum von Stimuli für die unterschiedlichsten Aufgaben zu ermöglichen.

Literaturverzeichnis

[AB73] J. R. Anderson, G. H. Bower. *Human Associative Memory.* Winston and Sons, Washington, D.C., 1973. (Zitiert auf Seite 22)

[ABB+04] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, Y. Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004. (Zitiert auf Seite 22)

[ABD04] J. R. Anderson, D. Bothell, S. Douglass. Eye Movements Do Not Reflect Retrieval Processes Limits of the Eye-Mind Hypothesis. *Psychological Science*, 15(4):225–231, 2004. (Zitiert auf Seite 18)

[AL98] J. R. Anderson, C. J. Lebiere. *The atomic components of thought.* Psychology Press, 1998. (Zitiert auf Seite 22)

[AMST11] W. Aigner, S. Miksch, H. Schumann, C. Tominski. *Visualization of time-oriented data.* Springer, 2011. (Zitiert auf Seite 56)

[And76] J. R. Anderson. *Language, Memory and Thought.* Mahwah, NJ: Erlbaum, 1976. (Zitiert auf Seite 22)

[And83] J. R. Anderson. *The Architecture of Cognition.* Harvard University Press, Cambridge, MA, 1983. (Zitiert auf Seite 22)

[And93] J. R. Anderson. *Rules of the Mind.* Lawrence Erlbaum Associates Inc, New Jersey, 1993. (Zitiert auf Seite 22)

[And01] J. R. Anderson. *Kognitive Psychologie.* Spektrum, Akad. Verl., Heidelberg, 3. aufl. Auflage, 2001. (Zitiert auf den Seiten 18, 21 und 22)

[Bir04] F. Maine de Biran. The influence of habit on the faculty of thinking. 1804. (Zitiert auf Seite 20)

[BKR+] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, T. Ertl. State-of-the-Art of Visualization for Eye Tracking Data. (Zitiert auf Seite 14)

[Bor97] W. N. Borst. *Construction of engineering ontologies for knowledge sharing and reuse.* Universiteit Twente, 1997. (Zitiert auf Seite 24)

[CS80] N. J. Cohen, L. R. Squire. Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210(4466):207–210, 1980. (Zitiert auf Seite 21)

[Duc07] A. T. Duchowski. *Eye tracking methodology: Theory and practice*, Band 373. Springer, 2007. (Zitiert auf den Seiten 13 und 14)

[Eng13] S. Engelhardt. *Modellierung kognitiver Prozesse in der Visualisierung.* Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2013. (Zitiert auf den Seiten 27, 28, 29, 33, 37, 38, 39 und 40)

[FF06] J. Funke, P. A. Frensch. *Handbuch der Allgemeinen Psychologie-Kognition.* Hogrefe Verlag, 2006. (Zitiert auf Seite 18)

[GK99] J. H. Goldberg, X. P. Kotval. Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics*, 24(6):631–645, 1999. (Zitiert auf Seite 15)

[GOS09] N. Guarino, D. Oberle, S. Staab. What is an Ontology? In *Handbook on ontologies*, S. 1–17. Springer, 2009. (Zitiert auf den Seiten 23, 24 und 41)

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993. (Zitiert auf den Seiten 23 und 24)

[Her13] D. Herr. *Neue visualisierungsbasierte Analysetechniken für Eye-Tracking-Daten.* Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2013. (Zitiert auf Seite 72)

[HNA+11] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, J. Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures.* Oxford University Press, 2011. (Zitiert auf den Seiten 13, 14 und 18)

[Jam90] W. James. The principles of psychology. 1890. (Zitiert auf Seite 20)

[JC80] M. A. Just, P. A. Carpenter. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87:329–354, 1980. (Zitiert auf Seite 17)

[Kao07] M. Y. Kao. *Encyclopedia of Algorithms.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. (Zitiert auf Seite 51)

[KHW14] K. Kurzhals, F. Heimerl, D. Weiskopf. ISeeCube - Visual Analysis of Gaze Data for Video. In ACM, Herausgeber, *Proceedings of the 2014 Symposium on Eye Tracking Research and Applications (ETRA)*, S. 43–50. 2014. (Zitiert auf den Seiten 30 und 31)

[KW78] J. B. Kruskal, M. Wish. *Multidimensional scaling*, Band 11. Sage, 1978. (Zitiert auf Seite 32)

[KW13] K. Kurzhals, D. Weiskopf. Space-Time Visual Analytics of Eye-Tracking Data for Dynamic Stimuli. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 12(19):2129–2138, 2013. (Zitiert auf Seite 56)

[Lai12] J. Laird. *The Soar cognitive architecture.* MIT Press, 2012. (Zitiert auf Seite 22)

[McD23] W. McDougall. Outline of psychology. 1923. (Zitiert auf Seite 20)

