Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3281

What now? - Entwurf und Implementierung einer intelligenten Dialogführung zum Ergänzen von unterspezifizierten Fragen

Geoffrey-Alexeij Heinze

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuer:	Harald Bosch, M. Sc. Dipl.-Inf. Dennis Thom
begonnen am:	7. Dezember 2011
beendet am:	7. Juni 2012
CR-Klassifikation:	E.1, H.4.2, H.5.2, I.2.4, J.3

Zusammenfassung

Da die Komplexität von Computerprogrammen immer weiter zunimmt, werden immer bessere Benutzungsoberflächen notwendig, um ihre Bedienbarkeit zu gewährleisten.

In dieser Diplomarbeit wurden verschiedene Methoden für die intelligente Dialogführung mit dem Benutzer untersucht und entwickelt. Die intelligente Dialogführung orientiert sich dabei an den Zielen des Benutzers, um ihn während seiner Arbeit mit der Benutzungsoberfläche gezielt unterstützen zu können.

Die in dieser Arbeit entwickelte Lösung wurde in einer Benutzungsoberfläche für den webbasierten Dienst PESCaDO umgesetzt. PESCaDO erlaubt es seinen Benutzern, personalisierte Berichte zu Wetter und anderen Umwelteinflüssen, wie zum Beispiel Pollenbelastungen, zu generieren. PESCaDO verwendet dabei eine Ontologie, um die Bedürfnisse des Benutzers mit den vorhandenen Daten abzugleichen.

In einer anschließenden Evaluation wurde die entwickelte Benutzungsoberfläche und die darin umgesetzten Methoden zur intelligenten Dialogführung untersucht, wobei die Wirksamkeit der entwickelten Lösung bestätigt werden konnte.

Abstract

With the ever increasing complexity of computer applications, it is necessary to constantly develop better user interfaces in order to ensure an acceptable level of usability.

This diploma thesis examines existing methods and develops new methods for intelligent user interfaces. These intelligent user interfaces orient themselves on the user's goals to provide targeted assistance.

The solution developed in this work has been implemented in a user interface for the web-based service PESCaDO. PESCaDO allows its users to generate personalized reports regarding certain environmental conditions, like the weather or the pollen concentration in the air. PESCaDO uses an ontology in order to compare the user's needs with the available data.

As a final step, the developed user interface and the contained methods for intelligent user interfaces have been tested in an evaluation. The evaluation proved the effectivity of the solution developed in this work.

Inhaltsverzeichnis

1 Einleitung **9**

2 Grundlagen **11**

2.1 PESCaDO 11
2.1.1 Architektur von PESCaDO 12
2.1.2 Benutzungsoberfläche von PESCaDO 14
2.2 Der Ontologie-Begriff 16
2.2.1 Die PESCaDO Ontologie 17
Struktur der Problem Description Language 17
2.3 Bestehende Lösungen 20
2.3.1 Definition „Intelligente Dialogführung" 20
2.3.2 Kategorisierung der Systeme 20
2.3.3 Maschinelle Lernverfahren 22

3 Lösung **25**

3.1 Problemstellung 25
3.2 Allgemeine Beschreibung der Lösung 25
3.3 Generierung der Regeln 27
3.3.1 Struktur einer Regel 28
3.3.2 Gewichtungsfaktoren 28
3.3.3 Ableitung von Regeln aus Beziehungstypen 29
3.4 Gewichten und Auswerten der Eingaben 31
3.4.1 Berechnung des Gewichts einer Eingabe 32
3.4.2 Auswerten der Eingaben 33
3.5 Möglichkeiten zur Gestaltung der Benutzungsoberfläche 34
3.5.1 Evaluation der alten Benutzungsoberfläche 34
3.5.2 Aufbau der Benutzungsoberfläche 34
3.5.3 Gestaltung der Eingabefelder 38
Erforderliche Eingabefelder 38
Verbotene Eingabefelder 39
Nicht ausgefüllte erforderliche Eingabefelder 39
Fehlerhaft ausgefüllte Eingabefelder 40
Optionale Eingabefelder 42
Dynamische Prüfung auf nicht ausgefüllte erforderliche Eingabefelder 42
Andere Gestaltungs- und Hervorhebungsmöglichkeiten 43
3.5.4 Nicht verwendete Methoden zur Gestaltung und Hervorhebung 44

4 Implementierung **47**
4.1 Architektur der Benutzungsoberfläche . . . 47
4.2 Implementierung der Benutzungsoberfläche . . . 48
4.2.1 Helferklassen . . . 49
4.2.2 Presenter . . . 51
4.2.3 View . . . 51
DialogWidget und FieldWidget . . . 52
DialogPage . . . 52
4.2.4 QueryManager . . . 53
4.2.5 DataStore . . . 53
4.2.6 Diviner . . . 53
4.2.7 Paintbox . . . 54
4.2.8 Ruleset . . . 54
4.3 Generierung der Regeln . . . 55

5 Evaluation **57**
5.1 Geplanter Ablauf der Evaluation . . . 57
5.1.1 Varianten und Aufgaben . . . 58
5.1.2 Fragebogen . . . 60
5.1.3 Messwerte . . . 62
5.2 Anpassungen der Benutzungsoberfläche . . . 62
5.3 Durchführung und Ergebnisse der Evaluation . . . 63
5.3.1 Verteilung der Evaluationsvarianten . . . 63
5.3.2 Basisfragen . . . 64
5.3.3 Fragebogen . . . 65
5.3.4 Für Anfrage benötigte Zeit . . . 72
5.3.5 Häufigkeit der angezeigten Dialoge . . . 76
5.3.6 Häufigkeit von hervorgehobenen Fehlern . . . 79
5.3.7 Anmerkungen der Testpersonen . . . 80
5.4 Auswertung und Erkenntnisse der Evaluation . . . 83
5.4.1 Überprüfung der Thesen . . . 83
5.4.2 Signifikanz des Ergebnisses . . . 86
5.4.3 Relevanz des Ergebnisses . . . 87
5.4.4 Erkenntnisse der Evaluation . . . 87

6 Zusammenfassung und Ausblick **89**

Literaturverzeichnis **91**

Abbildungsverzeichnis

2.1 Architektur von PESCaDO 12
2.2 Alte Benutzungsoberfläche von PESCaDO 15

3.1 Funktionsweise der Lösung 26
3.2 Neue Benutzungsoberfläche von PESCaDO 35
3.3 Schematische Darstellung eines Eingabefelds 38
3.4 Erforderliches Eingabefeld 38
3.5 Verbotenes Eingabefeld 39
3.6 Nicht ausgefülltes erforderliches Eingabefeld 39
3.7 Fehlerhaft ausgefülltes Eingabefeld 41
3.8 Hinweis auf nicht ausgefülltes erforderliches Eingabefeld 42
3.9 Dynamisches Anzeigen abhängiger Felder 43
3.10 Eingabefeld mit Beschreibungstext 43
3.11 Übersicht der Dialogseiten 44

4.1 Architektur der neuen Benutzungsoberfläche von PESCaDO 48

5.1 Punkteverteilung der Fragen 1 bis 10 für die alte Benutzungsoberfläche 66
5.2 Punkteverteilung der Fragen 1 bis 10 für die neue Benutzungsoberfläche ohne Hervorhebung 67
5.3 Punkteverteilung der Fragen 1 bis 10 für die neue Benutzungsoberfläche . . . 68
5.4 Punkteverteilung der Fragen 11 bis 22 für die neue Benutzungsoberfläche . . . 69
5.5 Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche ohne Hervorhebungen 69
5.6 Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche 70
5.7 Für Anfragen benötigte Zeit bei der ersten Aufgabe 74
5.8 Für Anfragen benötigte Zeit bei der zweiten Aufgabe 75
5.9 Häufigkeit der Dialoge bei alter Benutzungsoberfläche 77
5.10 Häufigkeit der Dialoge bei neuer Benutzungsoberfläche ohne Hervorhebung . 77
5.11 Häufigkeit der Dialoge bei neuer Benutzungsoberfläche 78
5.12 Häufigkeit von hervorgehobenen Fehlern 79

Tabellenverzeichnis

5.1 Gesamtverteilung der ausgegebenen Varianten 64
5.2 Verteilung der ausgegebenen Varianten und abgebrochene Aufgaben der 56 vollständigen Evaluationsdurchläufe 64
5.3 Alter der Teilnehmer 64
5.4 Computerkenntnisse der Teilnehmer 65
5.5 Punkteverteilung der Fragen 1 bis 10 für die alte Benutzungsoberfläche 66
5.6 Punkteverteilung der Fragen 1 bis 10 für die neue Benutzungsoberfläche ohne Hervorhebungen 67
5.7 Punkteverteilung der Fragen 1 bis 22 für die neue Benutzungsoberfläche . . . 70
5.8 Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche ohne Hervorhebungen 70
5.9 Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche 71
5.10 Für Anfragen benötigte Zeit bei der ersten Aufgabe 72
5.11 Für Anfragen benötigte Zeit bei der zweiten Aufgabe 73
5.12 Häufigkeit der Dialoge bei alter Benutzungsoberfläche 76
5.13 Häufigkeit der Dialoge bei neuer Benutzungsoberfläche ohne Hervorhebungen 77
5.14 Häufigkeit der Dialoge bei neuer Benutzungsoberfläche 78
5.15 Häufigkeit hervorgehobener Fehler 80
5.16 t-Werte der Fragen 1 bis 24 und 29 87

1 Einleitung

Die Leistungsfähigkeit und Komplexität von Computern und Anwendungen entwickelt sich rasant, während der Mensch als Benutzer sich nur relativ langsam entwickelt. Daraus entsteht das Bedürfnis, für komplexe Anwendungen möglichst einfach zu bedienende Benutzungsoberflächen (im Nachfolgenden auch mit UI – vom englischen Begriff User Interface – abgekürzt) zu entwickeln. Diese Benutzungsoberflächen sollen den Benutzer bei seiner Arbeit mit der Anwendung auf vielfältige Weise unterstützen. Sie sollen zum einen dabei helfen, Eingaben zu vermeiden, die zu einem Fehler führen würden. Zum anderen soll der Benutzer dabei unterstützt werden, die Eingaben zu finden, die zu dem vom Benutzer gesuchten Ergebnis führen. Gleichzeitig soll die Kontrolle und Verantwortung jedoch beim Benutzer bleiben [Höoo]. Aus diesen Anforderungen wurden bereits verschiedenste Methoden zur Dialogführung mit dem Benutzer entwickelt.

Ziel dieser Diplomarbeit ist die Entwicklung und Implementierung einer Lösung zur intelligenten Dialogführung für den webbasierten Dienst PESCaDO. Mit PESCaDO kann der Benutzer Anfragen erstellen und sich so individuell zusammengestellte Umwelt- und Wetterauskünfte erzeugen lassen. Mit Informationen über den Benutzer werden diese Auskünfte zusätzlich auf die Bedürfnisse des Benutzers zugeschnitten, um zum Beispiel eine vom Benutzer angegebene Pollenallergie im angezeigten Ergebnis berücksichtigen zu können. Da das Erstellen dieser Anfragen das Ausfüllen diverser Eingabefelder erfordert, die sich zudem gegenseitig bedingen und in manchen Kombinationen zu keinem oder einem fehlerhaften Ergebnis führen, soll der Benutzer bei diesem Schritt durch eine intelligente Dialogführung unterstützt werden.