[Mil62] B. Milner. Les troubles de la memoire accompagnant des lesions hippocampiques bilaterales. *Physiologie de l'hippocampe*, S. 257–72, 1962. (Zitiert auf Seite 20)

[Mil03] G. A. Miller. The cognitive revolution: a historical perspective. *Trends in cognitive sciences*, 7(3):141–144, 2003. (Zitiert auf Seite 18)

[NEP13] A. Neupert, T. Eikmeier, S. Plohmer. Dynamische Ontologie Visualisierung, 2013. (Zitiert auf Seite 62)

[NL13] S. Negru, S. Lohmann. A Visual Notation for the Integrated Representation of OWL Ontologies. In *WEBIST*, S. 308–315. 2013. (Zitiert auf den Seiten 24 und 25)

[NSA74] U. Neisser, W. Schlund, U. Aeschbacher. *Kognitive Psychologie*. Ernst Klett Verlag, 1974. (Zitiert auf Seite 17)

[NW70] S. B. Needleman, C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970. (Zitiert auf Seite 32)

[PB06] T. Pellegrini, A. Blumauer. *Semantic Web*. Springer, 2006. (Zitiert auf Seite 23)

[PDZ86] W. Pschyrembel, O. Dornblüth, C. Zink, Herausgeber. *Pschyrembel klinisches Wörterbuch Mit klinischen Syndromen und Nomina Anatomica*. De Gruyter, Berlin, 255. Auflage, 1986. (Zitiert auf Seite 13)

[Pin90] S. Pinker. *A theory of graph comprehension*, S. 73–126. 1990. (Zitiert auf Seite 27)

[RBR+14] M. Raschke, T. Blascheck, M. Richter, T. Agapkin, T. Ertl. Visual Analysis of Perceptual and Cognitive Processes. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAAP)*, Band 2014. 2014. (Zitiert auf Seite 15)

[RCE12] M. Raschke, X. Chen, T. Ertl. Parallel Scan-Path Visualization. In *Proceedings of the 2012 Symposium on Eye-Tracking Research and Applications*, Band 2012, S. 165–168. ACM, New York, NY, USA, 2012. (Zitiert auf den Seiten 15 und 16)

[RD05] D. C. Richardson, R. Dale. Looking To Understand: The Coupling Between Speakers' and Listeners' Eye Movements and Its Relationship to Discourse Comprehension. *Cognitive Science*, 29(6):1045–1060, 2005. (Zitiert auf den Seiten 29 und 30)

[Ruh12] Ruhr-Universität Bochum. Kernspintomographie, 2012. URL `http://www.radiologie.ruhr-uni-bochum.de/patient/institut/mrt`. (Zitiert auf Seite 64)

[Ryl49] G. Ryle. The concept of mind. 1949. (Zitiert auf Seite 20)

[SBF98] R. Studer, V. R. Benjamins, D. Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1):161–197, 1998. (Zitiert auf Seite 24)

[Squ04] L. R. Squire. Memory systems of the brain: a brief history and current perspective. *Neurobiology of learning and memory*, 82(3):171–177, 2004. (Zitiert auf den Seiten 20 und 21)

[Str96] G. Strube, Herausgeber. *Wörterbuch der Kognitionswissenschaft*. Klett-Cotta, Stuttgart, 1996. (Zitiert auf den Seiten 17, 18, 19 und 21)

[Str14] S. Strohmaier. *Visuelle Analyse von Eyetracking-Experimenten mit einer Vielzahl von Areas of Interest*. Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2014. (Zitiert auf den Seiten 72 und 79)

[SZM91] L. R. Squire, S. Zola-Morgan. The medial temporal lobe memory system. *Science*, 253(5026):1380–1386, 1991. (Zitiert auf Seite 20)

[Tob13] Tobii Technology. An introduction to eye tracking and Tobii Eye Trackers, 2013. URL http://www.tobii.com/en/eye-tracking-research/global/library/white-papers/tobii-eye-tracking-white-paper. (Zitiert auf Seite 14)

[Tul72] E. Tulving. Episodic and semantic memory. In E. Tulving, W. Donaldson, Herausgeber, *Organization of memory*, S. 381–403. Academic Press, New York, 1972. (Zitiert auf Seite 21)

[WHRK06] J. M. West, A. R. Haake, E. P. Rozanski, K. S. Karn. eyePatterns: software for identifying patterns and similarities across fixation sequences. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, S. 149–154. ACM, 2006. (Zitiert auf Seite 32)

[Win72] T. Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972. (Zitiert auf Seite 21)

[Win75] T. Winograd. Frame representations and the declarative/procedural controversy. *Representation and understanding: Studies in cognitive science*, S. 185–210, 1975. (Zitiert auf Seite 21)

[Yar67] A. L. Yarbus. *Eye Movements and Vision.* Plenum. New York., 1967. (Zitiert auf Seite 13)

Alle URLs wurden zuletzt am 16. 07. 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