Die gültigen Kombinationen von Eingaben werden von PESCaDO in einer Ontologie beschrieben. Eine Ontologie ist eine formale Darstellung von Begriffen und deren Beziehungen zueinander. Auf diese Weise kann mit einer Ontologie Wissen über die enthaltenen Begriffe in einer computerlesbaren Form gespeichert und weitergegeben werden.

Im Rahmen dieser Diplomarbeit wurde daher zunächst untersucht, wie die in der Ontologie enthaltenen Informationen genutzt werden können, um die Eingaben des Benutzers zu validieren und ihn dabei zu unterstützten, gültige Anfragen zu erstellen. Anschließend wurde betrachtet, wie diese Informationen genutzt werden können, um den Benutzer besser durch den Dialog zum Erstellen einer Anfrage zu begleiten und für ihn relevante Informationen anzuzeigen. Die dabei entwickelten Methoden wurden dann in einer Benutzungsoberfläche für PESCaDO implementiert. Abschließend wurde mit dieser Oberfläche eine Benutzerstudie durchgeführt, bei der die zuvor entwickelte Lösung getestet und auf ihre Tauglichkeit hin untersucht wird.

Dieser Reihenfolge folgt auch die Gliederung dieser Arbeit. So werden in **Kapitel 2 – Grundlagen** zunächst die für diese Arbeit relevanten Grundlagen erläutert. Das umfasst neben einem Überblick über bereits existierende Methoden zur intelligenten Dialogführung auch das PESCaDO-System und dessen Ontologie, auf dem die in dieser Arbeit entwickelte Lösung aufsetzt.
In **Kapitel 3 – Lösung** werden anschließend die in dieser Diplomarbeit entwickelten Methoden zur intelligenten Dialogführung für PESCaDO beschrieben.
Kapitel 4 – Implementierung beschreibt nachfolgend, wie die entwickelte Lösung umgesetzt und die Benutzungsoberfläche für PESCaDO implementiert wurde.
In **Kapitel 5 – Evaluation** werden die auf dieser Benutzungsoberfläche durchgeführte Benutzerstudie und die aus der Studie gewonnenen Erkenntnisse beschrieben.
Den Abschluss bildet **Kapitel 6 – Zusammenfassung und Ausblick**, das die Ergebnisse dieser Diplomarbeit zusammenfasst und einen Ausblick auf weitere Entwicklungsmöglichkeiten gewährt.

2 Grundlagen

In diesem Kapitel werden die für die in dieser Arbeit entwickelte Lösung relevanten Grundlagen beschrieben. Zunächst wird dabei auf PESCaDO und seine Ontologie eingegangen, welche Grundlage und Rahmen dieser Arbeit bilden. Anschließend werden in diesem Kapitel bereits existierende Methoden für intelligente Dialogführung betrachtet und auf ihre Vor- und Nachteile hin untersucht.

2.1 PESCaDO

Das PESCaDO (Personalized Environmental Service Configuration and Delivery Orchestration) Projekt[1] [BTE11, WVT+11] ist ein von der EU gefördertes Projekt über drei Jahre, das von sieben verschiedene Instituten getragen wird. Ziel von PESCaDO ist, die im Internet aus verschiedensten Quellen verfügbaren Wetter- und Umweltdaten zu sammeln, aufzubereiten und über eine zentrale und einheitlich zu bedienende Schnittstelle anzubieten. Wichtigster Punkt hierbei ist die Anpassung der Ergebnisse an die Bedürfnisse des Benutzers.

Der Benutzer kann dann PESCaDO verwenden, um sich zum Beispiel für das nächste Wochenende eine Wettervorhersage für ein bestimmtes Gebiet erstellen zu lassen, in dem er eine sogenannte Anfrage erstellt und diese an den Server sendet. Oder er verwendet PESCaDO um zu prüfen, ob er auf dem Nachhauseweg von der Arbeit mit Glatteis rechnen muss. Oder er erstellt eine der anderen, möglichen Anfragen, auf die im Nachfolgenden noch genauer eingegangen wird. Der Server wiederum generiert eine zur Anfrage passende Antwort und sendet diese zurück an den Benutzer.

Der Vorteil von PESCaDO liegt dabei darin, dass der Benutzer nur den Umgang mit einem Dienst erlernen muss, statt mit mehreren verschiedenen Diensten. Während die verschiedenen Dienste meist eine jeweils eigene, spezifische Art der Anfrageerstellung haben, bietet PESCaDO für verschiedene Anfragen jeweils die selbe, einheitliche Art der Anfrageerstellung. Zudem entfällt für den Benutzer bei der Verwendung von PESCaDO die Notwendigkeit, einen jeweils zu seinen Interessen passenden Dienst suchen zu müssen.

Über Informationen, die der Benutzer in seinem Profil hinterlegt, schneidet PESCaDO zusätzlich die Antwort auf die Anfrage eines Benutzers noch weiter auf seine Bedürfnisse zu. So kann ein Benutzer beispielsweise in seinem Profil angegeben, dass er gegen Birkenpollen allergisch ist. Formuliert dieser Benutzer nun in PESCaDO eine Anfrage, bei der er angibt

[1] http://www.pescado-project.eu/

in einem bestimmten Zeitraum an einem bestimmten Ort wandern gehen zu wollen, liefert PESCaDO in seinem Ergebnis nicht nur Informationen zu den Wetterverhältnissen, sondern auch über die zu erwartende Pollenbelastung und andere Werte, die aufgrund der Eingaben des Benutzers als relevant eingestuft wurden.

2.1.1 Architektur von PESCaDO

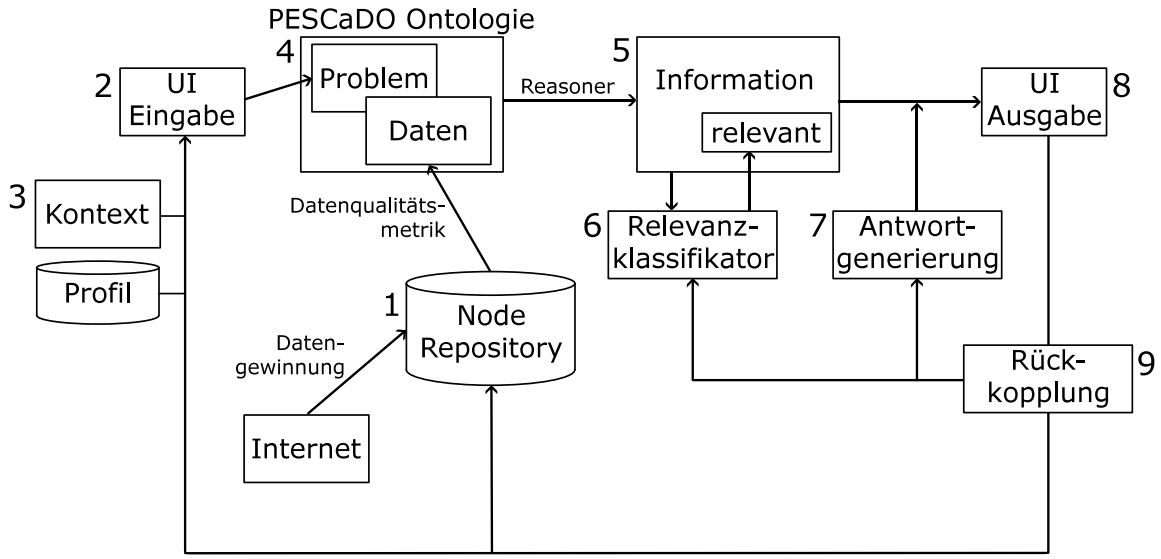


Abbildung 2.1: Architektur von PESCaDO

Um dem Benutzer Informationen zum Wetter und anderen Umweltaspekten liefern zu können, umfasst PESCaDO verschiedene Komponenten, deren Zusammenspiel in Abbildung 2.1 skizziert ist. Zunächst benötigt PESCaDO selbst Zugriff auf Webseiten und Webservices, die die Wetter- und Umweltdaten anbieten. Diese Webseiten und -services werden innerhalb von PESCaDO als Environmental Service Nodes (oder auch nur Service Nodes) bezeichnet. Die Environmental Service Node Discovery Komponente von PESCaDO sucht kontinuierlich nach solchen Service Nodes und speichert die gefundenen in einem Environmental Service Node Repository ab (**1** in Abbildung 2.1). Die relevanten Daten, wie zum Beispiel Temperatur oder Grad der Bewölkung, werden mit Hilfe verschiedener Techniken wie Texterkennung automatisch extrahiert.

Da die so gewonnenen Informationen jedoch je nach Quelle in sehr unterschiedlichen Formaten vorliegen, wird eine Zwischenschicht benötigt, in der die Informationen aus allen Quellen in einem einheitlichen Format gespeichert werden können. Nur so ist es effizient möglich, mit diesen Daten zu arbeiten. Diese Zwischenschicht bei PESCaDO ist die PESCaDO Ontologie. Dabei ist zu beachten, dass in der Ontologie lediglich eine Beschreibung der auf

den einzelnen Service Nodes verfügbaren Informationen, nicht jedoch die Informationen selbst, gespeichert werden.

Möchte der Benutzer nun wie im vorhergehenden Beispiel eine Anfrage formulieren, verwendet er dazu die Benutzungsoberfläche von PESCaDO (**2**). Dort kann er auf einfache Weise sein Anliegen beschreiben, wobei zusätzlich vom Benutzer im Profil angegebene oder aus dem Kontext gewonnene Informationen (**3**) mit einfließen. Die von ihm getätigten Eingaben werden anschließend mit Hilfe der Problem Description Language (kurz PDL) in einem von PESCaDO interpretierbaren Satz formuliert. Mit diesem PDL Satz und der PESCaDO Ontologie (**4**) können die zur Beantwortung der Anfrage relevanten Service Nodes ermittelt werden. Die PDL ist ein Teil der PESCaDO Ontologie und wird mit dieser in Kapitel 2.2 erläutert.

Von den ermittelten Service Nodes werden dann die benötigten Informationen abgerufen und extrahiert (**5**). Die gewonnenen Informationen werden in ein einheitliches Format gebracht und auf ihre Relevanz für den Benutzer hin überprüft (**6**). Anschließend werden die relevanten Informationen aufbereitet (**7**) und dem Benutzer als Ergebnis seiner Anfrage präsentiert (**8**). Beim Aufbereiten wird aus den fragmentierten Informationen ein leicht lesbarer Text generiert. Zusätzlich werden die Informationen auf verschiedene Weisen grafisch dargestellt.

Ist der Benutzer mit dem Ergebnis oder der Darstellung des Ergebnisses nicht zufrieden, kann er über Interaktion mit dem Ergebnis die Darstellung oder auch die zu dem Ergebnis führende Anfrage anpassen (**9**), um die von ihm gesuchten Informationen zu erhalten. Die durch diese Rückkopplung gewonnenen Informationen über den Benutzer werden gespeichert und können bei der nächsten Anfrage genutzt werden, um das Ergebnis besser an die für den Benutzer relevanten Aspekte anzupassen.

Da die meisten Komponenten von PESCaDO für diese Arbeit nur wenig Relevanz besitzen, wird ihre Funktionsweise im Nachfolgenden lediglich grob erläutert. Auf die relevanten Komponenten wird in jeweils einem eigenen Abschnitt eingegangen.

Datengewinnung Zur Datengewinnung durchsucht die Environmental Service Node Discovery Komponente (ESND) das Internet und verwendet dabei zwei verschiedene Ansätze, um Service Nodes für PESCaDO zu finden.
Zum einen wird ein Crawler eingesetzt, der mit Hilfe von maschinellen Lernverfahren und einer Reihe von vordefinierten Seiten trainiert wurde. Zum anderen werden allgemeine Suchmaschinen (wie zum Beispiel Google) mit bestimmten Schlüsselwörtern zum Finden von Service Nodes verwendet. Dank der Hilfe verschiedener Übersetzungswerkzeuge kann diese Suche auch in verschiedenen Sprachen durchgeführt werden.

Aus den so gefunden Webseiten und -services werden anschließend die enthaltenen Daten extrahiert und auf ihre Eignung hin untersucht. Hierbei können auch mehrere Quellen miteinander verglichen werden, wodurch zum Beispiel stark vom Durchschnitt abweichende Quellen aussortiert werden können. Die als brauchbar eingestuften Quellen werden dann vom ESND im Node Repository gespeichert.

Reasoner Mit Hilfe des Reasoners werden das vom Benutzer beschriebene Problem in PDL und die vom ESND gelieferten Informationen zu den Service Nodes miteinander verglichen, um so die Service Nodes zu finden, deren Informationen zum Problem des Benutzers passen. Von den geeigneten Service Nodes werden dann die aktuellen Informationen abgefragt.

Relevanzklassifikator Der Relevanzklassifikator wählt aus den verfügbaren Daten diejenigen aus, die aufgrund der Anfrage des Benutzers (in PDL) und den zum Benutzer hinterlegten Informationen angezeigt werden sollen.

Antwortgenerierung Aus den als relevant eingestuften Informationstücken und mit Hilfe benutzerspezifischer Informationen werden in der Komponente zur Antwortgenerierung die anzuzeigenden Daten aufbereitet. Ebenso wird aus den Informationen ein leicht verständlicher Text generiert, der neben der grafischen Darstellung des Ergebnisses angezeigt wird.

UI Ausgabe Die Ausgabe der UI (oder auch Visualisierung des Ergebnisses) stellt die zuvor generierte Antwort für den Benutzer dar. Hierzu werden zum einen textuelle Elemente verwendet, zum anderen werden die Informationen in grafischer Weise, zum Beispiel mit Symbolen oder verschiedenen Kartentypen, dargestellt.

Rückkopplung Nach dem der Benutzer die Antwort zu seiner Anfrage erhalten hat, kann er direkt und indirekt Rückmeldung geben, wie zufrieden er mit dem Ergebnis war – sowohl inhaltlich als auch mit der Visualisierung des Ergebnisses. Diese Informationen werden bei späteren Anfragen genutzt, um das Ergebnis besser an die Anforderungen des Benutzers anzupassen.

2.1.2 Benutzungsoberfläche von PESCaDO

Die aktuell verwendete Benutzungsoberfläche von PESCaDO (Abbildung 2.2) wird in der Abteilung für Visualisierung und Interaktive Systeme (VIS) an der Universität Stuttgart entwickelt und ist dafür verantwortlich, aus den Eingaben des Benutzers seine Anfrage in PDL zu formulieren und die anschließend für die Anfrage zurückgelieferten Ergebnisse anzuzeigen.

Im Rahmen dieser Arbeit wurde die Benutzungsoberfläche von PESCaDO komplett neu gestaltet, die in diesem Kapitel vorgestellte Oberfläche wird daher im Nachfolgenden als die alte Benutzungsoberfläche bezeichnet. Sofern im Nachfolgenden lediglich von einer Benutzungsoberfläche gesprochen wird, ist damit stets die in Kapitel 3.5 vorgestellte neue Benutzungsoberfläche gemeint.

Die alte Benutzungsoberfläche gliedert sich in die folgenden drei Bereiche:

Mitte In der Mitte (3 in Abbildung 2.2) der alten Benutzungsoberfläche befindet sich eine Landkarte. Auf dieser Landkarte kann der Benutzer zum einen das Gebiet auswählen, auf das sich seine Anfrage bezieht, zum anderen werden hier die Ergebnisse seiner Anfrage visualisiert.

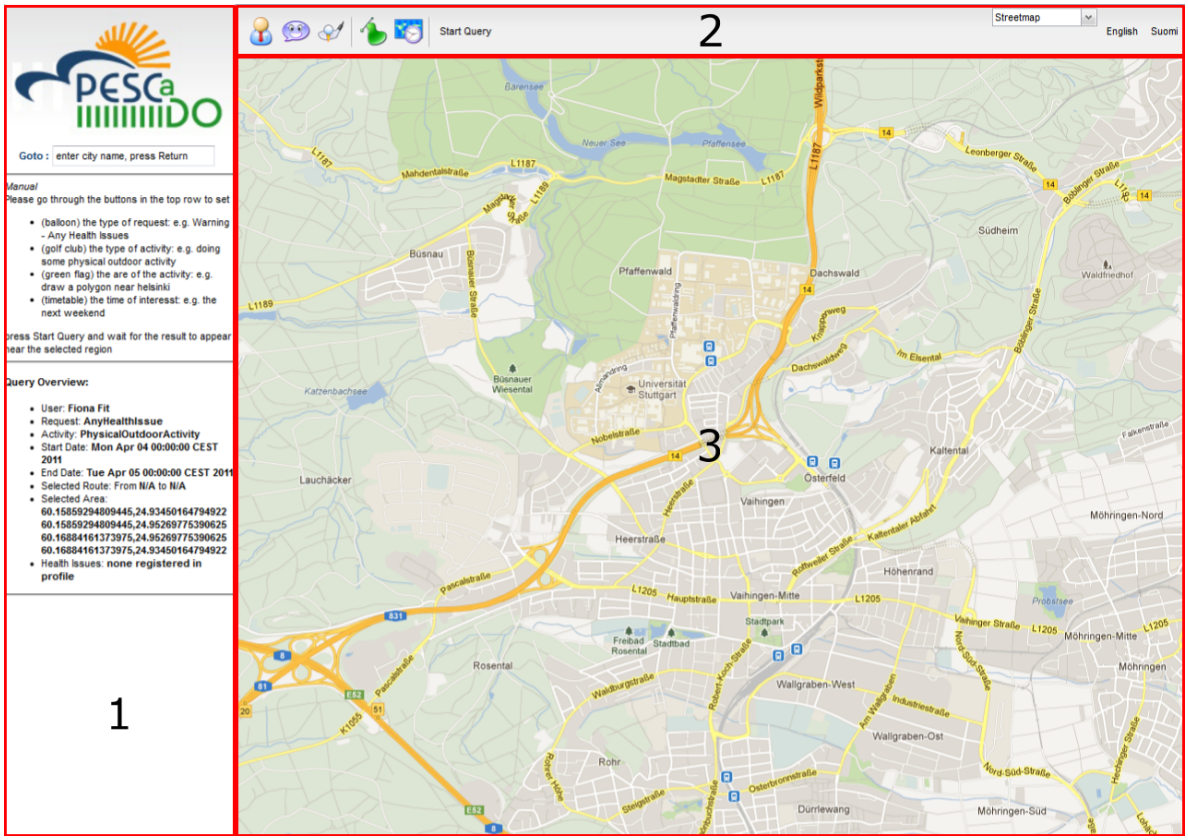


Abbildung 2.2: Alte Benutzungsoberfläche von PESCaDO

Linker Rand Am linken Rand (**1** in Abbildung 2.2) befindet sich direkt unterhalb des PESCaDO Logos ein Eingabefeld, in das der Benutzer den Namen einer Stadt eingeben kann, um die Karte in der Mitte auf diese Stadt zu zentrieren.
Unterhalb des Eingabefeldes werden dem Benutzer die Funktionen der am oberen Rand angeordneten Schaltflächen erklärt.
Unterhalb der Erklärungen sieht der Benutzer die Details seiner aktuellen Anfrage. Diese beinhalten zum Beispiel den Typ der Anfrage oder die für die Anfrage ausgewählte Aktivität. Sobald der Benutzer seine Anfrage ändert, ändern sich auch die hier angezeigten Informationen.

Oberer Rand Am oberen Rand (**2** in Abbildung 2.2) befinden sich auf der linken Seite sechs Schaltflächen (von links nach rechts):

- Mensch: hier kann der Benutzer ein Benutzerprofil wählen.
- Sprechblase: hier kann der Benutzer den Anfragetyp festlegen.
- Golfschläger: hier kann der Benutzer eine Aktivität auswählen.

- Grüne Fahne: hier wählt der Benutzer das für die Anfrage relevante Gebiet auf der Karte aus.
- Uhr: hier wählt der Benutzer den für die Anfrage relevanten Zeitraum aus.
- Anfrage senden: hiermit sendet der Benutzer seine Anfrage an den Server, wo sie verarbeitet wird. Sobald das Ergebnis vorliegt, wird es in der Mitte angezeigt.

Die ersten fünf dieser Schaltflächen öffnen jeweils ein separates, verschiebbares Dialogfenster innerhalb des Browserfensters. In diesen Dialogfenstern kann der Benutzer dann die entsprechenden Eingaben tätigen.

Über die Schaltflächen auf der rechten Seite am oberen Rand kann die Darstellungsart der Karte angepasst werden (Kartenansicht, Satellitenbilder oder Satellitenbilder mit eingeblendeter Karte) und die Sprache der alten Benutzungsoberfläche gewählt werden.

2.2 Der Ontologie-Begriff

Der Begriff „Ontologie" stammt ursprünglich aus dem Bereich der Philosophie [BS91], wo er die Lehre vom Seienden beschreibt. Im Gebiet der Informatik wird die Bezeichnung Ontologie für formale Sammlungen von Begriffen und den Beziehungen zwischen diesen Begriffen verwendet. Ontologien werden genutzt, um Wissen in maschinenlesbarer Form zu repräsentieren und einen einfachen Austausch dieses Wissens zwischen verschiedenen Anwendungen zu ermöglichen. Im Gegensatz zu einer Taxonomie, die Begriffe hierarchisch gliedert, bildet eine Ontologie ein Netzwerk über die Beziehungen zwischen den einzelnen Begriffen.

Die für diese Arbeit wichtigsten Bestandteile einer Ontologie [HKRS08] sind:

Begriffe (auch als Klassen bezeichnet) beschreiben eine Menge gemeinsamer Eigenschaften, die alle einem Begriff zugeordneten Objekte aufweisen. Die Klassen können zudem in einer Klassenhierarchie mit Über- und Unterklassen angeordnet werden. Dabei können die Kindklassen Eigenschaften ihrer Elternklassen erben. Zwei Beispielklassen wären *Stadt* oder *Land*.

Instanzen werden aus Klassen erzeugt und repräsentieren individuelle Objekte einer Klasse. Sie besitzen eine individuelle Ausprägung der durch die Klasse beschriebenen gemeinsamen Eigenschaften und werden daher auch als Individuen bezeichnet. Eine Beispielinstanz wäre die Instanz *Stuttgart* von der Klasse *Stadt*.

Relationen stellen die Beziehungen zwischen den Begriffen dar. Ein Beispiel wäre die Relation *befindet sich in* die zwischen den Begriffen *Stadt* und *Land* besteht. Beziehungen zwischen Klassen gelten ebenfalls für aus den Klassen erstellte Instanzen. Zusätzlich können Beziehungen zwischen einzelnen Instanzen beschrieben werden.

Da ausführliche Ontologien sehr schnell an Umfang zulegen, sind die meisten verwendeten Ontologien auf einen kleinen, dem Einsatzzweck angepassten Begriffsraum beschränkt. Für viele Bereiche, zum Beispiel in der Medizin, gibt es bereits existierende Ontologien, wodurch der Aufwand zur Erstellung einer eigenen Ontologie gespart werden kann. Jedoch ist zu beachten, dass zwei Ontologien zum selben Thema, je nach den Intentionen der Ersteller, durchaus unterschiedlich ausfallen können.

Zur Beschreibung von Ontologien gibt es verschiedene Sprachen, unter anderem das RDF-Schema, DAML+OIL oder die vom W3C propagierte Web Ontology Language (kurz OWL) [OWL]. Für jede der Sprachen gibt es einen oder mehrere Editoren, mit denen die Ontologie erstellt und gepflegt werden kann.

Die von PESCaDO verwendete Ontologie ist in OWL beschrieben, für die Arbeit mit der Ontologie wurden die Programme Protégé[2] und TopBraid Composer[3] verwendet.

2.2.1 Die PESCaDO Ontologie

Die PESCaDO Ontologie [Ros12] dient zum einen, wie zuvor bereits erwähnt, als Mittelschicht, um die von den verschiedenen Environmental Service Nodes verfügbaren Daten einheitlich darstellen zu können. Die hierfür notwendigen Elemente der Ontologie sind im Nachfolgenden jedoch nur von geringer Bedeutung und werden daher nicht genauer betrachtet.

Zum anderen definiert die PESCaDO Ontologie die für diese Arbeit relevante Problem Description Language PDL [RSW+10], mit der eine Anfrage des Benutzers formal dargestellt und zur Verarbeitung an den Server gesendet werden kann. Durch die PDL werden dabei alle gültigen Anfragen definiert. Erst mit Hilfe der formalen Darstellung durch die PDL kann der Reasoner (siehe Kapitel 2.1.1) die Anfrage des Benutzers mit den Service Nodes abgleichen, um so geeignete Informationsquellen zu finden.

Dieser Arbeit liegt die PESCaDO Ontologie mit dem Stand vom 29. Januar 2012 zu Grunde. Alle Änderungen, die anschließend an der PESCaDO Ontologie vorgenommen wurden, sind in dieser Arbeit nicht berücksichtigt worden.

Struktur der Problem Description Language

Die Basisklasse der PDL ist **Problem**, jede Anfrage ist eine Instanz von *Problem* und kann ein oder mehrere *Request* besitzen.

Die Klasse **Request** stellt eine allgemeine Anfrage dar. Die *Request* Klasse beinhaltet stets genau einen Start- und Endzeitpunkt, mit denen der Zeitraum für die Anfrage eingeschränkt

[2] http://protege.stanford.edu/
[3] http://www.topquadrant.com/products/TB_Composer.html

werden, sowie genau einen *User*, welcher weitere, benutzerspezifische Informationen enthalten kann. Zusätzlich kann *Request* ein oder mehrere *GeoArea* (das für die Anfrage relevante Gebiet oder eine Route) enthalten. Die direkten Kindklassen von *Request* dienen zur Ordnung der verschiedenen Anfragetypen. Für die eigentliche Anfrage wird stets eine Kindklasse der direkten Kindklassen von *Request* verwendet, die die Art der Anfrage näher beschreiben und gegebenenfalls die geerbten Eigenschaften ergänzen oder weiter einschränken.

Die Klasse **User** besitzt diverse Kindklassen, die verschiedene Benutzergruppen beschreiben. Die beiden wichtigsten Kindklassen sind *AdministrativeUser* und *EndUser*. *AdministrativeUser* wird immer dann verwendet, wenn die Ergebnisse der Anfrage allgemeiner Natur sind und nicht an den Benutzer angepasst werden müssen. *EndUser* wird verwendet, um die Ergebnisse der Anfrage auf den Benutzer und dessen, zum Beispiel in seinem Profil angegebenen, individuellen Eigenschaften zuzuschneiden. Zum momentanen Zeitpunkt können die Benutzer noch keine eigenen Profile anlegen. Stattdessen werden die in der Ontologie enthaltenen Instanzen *Arnold Admin* von *AdministrativeUser* sowie *Fiona Fit* und *Allen Allergic* von *EndUser* verwendet. Für den Benutzer *Allen Allergic* wurde dabei eine Pollenallergie im Profil hinterlegt.

Die Klasse **GeoArea** beschreibt geografische Bereiche und wird durch drei Kindklassen näher spezifiziert. Die Kindklasse *Region* beschreibt ein flächiges Gebiet, *Route* eine Strecke und *SpotLocation* einen einzelnen Punkt.

Die Kindklassen von *Request* sind

InstructionRequest weist keine Änderungen zu *Request* auf und besitzt genau eine weitere Kindklasse:

- **SuggestAdministrativePlan** erlaubt keine *Activity* und erfordert einen von *AdministrativeUser* abgeleiteten Benutzer. Mit *SuggestAdministrativePlan* können allgemeine Informationen zu Temperatur, Niederschlag, Wind und Luftqualität angefordert werden.

ReportRequest weist keine Änderungen zu *Request* auf und besitzt die folgenden Kindklassen:

- **CheckAirQualityLimits** erlaubt keine *Activity* und erfordert einen von *AdministrativeUser* abgeleiteten Benutzer. Es können nur Gebiete vom Typ *Region* verwendet werden. Mit *CheckAirQualityLimits* können allgemeine Informationen zur Luftqualität angefordert werden.
- **CheckBlackIceCondition** erlaubt keine *Activity* und erfordert einen von *AdministrativeUser* abgeleiteten Benutzer. Mit *CheckBlackIceCondition* können allgemeine Informationen zur Glatteisgefahr angefordert werden.
- **CompareAirQualityInMultipleRegions** erfordert eine *Activity* vom Typ *LongTermStaying* und einen von *EndUser* abgeleiteten Benutzer. Zudem sind mindesten zwei *GeoArea* erforderlich. Mit *CompareAirQualityInMultipleRegions* kann die Luftqualität von zwei oder mehr Gebieten verglichen werden.

ReportAirQualityForecast erlaubt keine *Activity* und erfordert einen von *AdministrativeUser* abgeleiteten Benutzer. Mit *ReportAirQualityForecast* kann eine allgemeine Vorhersage der Luftqualität angefordert werden.

WarningRequest weist keine Änderungen zu *Request* auf und besitzt die folgenden Kindklassen:

AnyHealthIssue erfordert eine *Activity*, die nicht vom Typ *LongTermStaying* ist, sowie einen von *EndUser* abgeleiteten Benutzer. Mit *AnyHealthIssue* können auf den Benutzer zugeschnittene Informationen zu möglichen gesundheitlichen Problemen angefordert werden, die im angegebenen Gebiet im angegebenen Zeitraum bei der angegebenen Aktivität auftreten können.

AnyRestrictionForPrivateTransport erlaubt keine *Activity* und erfordert einen von *EndUser* abgeleiteten Benutzer. Mit *AnyRestrictionForPrivateTransport* können auf den Benutzer zugeschnittene Informationen zu möglichen Behinderungen bei der Fortbewegung angefordert werden.

WarningDueToEnvironmentalConditions erlaubt keine *Activity* und erfordert einen von *EndUser* abgeleiteten Benutzer. Mit *WarningDueToEnvironmentalConditions* können allgemeine Informationen zu Wetter- und Umweltwarnungen angefordert werden.

Alle Kindklassen von *Request* besitzen zudem meist mehrere Relationen *hasRelevantAspect*, mit denen für diesen Anfragetyp relevante Aspekte der Ontologie, wie zum Beispiel die Konzentration bestimmter Gase oder Pollen in der Luft, mit dem Anfragetyp verknüpft werden. Da diese Aspekte jedoch hauptsächlich zur Bearbeitung der Anfrage verwendet werden und vom Benutzer nicht beeinflusst werden können, wird im Nachfolgenden nicht näher auf sie eingegangen.

Die Klasse **Activity** beschreibt die vom Benutzer für seine Anfrage gewählte Aktivität. Die Aktivität wird verwendet, um zum Beispiel aufgrund des hohen Alters des Benutzers und einer hohen Ozonbelastung bei einer körperlich anstrengenden Aktivität eine Warnung auszugeben. Bei einer anderen Aktivität oder einem jüngeren Benutzer wird jedoch eventuell noch keine Warnung erzeugt. *Activity* hat vier Kindklassen, von denen jeder, abgesehen von *AttendingOpenAirEvent*, jeweils eine Klasse zugeordnet ist, aus der die zur Kindklasse gehörenden Instanzen erzeugt werden. Sofern mindestens eine Instanz zugeordnet ist, wählt der Benutzer diese aus, ansonsten wählt er direkt die Kindklasse von *Activity*.

AttendingOpenAirEvent ist die einzige Kindklasse ohne Instanzen und beschreibt eine Veranstaltung im Freien.

LongTermStaying beschreibt einen längeren Aufenthalt. Die zugeordnete Klasse *LongTermStayingType* besitzt die Instanzen *Going On Holiday* und *Living*

PhysicalOutdoorActivity beschreibt eine physische Aktivität im Freien. Die zugeordnete Klasse *PhysicalOutdoorActivityType* besitzt die Instanzen *Cycling*, *Hiking*, *Jogging*, *Mountainbiking*, *Nordic Walking*, *Playing Tennis*, *Skiing* und *Walking*

Travelling beschreibt die Überwindung einer Distanz. Die zugeordnete Klasse *TravelModality* besitzt die Instanzen *Cycling*, *Car*, *Feet* und *Public Transport*

Durch diese Klassen und die resultierenden Kombinationsmöglichkeiten sind somit alle gültigen Anfragen definiert.

2.3 Bestehende Lösungen

Da das in dieser Arbeit bearbeitete allgemeine Problem, die intelligente Dialogführung, bereits seit langem besteht, wurden bereits eine Vielzahl von Lösungen entwickelt. Vor der Entwicklung der in dieser Arbeit vorgestellten Lösung wurden daher bereits existierende Systeme betrachtet, die den Benutzer bei seinen Eingaben unterstützen sollen. In diesem Kapitel werden die Ergebnisse dieser Recherche zusammengefasst.

2.3.1 Definition „Intelligente Dialogführung"

Da der Begriff „Intelligente Dialogführung", insbesondere der Teil „intelligente", einen recht großen Raum für Interpretationen lässt, wird hier zunächst definiert, wie der Begriff „intelligente Dialogführung" in dieser Arbeit zu verstehen ist.

Als **Dialogführung** wird das Anleiten und Begleiten des Benutzers durch eine Reihe von Eingabemöglichkeiten bezeichnet. Diese Eingabemöglichkeiten können dabei unterschiedlicher Art und auf mehrere Seiten oder Formulare verteilt sein.

Als **intelligent** wird diese Dialogführung dann bezeichnet, wenn das System, das den Benutzer führt, auf diesen und dessen Eingaben individuell reagieren kann. Individuell bedeutet hierbei, dass die jeweiligen Eingaben im Kontext zum Ziel des Benutzers verstanden werden und der Benutzer zielgerichtet unterstützt wird.

Meist wird diese Definition weniger allgemein und auf die verwendete Technik bezogen (beispielsweise Agentensysteme) verfasst, wie zum Beispiel bei den von Delisle und Moulin in [DM02] beschriebenen verschiedenen intelligenten Systemen. Obige Definition entspricht dabei am ehesten den ebenfalls in [DM02] beschriebenen „intelligent support systems", die den Benutzer mit kontextabhängigen Informationen unterstützen. Im Gegensatz zu den dort beschriebenen „Intelligent Assistants" ist bei der obigen Definition eine Anpassung an den Benutzer mit Hilfe von maschinellen Lernverfahren nicht unbedingt erforderlich.

2.3.2 Kategorisierung der Systeme

Aufgrund der schier endlosen Zahl verschiedener Systeme, die zur Benutzerführung existieren, ist es sinnvoll diese zunächst grob zu Gliedern. Hierfür wurden die in [Dry97] beschriebenen Kategorien Wizard und Guide übernommen. Zusätzlich wurde die Kategorie der reaktiven Systeme eingeführt, in der Agentensysteme und mixed-initiative-Systeme zusammengefasst werden.

Guide Guide-Systeme unterstützen den Benutzer bei seinen Eingaben, indem sie für den Benutzer hilfreiche Informationen anzeigen. Solche Informationen könnten zum Beispiel die für ein Eingabefeld erforderliche Datumsformatierung sein oder die Information, wie der Benutzer anschließend zum nächsten Schritt des Formulars gelangt. Guide-Systeme sind dabei auf das Anzeigen von Informationen beschränkt, alle Eingaben werden durch den Benutzer getätigt. Ein Guide kann nach der obigen Definition intelligent sein, sofern die angezeigten Informationen anhand der Eingaben des Benutzers angezeigt oder angepasst werden können. Ein Beispiel für solch ein Guide-System ist das in [Sel94] beschriebene COACH System, das den Benutzer durch Hinweise unterstützt, selbst jedoch keine Eingaben tätigt oder ergänzt. Ein aktuelleres Beispiel sind die häufig bei der Registrierung auf Webseiten, wie zum Beispiel Google[4], anzutreffenden Eingabefelder für das Passwort, die den Benutzer während der Eingabe über die „Stärke" seines Passworts informieren.

Der Vorteil dieser Systeme ist, dass sie dem Benutzer bei einem Eingabefeld kurz und prägnant die erwartete Eingabe beschreiben können oder andere relevante Hinweise geben können. Sobald jedoch Eingabefelder eine komplexe Eingabe erfordern oder Beziehungen zwischen verschiedenen Eingabefeldern bestehen, werden die zum Verständnis nötigen Erklärungen mitunter sehr lang und/oder kompliziert.

Wizard Wizard-Systeme unterstützen den Benutzer bei einer Aufgabe, indem sie die für diese Aufgabe erforderlichen Eingaben nacheinander, in mehreren separaten Schritten anzeigen. Bei jedem Schritt können dem Benutzer weitere Informationen zu den für diesen Schritt erforderlichen Eingaben angezeigt werden. Ein Wizard folgt dabei stets einer im Voraus festgelegten Abfolge von Schritten. Es ist möglich, dass die festgelegte Abfolge Verzweigungen enthält, bei denen Aufgrund der Eingabe des Benutzers entschieden wird, welcher Pfad weiter verfolgt wird. Ein Beispiel für ein solches System ist [SV04], bei dem ein Wizard genutzt wird, um eine Reihe von Schritten zu durchlaufen, die durch einen Entscheidungsbaum beschrieben werden. Anhand der Eingaben wird in diesem Fall anschließend eine Benutzungsoberfläche generiert. Am besten bekannt sind Wizard-Systeme jedoch wohl durch Installationswizards wie den Windows Installer[5], die den Benutzer durch die Installation einer Anwendung auf seinem Computer führen.

Der Vorteil dieser Systeme ist, dass der Benutzer jederzeit genau weiß welche Eingabe gerade von ihm verlangt wird und wie er anschließend fortfahren kann. Der Nachteil hierbei ist jedoch, dass der Benutzer in eine strikte Abfolge der Eingaben gezwungen wird. Mit steigender Komplexität der Aufgabe wird es dabei zunehmend schwieriger, denn Zusammenhang zwischen den einzelnen Eingaben verfolgen zu können.

reaktive Systeme Als reaktive Systeme werden in dieser Arbeit all jene Systeme bezeichnet, die in der Lage sind, selbst *aktiv* in den vom Benutzer bearbeiteten Dialog einzugreifen, um zum Beispiel Eingaben zu tätigen. Da sie aus Gründen der Benutzbarkeit jedoch im

[4] https://accounts.google.com/SignUp
[5] http://msdn.microsoft.com/de-de/library/cc185688.aspx

allgemeinen nur als Reaktion auf eine Eingabe des Benutzers aktiv werden, werden sie hier als „reaktive Systeme" bezeichnet.
Reaktive Systeme werden meist mit Hilfe sogenannter „Agenten" realisiert. Da eine allgemeingültige Definition dieses Begriffs schwierig ist, wird in dieser Arbeit die von Wooldridge [Wooo2] vorgeschlagene und für diese Arbeit ausreichende Definition verwendet:

> An *agent* is a computer system that is *situated* in *some environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives.

Die Spannweite der Komplexität von Agenten in den untersuchten Lösungen ist dabei sehr groß. Sie reicht von einfachsten Agenten, die vom Benutzer getätigte Eingaben in einem Feld speichern, um bei späteren Eingaben eine auto-complete Funktion anzubieten, bis hin zu Agenten, die das Verhalten des Benutzers beobachten und Methoden des maschinellen Lernens verwenden, um den Dialog besser an den Benutzer anpassen zu können. Beispiele für Agentensysteme sind unter anderem [Yano9], bei dem auf einer Webseite mit Expertenantworten zu verschiedenen Fragen ein Agent verwendet wird, um die Ergebnisse der Suche an den Benutzer anzupassen, sowie [LCW09], bei dem mehrere Agenten verwendet werden, um eine personalisierbare Routenplanung für Touristen zu realisieren, die auf die Wünsche nach bestimmten Sehenswürdigkeiten oder Restaurants eingeht.
Ebenfalls in die Kategorie der reaktiven Systeme fallen die sogenannten „mixed-initiative" Systeme, bei denen die Initiative der Eingabe zwischen Computer und Benutzer wechseln kann [AGA06]. Zwar können diese Systeme mit Agenten realisiert werden, meist sind sie jedoch wesentlich strikter aufgebaut, um eine annehmbare Benutzbarkeit zu gewährleisten. Ein Beispiel für ein mixed-initiative Systeme ist [FMO$^+$01], das ein System zur Reiseplanung beschriebt, bei dem nach einer Eingabe des Benutzers die Initiative zum Computer wechseln kann, der seinerseits Eingaben tätigt und anschließend wieder auf Eingaben des Benutzers wartet.

Der Vorteil dieser Systeme ist, dass in einem großen Maße auf den Benutzer eingegangen werden kann und eine Anpassung des Systems an den Benutzer möglich ist. Hierbei besteht jedoch die Gefahr, dass die Benutzbarkeit des Systems sinkt, wenn der Benutzer die Anpassungen nicht mehr nachvollziehen kann [DINo8].

Eine strikte Trennung in die oben genannten Kategorien ist offensichtlich nicht immer möglich. Eine große Zahl von Systemen kombiniert Ansätze aus den verschiedenen Kategorien miteinander. Dennoch sollten diese drei Kategorien einen Überblick über die unterschiedlichen Funktionsweisen und Möglichkeiten der bestehenden Lösungen erlauben.

2.3.3 Maschinelle Lernverfahren

Während der Untersuchung der bereits bestehenden Lösungen zeigte sich, dass viele dieser Lösungen maschinelle Lernverfahren einsetzen, um die Benutzungsoberfläche besser an den Benutzer anpassen und ihn unterstützen zu können. Ein Beispiel hierfür ist zum Beispiel [HCo7], das neuronale Netze verwendet, um aus den Eingaben des Benutzers Empfehlungen für weitere Eingaben zu generieren. Auch bei der in dieser Arbeit vorgestellten Lösung wäre

der Einsatz maschineller Lernverfahren möglich gewesen. Da maschinelle Lernverfahren jedoch sehr komplex sind und die für diese Arbeit verfügbare Zeit begrenzt ist, werden Methoden des maschinellen Lernens in dieser Arbeit explizit nicht berücksichtigt.

3 Lösung

In diesem Kapitel wird die im Rahmen dieser Diplomarbeit entwickelte Lösung in allgemeiner Weise vorgestellt. Im anschließenden Kapitel 4 wird auf die Details der Implementierung der Lösung eingegangen.

3.1 Problemstellung

Im Rahmen dieser Diplomarbeit soll eine Lösung zur intelligenten Dialogführung für das PESCaDO System entwickelt werden. Die Lösung soll dabei die folgenden Anforderungen erfüllen:

1. Die Lösung soll sicherstellen, dass alle erforderlichen Eingaben vom Benutzer getätigt werden. Dabei ist zu beachten, dass sich die Gesamtheit der erforderlichen Eingaben in Abhängigkeit von den bisher getätigten Eingaben verändern kann.
2. Die Lösung soll dem Benutzer, basierend auf seinen bisherigen Eingaben und/oder gespeicherten Profildaten, Vorschläge für weitere Eingaben machen. Die vorgeschlagenen Eingaben sollen im Bezug auf das vom Benutzer verfolgte Ziel sinnvoll sein.
3. Die Lösung soll dem Benutzer Vorschläge für Eingabewerte machen. Diese Vorschläge sollen anhand der bisherigen Eingaben, gespeicherten Profildaten oder aus anderen Quellen verfügbaren Daten erzeugt werden und im aktuellen Kontext sinnvoll sein.
4. Die Lösung soll den Benutzer auf fehlerhafte Eingaben hinweisen und nach Möglichkeit bei der Korrektur unterstützen.
5. Die Lösung soll dem Benutzer die Möglichkeit geben, eine Anfrage weiter bearbeiten zu können, nachdem sie an den Server gesendet und das Ergebnis angezeigt wurde.

3.2 Allgemeine Beschreibung der Lösung

Die in dieser Arbeit vorgestellte Lösung für die in Kapitel 3.1 beschriebenen Probleme besteht aus zwei Teilen. Der erste Teil der Lösung befasst sich mit den technischen Aspekten, zum Beispiel wie erforderliche Felder oder fehlerhafte Eingaben erkannt werden können. Der zweite Teil der Lösung befasst sich mit dem Benutzer und wie mit diesem kommuniziert und interagiert werden kann.

Um sicherzustellen, dass alle erforderlichen Eingabefelder ausgefüllt werden und keine fehlerhaften Eingaben vorhanden sind, muss der Benutzer auf solche Fälle hingewiesen werden können. Daher ist es zunächst erforderlich, die Abhängigkeiten zwischen den einzelnen Feldern – und somit ihren Einfluss auf die Anfrage – zu erkennen. Es bietet sich hierbei an, die bereits existierende PESCaDO Ontologie zu verwenden. In dieser Ontologie, genauer der PDL, sind bereits alle gültigen Anfragemöglichkeiten definiert. Aus diesen Informationen können leicht die Abhängigkeiten zwischen den einzelnen Feldern extrahiert und verwendet werden, um die Eingaben des Benutzers zu überprüfen und entsprechend zu reagieren.

Die in der Ontologie definierten Klassen entsprechen dabei den Eingabefeldern, die Instanzen einzelner Klassen den möglichen Eingabewerten für das Eingabefeld der jeweils instantiierten Klasse. Es ist dabei zu beachten, dass auch Klassen als Eingabewerte eines Feldes dienen können, wie zum Beispiel beim Anfragetyp, bei dem die Kindklassen als Eingabewerte für das Feld der Elternklasse dienen. Innerhalb einer Anfrage wird jede Klasse nur einmal verwendet, daher können die Namen der Eingabefelder mit den Namen der Klassen gleichgesetzt werden, ohne dass die Gefahr mehrerer gleichnamiger Eingabefelder besteht.

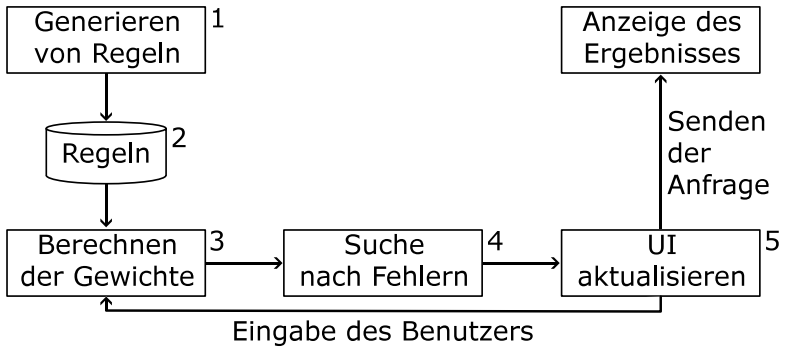


Abbildung 3.1: Funktionsweise der Lösung

Erste Prototypen haben jedoch gezeigt, dass es relativ viel Zeit in Anspruch nimmt, die komplette PESCaDO Ontologie zu laden und auszuwerten. Zudem ist nur ein relativ kleiner Teil der Ontologie für das Erkennen erforderlicher oder fehlerhafter Eingaben notwendig. Daher wird in einem ersten Schritt die Ontologie analysiert und eine Reihe von Regeln erstellt (**1** in Abbildung 3.1), welche die Abhängigkeiten der einzelnen Klassen der Ontologie (und damit die der zugehörigen Eingabefelder und Eingabewerte) beschreiben. Sofern die Ontologie nicht verändert wird, muss dieser Schritt nur ein Mal durchgeführt werden. Statt der gesamten Ontologie werden nun nur noch die kompakteren Regeln (**2**) verwendet, was wesentlich effizienter ist.

Um mit Hilfe dieser Regeln die einzelnen Eingaben bewerten zu können, wird zunächst jeder darin enthaltenen Klasse ein Gewicht zugeordnet. Zusätzlich erhält jede Regel einen Faktor, mit dem das Gewicht einer Klasse verändert werden kann. So kann das Gewicht

erforderlicher Klassen erhöht, das von zu Fehlern führenden Klassen gesenkt werden. Im Nachfolgenden werden Eingaben, die im aktuellen Kontext zu Fehlern bei der Verarbeitung der Anfrage führen, auch als verbotene Eingaben bezeichnet. Da jede dieser Klassen genau einem Eingabefeld oder -wert entspricht, können die Gewichte der Klassen direkt für die Bewertung der Eingabefelder und -werte verwendet werden. Hierbei ist jedoch zu beachten, dass nur Eingabewerte, die eine Klasse aus der Ontologie repräsentieren, ein Gewicht besitzen. Eingabewerte, die Instanzen einer Klasse darstellen, besitzen kein eigenes Gewicht. Wird im Nachfolgenden von einem Eingabewert mit Gewicht gesprochen, ist damit stets ein eine Klasse der Ontologie repräsentierender Eingabewert gemeint.

Nach jeder Eingabe, die der Benutzer tätigt, werden dann mit Hilfe der Regeln die aktuellen Gewichte der Eingabefelder und der Eingabewerte berechnet (**3**). Anschließend wird geprüft, ob zum Beispiel ein aufgrund des Gewichts verbotenes Feld ausgefüllt, beziehungsweise ein erforderliches Feld nicht ausgefüllt wurde (**4**), um eine entsprechende Meldung auszugeben (**5**).

Die Gewichte der Eingabefelder und -werte werden zudem verwendet, um dem Benutzer, wie unter Punkt 2 in Kapitel 3.1 beschrieben, Eingaben vorzuschlagen. Dazu wird die Aufmerksamkeit des Benutzers auf die wichtigen Eingabefelder und -werte mit einem hohen Gewicht gelenkt, während die Felder und Werte mit einem niedrigen Gewicht in den Hintergrund gestellt werden.

Um Eingabewerte vorzuschlagen (Punkt 3 in Kapitel 3.1), wird geprüft, ob die zum Beispiel im Profil hinterlegten Werte selbst (falls gewichtet) oder die Felder, in denen sie ausgewählt wurden, aufgrund ihres Gewichts in der aktuellen Anfrage erlaubt, also nicht verboten, sind. Ist dies der Fall, wird das entsprechende Feld der Anfrage mit dem entsprechenden Wert belegt. Stehen mehrere gewichtete Werte für ein Feld zur Auswahl, wird der Wert mit dem höchsten Gewicht gewählt.

Im Nachfolgenden werden nun die einzelnen Komponenten dieser Lösung ausführlich beschrieben. In Kapitel 3.5 wird anschließend auf verschiedene Möglichkeiten zur Gestaltung der Benutzungsoberfläche eingegangen, mit denen die im ersten Teil der Lösung erlangten Informationen dargestellt werden können.

3.3 Generierung der Regeln

Um die die Abhängigkeiten zwischen den Eingabefeldern und -werten beschreibenden Regeln zu generieren, muss zunächst betrachtet werden, was für eine Beziehung zwischen je zwei Klassen innerhalb der Ontologie besteht. Anhand dieser Beziehung kann erkannt werden, ob eine Klasse eine andere bedingt oder verbietet. Aus diesen Informationen wird dann eine Regel mit einem entsprechenden Faktor erzeugt.
Wiederholt man den Schritt der Regelgenerierung für alle in der PDL definierten Klassen, so erhält man ein Regelwerk, das, im Bezug auf das Erfordern oder Verbieten von Klassen, äquivalent zur Ontologie ist.

3.3.1 Struktur einer Regel

Die verwendeten Regeln sind alle gleich aufgebaut und besitzen die folgenden Eigenschaften:

Rule ist ein logischer Ausdruck bestehend aus den logischen Operatoren AND, OR und NOT (dargestellt durch &&, || und !), Klammern ( ), den Namen von Klassen der PESCaDO Ontologie sowie dem Ausdruck *X.Value=Y*, wobei *X* der Name einer ein Eingabefeld repräsentierenden Klasse und *Y* der ausgewählte Wert, also eine Kindklasse oder Instanz von *X*, ist. Die Klassennamen können hier verwendet werden, da sie in der Ontologie einzigartig sein müssen.

Faktor beschreibt, wie das Gewicht einer Klasse durch diese Regel verändert wird. Mögliche Werte werden weiter unten beschrieben.

Affected ist der Name der von dieser Regel betroffenen Klasse. Sofern **Rule** als wahr ausgewertet wird, wird die Regel auf diese betroffene Klasse angewandt.

Ein Beispiel für eine Regel kann in Kapitel 4.3 betrachtet werden.

3.3.2 Gewichtungsfaktoren

Der jeder Regel zugeordnete Faktor erlaubt es später, die Gewichte Klassen aus der Ontologie (und damit die der zugehörigen Eingabefelder und -werte) zu manipulieren. Es wurde hierbei ein Faktor gewählt, da es vorkommen kann, dass mehrere Regeln auf das Gewicht der selben Klasse angewendet werden. Würde dann statt eines Faktors zum Beispiel ein Summand verwendet, um das Gewicht zu verändern, könnte es vorkommen, dass eine eigentlich verbotene Klasse (welche ein sehr kleines Gewicht haben sollte), aufgrund von vielen darauf angewendeten Regeln ein zu großes Gewicht hat. Durch die Wahl des Faktors 0 für verbotene Klassen wird auf einfache Weise sichergestellt, dass eine verbotene Klasse stets das Gewicht 0 hat. Da keine negativen Faktoren verwendet werden ist dies zudem stets das kleinste mögliche Gewicht.

Es werden die folgenden Faktoren verwendet:

- *fRequired* = 1024
- *fRecommended* = 2
- *fNeutral* = 1
- *fForbidden* = 0

Regeln, die eine Klasse erfordern, verwenden den Faktor *fRequired*, Regeln, die eine Klasse verbieten, verwenden *fForbidden*. Dadurch wird das Gewicht verbotener Klassen auf 0 reduziert und kann auch durch weitere Regeln nicht mehr verändert werden. Jedoch kann es nach wie vor geschehen, dass durch die wiederholte Anwendung von *fRecommended* eine nicht erforderliche Klasse ein höheres Gewicht erreicht als eine erforderliche Klasse. Um dies zu verhindern, muss abgeschätzt werden, wie viele Regeln maximal auf ein Gewicht wirken

können. Wurde bereits ein vollständiges Regelwerk generiert, kann auch die exakte Anzahl A ermittelt werden. Der Faktor *fRequired* muss dann so gewählt werden, dass er größer als $fRecommended^A$ ist.

Der Faktor *fNeutral* wird von keiner Regel verwendet und falls doch, kann diese Regel entfernt werden, da sie keinen Einfluss auf das Gewicht hat. Er ist hier lediglich der Vollständigkeit halber aufgeführt.

Der Faktor *fRecommended* wird von Regeln verwendet, die eine Klasse (und damit eine Eingabe) empfehlen. Eine empfohlene Eingabe ist nicht erforderlich um die Anfrage zu vervollständigen, ist jedoch im Kontext der Anfrage sinnvoll.

In seltenen Fällen kann es vorkommen, dass die eigentlich widersprüchlichen Faktoren *fRequired* und *fForbidden* auf das selbe Gewicht angewandt werden. Da in diesem Fall der Effekt von *fForbidden* überwiegt, müssen diese Fälle genau betrachtet werden, um Fehler ausschließen zu können. Ein Beispiel, in dem dieser Fall auftritt, wäre eine Kindklasse von **Activity** (siehe Kapitel 2.2.1), die wegen ihres Status als Kindklasse erforderlich ist, aufgrund des ausgewählten Anfragetyps jedoch verboten wird.

3.3.3 Ableitung von Regeln aus Beziehungstypen

Um aus der komplexen Ontologie Regeln zu generieren, werden wie bereits erwähnt die Beziehungen zwischen je zwei Klassen betrachtet. Abhängig von der Art der Beziehung wird dann eine Regel erstellt, wobei der selbe Beziehungstyp stets die selbe Art von Regeln erzeugt. Auf diese Weise können für die relevanten Beziehungen leicht Regeln erstellt werden, während irrelevante Beziehungstypen direkt übersprungen werden können.

Für das Generieren der Regeln werden die folgenden Beziehungstyp-Regel-Schemata verwendet (Großbuchstaben stehen dabei für den Namen einer Klasse, während der Beziehungstyp mit einem Kleinbuchstaben beginnt):

Y subClassOf X

Y ist hierbei Kindklasse von X.
Da für eine gültige Anfrage immer ein Blatt im Baum der Subklassen einer Klasse ausgewählt werden muss, erzeugt dieser Beziehungstyp die folgende Regel:
Rule: X AND NOT (Z1 OR Z2 OR ...)
Affected: Y
Factor: *fRequired*
Z1, *Z2* etc. sind dabei die anderen Kindklassen von X. Dies ist notwendig, da ansonsten auch nach Auswahl einer Kindklasse von X die Regeln aller anderen Kindklassen von X als wahr ausgewertet würden. Es kann jedoch immer nur eine Kindklasse ausgewählt werden.

X hasSomeValuesFrom(Z1 or Z2 or ...)

Die Klasse X erfordert eine der Klassen *Z1*, *Z2*, etc.
Aus diesem Beziehungstyp werden mehrere Regeln generiert. Zuerst wird die Elternklasse Y ermittelt, von der alle in *hasSomeValuesFrom* enthaltenen Klassen abgeleitet

sein müssen:
Rule: X
Affected: Y
Factor: *fRequired*
Dies ist notwendig, da ansonsten später zwar die die Kindklassen darstellenden Eingabewerte, nicht jedoch das die Elternklasse darstellende Eingabefeld als erforderlich angesehen werden.

Da nur eine der Klassen *Zx* ausgewählt werden kann, wird anschließend für jede Klasse *Zx* die folgende Regel erstellt:
Rule: Y AND NOT (Z1 OR ... OR Zn)
Affected: Zx
Factor: *fRequired*
(Z1 OR ... OR Zn) enthält dabei alle Klassen, die in *hasSomeValuesFrom(Z1 or Z2 or ...)* enthalten sind, mit Ausnahme von *Zx*.

not (X hasSomeValuesFrom(Z1 or Z2 or ...))

Die Klasse *X* verbiete jede der Klassen *Z1*, *Z2* etc.
Aus diesem Beziehungstyp wird für jede Klasse *Zx* die folgende Regel generiert:
Rule: X
Affected: Zx
Factor: *fForbidden*

X hasOnlyValuesFrom(Y)

Die Klasse *X* erfordert eine beliebige Subklasse von *Y*. Es wird die folgende Regel generiert:
Rule: X
Affected: Y
Factor: *fRequired*
Da die Kindklassen von *Y* bereits durch die **subClassOf**-Regel in Abhängigkeit von *Y* als erforderlich markiert werden, müssen hier keine weiteren Regeln dafür generiert werden.

X hasExactCardinality(1, Y)

Die Klasse *X* erfordert genau ein Mal die Klasse *Y*. Da aufgrund der einfachen Struktur der Regeln keine Kardinalitäten unterstützt werden und aufgrund des Aufbaus der Benutzungsoberfläche (siehe Kapitel 3.5.2) jedes Eingabefeld (und damit jede Klasse) nur ein Mal vorkommt, genügt es in diesem Fall die folgende Regel zu generieren:
Rule: X
Affected: Y
Factor: *fRequired*

X hasMinCardinality(2, Y)

Die Klasse *X* erfordert mindestens zwei Mal die Klasse *Y*. Da die Eigenschaft einer minimalen Anzahl mit den einfachen Bausteinen der Regeln nicht dargestellt werden kann, wird hierfür ein Präfix verwendet, der vor die mehrfach erforderliche Klasse gestellt wird. Dadurch kann bei der späteren Auswertung der Regel erkannt werden,

dass die Klasse *Y* hier mehrfach (also mindestens zwei Mal) vorhanden sein muss.
Es wird die folgende Regel generiert:
Rule: X
Affected: balance-multi:Y
Factor: *fRequired*
Da OWL auf XML basiert und Namespaces verwendet ist darauf zu achten, dass der Präfix mit keinem Namespace-Präfix kollidiert, welcher eventuell in den Regeln verwendet wird.

Einen Sonderfall bilden der Start- und Endzeitpunkt, die in der Klasse *Request* gefordert werden. In diesen beiden Fällen werden keine Klassen gefordert, die über ihre einzigartigen Namen identifiziert werden können, sondern beide Male ein Datumstyp. Für diese beiden Fälle werden anhand der Namen der Relationen zwei neue, in der Ontologie nicht enthaltene Klassen hinzugefügt:

X hasFromDateTime
Die Klasse *X* erfordert einen Startzeitpunkt.
Es wird die folgende Regel erzeugt:
Rule: X
Affected: HasFromDateTimeClass
Factor: *fRequired*

X hasToDateTime
Die Klasse *X* erfordert einen Endzeitpunkt.
Es wird die folgende Regel erzeugt:
Rule: X
Affected: HasToDateTimeClass
Factor: *fRequired*

Während die Ontologie noch weitere Beziehungstypen beschreibt, reichen die oben aufgeführten Beziehungstypen für die Generierung der Regeln bereits aus.

3.4 Gewichten und Auswerten der Eingaben

Wurden die Regeln wie im vorhergehenden Abschnitt beschrieben erstellt, können diese nun verwendet werden, um die Eingaben zu gewichten. Dazu müssen jedoch zuerst der aktuelle Zustand der Anfrage, das heißt welche Felder mit welchen Werten ausgefüllt wurden, ermittelt werden. In dieser Lösung wird dazu für jede Eingabe, die der Benutzer tätigt, der Name der zu dem jeweiligen Eingabefeld gehörenden Klasse aus der PESCaDO Ontologie in einer Liste von *Key/Value* Paaren hinterlegt. Der Name der Klasse ist dabei der Schlüssel *Key*, während *Value* den Namen des für das Feld ausgewählten Wertes beinhaltet. Hierbei wird unterschieden, ob der ausgewählte Wert eine Klasse ist, die selbst Kindklassen besitzt und daher noch weitere Eingaben erforderlich sind, oder ob es sich um eine Klasse auf Blattebene der Klassenhierarchie, beziehungsweise eine Instanz einer Klasse handelt. Im ersten Fall wird ein zusätzliches *Key/Value* Paar in die Liste eingefügt, bei dem *Key* dem Namen der

Klasse mit weiteren Kindklassen entspricht und dessen *Value* leer ist. Beim Einfügen von *Key/Value* Paaren werden Duplikate vermieden. Eventuell bereits bestehende *Keys* werden mit den neuen Werten überschrieben.

Eine Ausnahme hierbei bilden *Keys* mit dem Präfix *balance-multi:*, da diese mehrere Werte besitzen können und daher beim Einfügen nicht überschrieben werden dürfen. Der einzige Fall in dem dies momentan Auftritt ist die Auswahl mehrerer Gebiete für eine Anfrage. Werden mehrere Gebiete ausgewählt, so wird das erste noch ohne Präfix gespeichert. Ab dem zweiten Gebiet wird der Präfix *balance-multi:* im *Key* verwendet, so dass die entsprechende Regel das zweimalige Vorhandensein erkennen kann. Die Werte der nachfolgenden Gebiete werden bei dem *Value* des *Keys* mit Präfix angehängt und können folglich nur gemeinsam gelöscht oder überschrieben werden.

3.4.1 Berechnung des Gewichts einer Eingabe

Nach jeder Eingabe des Benutzers werden die Gewichte aller Eingabefelder und -werte neu berechnet. Die Eingabewerte werden im Nachfolgenden nicht länger explizit genannt, da das Vorgehen bei Eingabefeldern und -werten äquivalent ist.

Bei der Berechnung der Gewichte werden zunächst die Gewichte aller Eingabefelder auf *fNeutral* (hat den Wert 1) zurückgesetzt. Dann werden für jedes Eingabefeld E die Regeln R gesucht, die E betreffen, das heißt r_n.*Affected* entspricht der dem Eingabefeld zugeordneten Klasse. Für alle gefundenen Regeln R wird dann geprüft, ob r_n.*Rule* als wahr ausgewertet wird (im Nachfolgenden wird solch eine Regel auch als aktive Regel bezeichnet). Hierzu werden die Klassennamen in r_n.*Rule* durch *true* ersetzt, falls der Klassenname als *Key* in der Liste der bisherigen Eingaben enthalten ist. Andernfalls wird der Klassenname durch *false* ersetzt. Wird in r_n.*Rule* der Ausdruck *X.Value=Y* verwendet, so wird dieser Ausdruck durch *true* ersetzt, falls X in der Liste der bisherigen Eingaben als *Key* verwendet wird und der zugehörige Wert von *Value* gleich Y ist. Andernfalls wird auch dieser Ausdruck durch *false* ersetzt.

Der so erzeugte logische Ausdruck enthält keine Klassennamen mehr und kann ausgewertet werden. Sofern er als wahr ausgewertet wird, wird der Faktor der Regel r_n.*Factor* auf das Gewicht des Eingabefeldes E angewendet. Anschließend wird die nächste Regel aus R überprüft. Dies wird so lange wiederholt, bis alle Regeln für jedes Eingabefeld geprüft wurden.

Algorithmus 3.1 beschreibt dieses Vorgehen nochmals mit Pseudocode, die Auswertung einer Regel wurde dabei jedoch gekürzt dargestellt.

Die Vorgehensweise, bei der für jedes Eingabefeld alle Regeln durchsucht werden, wurde der Variante, jede Regel auszuwerten und bei allen aktiven Regeln die Gewichte der betroffenen Felder anzupassen, aus zwei Gründen vorgezogen. Zum einen ist das Auswerten von r_n.*Rule*, insbesondere wenn mehrere Klassennamen in r_n.*Rule* enthalten sind, relativ teuer (d.h. benötigt relativ viel Zeit) und sollte möglichst sparsam verwendet werden. Zum anderen

Algorithmus 3.1 Berechnung der Gewichte der Eingaben (Pseudocode)

```
for each e in Eingabefelder do
    e.Gewicht = 1;
    for each r in Regeln do
        if r.Affected == e.Ontologieklasse then
            if EVALUATE(r.Rule) == true then
                e.Gewicht = e.Gewicht * r.Factor;
            end if
        end if
    end for
end for
function EVALUATE(rule)
    return { true   rule wurde als wahr ausgewertet
           { false  sonst
end function
```

bietet der verwendete Algorithmus Raum für spätere Optimierungen, da mit ihm leicht auch nur ein Teil der Eingabefelder ausgewertet werden kann.

3.4.2 Auswerten der Eingaben

Nachdem die Gewichte aller Eingabefelder berechnet wurden, können nun die Eingaben des Benutzers ausgewertet werden. Hierzu wird für jedes Eingabefeld das dazugehörige Gewicht betrachtet.

Ist das Gewicht größer oder gleich *fRequired*, so ist das Eingabefeld erforderlich. Es wird nun geprüft, ob das Eingabefeld bereits in der Liste der getätigten Eingaben vorhanden ist und *Value* nicht leer ist, also ein Wert ausgewählt wurde. Ist dies nicht der Fall, wurde ein erforderliches Feld noch nicht ausgefüllt und eine entsprechende Fehlermeldung kann angezeigt werden.

Ist das Gewicht gleich *fForbidden*, so ist das Eingabefeld verboten. Es wird geprüft, ob das Eingabefeld bereits in der Liste der getätigten Eingaben vorhanden ist und *Value* nicht leer ist, also ein Wert ausgewählt wurde. Ist dies der Fall, wurde ein verbotenes Feld ausgefüllt (das bei der Verarbeitung der Anfrage zu einem Fehler führen würde) und eine entsprechende Fehlermeldung kann angezeigt werden.

Bei allen anderen Gewichten handelt es sich um optionale oder für die Anfrage nicht relevante Felder, für die keine Fehlermeldung ausgegeben wird. Nichts desto trotz kann darauf hingewiesen werden, dass durch das Ausfüllen von Feldern, deren Gewicht größer oder gleich *fRecommended* ist (sofern das Feld nicht erforderlich ist), das Ergebnis der Anfrage eventuell verbessert werden kann.

3.5 Möglichkeiten zur Gestaltung der Benutzungsoberfläche

Neben den in diesem Kapitel bisher beschriebenen Methoden zum Erkennen fehlerhafter oder unvollständiger Eingaben, spielt die Art und Weise, wie dieses Wissen dem Benutzer vermittelt und genutzt wird, um ihn beim Erstellen seiner Anfrage zu unterstützen, eine wichtige Rolle in dieser Arbeit. In diesem Abschnitt werden daher verschiedene Methoden zur Gestaltung der Benutzungsoberfläche beschrieben.

3.5.1 Evaluation der alten Benutzungsoberfläche

Eine erste Benutzerstudie [MHW], bei der die alte Benutzungsoberfläche untersucht wurde, offenbarte mehrere Probleme.

So erscheint die alte Benutzungsoberfläche zwar intuitiv und lädt zum Klicken auf die verschiedenen Buttons ein, jedoch verliert sich der Benutzer schnell in den einzelnen Dialogfenstern. Ebenfalls fehlte den Benutzern der Zusammenhang zwischen den Dialogen und eine Anleitung, wie eine Anfrage erstellt wird. In der vorhandenen Anleitung wird lediglich kurz erklärt, welchen Dialog die einzelnen Buttons am oberen Rand öffnen.

Auch die Dialoge mit ihren vielen Auswahlmöglichkeiten bereiteten Probleme, da den Benutzer nicht immer klar war, warum welche Auswahl getroffen werden muss.

Zu der Benutzerstudie sei angemerkt, dass zum einen die Benutzungsoberfläche sowie der serverseitige Dienst zum Generieren der Antwort in Teilen noch nicht funktionsfähig waren. Zum anderen basierte die damalige Benutzungsoberfläche auf einer älteren Version der Ontologie, in denen zum Beispiel die Aktivitäten noch weiter unterteilt waren. Nichts desto trotz können die meisten Ergebnisse dieser Benutzerstudie bei der Gestaltung der neuen Benutzungsoberfläche verwendet werden.

3.5.2 Aufbau der Benutzungsoberfläche

Auch wenn die Struktur der alten Benutzungsoberfläche mit den drei Bereichen relativ gut angenommen wurde, wird für diese Lösung eine neue Struktur eingeführt. Grund hierfür ist das Ziel, eine bessere Trennung zwischen der eigentlichen Anfrage und anderen Bereichen der Benutzungsoberfläche zu erreichen. Zu diesen Bereichen gehören eine Startseite, die die in der Benutzerstudie vermisste allgemeine Beschreibung von PESCaDO beinhaltet, einen Bereich, der das Ergebnis der Anfrage beinhaltet und einen Bereich, in dem der Benutzer persönliche Daten hinterlegen und so ein eigenes Profil erstellen kann. Die verschiedenen Bereiche der Benutzungsoberfläche können über die Hauptnavigation (**1** in Abbildung 3.2) jederzeit erreicht werden. Die Hauptnavigation bleibt dabei für jeden der Bereiche unverändert, während sich der unterhalb der Navigation angezeigte Inhalt des Bereichs ändern kann.

Die Karte wurde (voll funktionsfähig) von ihrem zentralen Platz in der alten Oberfläche in den Hintergrund verlegt. So ist sie noch immer stets präsent, jedoch liegt der Fokus nun auf

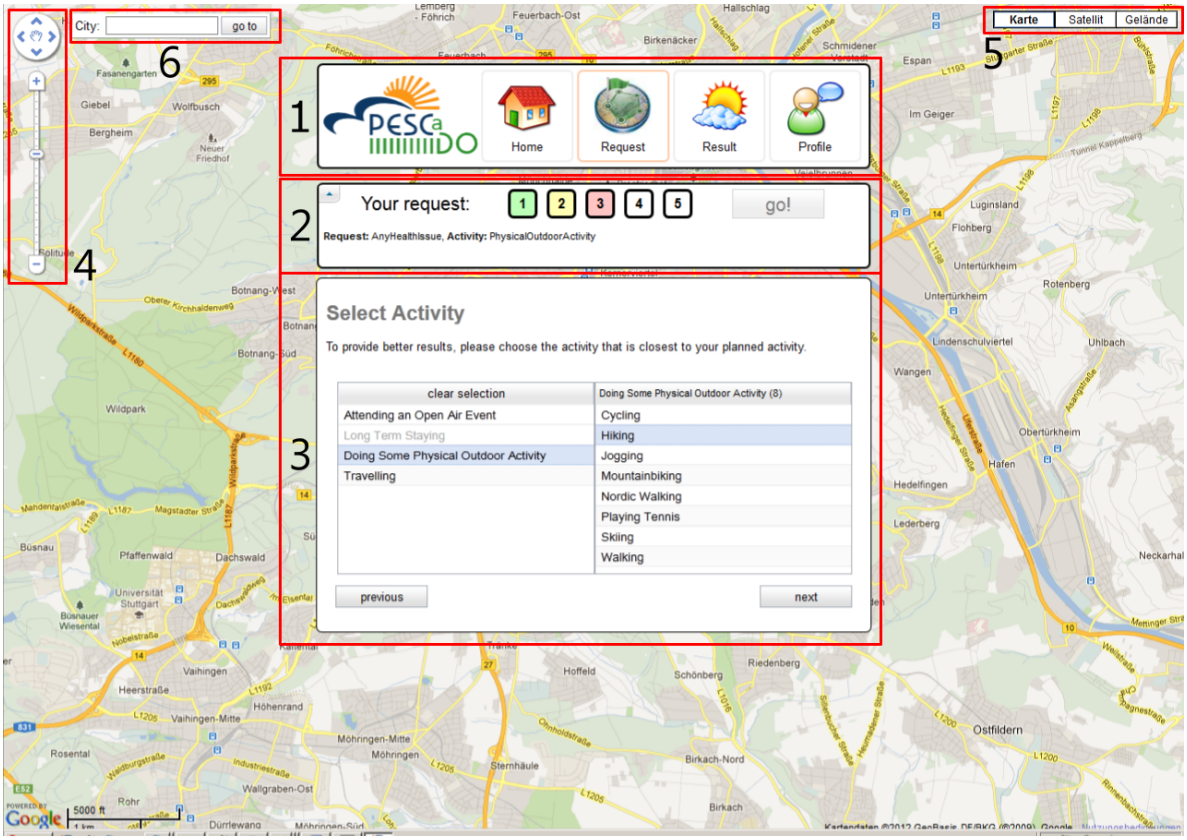


Abbildung 3.2: Neue Benutzungsoberfläche von PESCaDO

dem wichtigeren Dialog, der im Vordergrund eingeblendet ist. Zusätzlich wurden die nicht vorhandenen oder selbst implementierten Steuerelemente der Karte in der alten Version der Oberfläche durch die Standardsteuerelemente von Google Maps ersetzt (**4** und **5**). Diese sind vielen Benutzern bereits bekannt und vereinfachen so die Benutzung der Karte. Das Feld zur Suche eines Orts wurde ebenfalls auf die Karte ausgelagert (**6**) und ist dem Stil der anderen Steuerelemente der Karte nachempfunden, um so seinen Bezug zur Karte deutlich zu machen.

Da einer der größten Kritikpunkte der alten Benutzungsoberfläche die fehlende Orientierung beim Ausfüllen der Dialoge war, wird in dieser Lösung auf die in beliebiger Reihenfolge bearbeitbaren Dialoge verzichtet. Stattdessen wird der Benutzer, ähnlich wie bei einem Wizard (siehe Kapitel 2.3.2), durch eine Reihe von Dialogseiten geführt. Auf jeder Dialogseite füllt der Benutzer eine Reihe von Eingabefeldern aus und hat so am Ende des Dialogs seine Anfrage erstellt. Durch die sequentiell bearbeiteten Dialogseiten wird dabei sichergestellt, dass der Benutzer alle erforderlichen Eingabefelder ausfüllen kann und die Anfrage vollständig ist.

Diese Art der Dialogführung eignet sich zwar gut, um unerfahrene Benutzer anzuleiten. Erfahrenere Benutzer fühlen sich dabei jedoch schnell bevormundet (siehe [MHW]). Um dies

zu vermeiden wird die strikte Dialogstruktur durch zwei Anpassungen etwas aufgebrochen. Zum einen kann der Benutzer auf jeder Seite des Dialogs zur nächsten oder vorherigen Seite wechseln (sofern vorhanden), unabhängig davon, ob und welche Eingaben er auf der aktuellen Seite getätigt hat. Zum anderen kann der Benutzer jederzeit frei zu jeder anderen Dialogseite wechseln, unabhängig von seiner aktuellen Position im Dialog.

In einer ersten Version dieser Lösung war angedacht, die Dialogseiten dynamisch zu erstellen. Hierzu würden zunächst die Eingabefelder wie weiter oben beschrieben gewichtet und anhand ihres Gewichts in einer Liste absteigend sortiert. Die wichtigsten Felder stünden somit am Anfang dieser Liste, verbotene Felder an deren Ende. Auf jeder Dialogseite werden dann, beginnend am Anfang der Liste, eine bestimmte Zahl dieser Felder angezeigt. Dadurch werden dem Benutzer die wichtigsten Felder direkt am Anfang des Dialogs angezeigt. Sobald er alle erforderlichen Felder ausgefüllt hat, kann er entscheiden, ob er die Anfrage senden oder weitere Felder bearbeiten möchte.

Gegen diese Variante spricht jedoch das Problem, dass sich die Reihenfolge der Eingabefelder und der Aufbau der Dialogseiten mit jeder getätigten Eingabe ändern kann. So kann ein bisher erforderliches Feld aufgrund einer Eingabe zu einem verbotenen Feld werden und von seiner Position auf der ersten Dialogseite auf die letzte Dialogseite verschoben werden oder umgekehrt. Ein weiteres Problem ist der angestrebte gleichmäßige Aufbau der Dialogseiten, da man möglichst verhindern möchte, dass auf einer Dialogseite die drei komplexesten Eingaben auf den Benutzer warten und auf einer anderen lediglich drei simple ja-nein Eingaben. Auch ist es schwierig bei der dynamischen Erstellung der Dialogseiten auf einen Zusammenhang der Eingaben zu achten, so dass der Dialog dem Benutzer willkürlich zusammengewürfelt erscheint und es für den Benutzer schwieriger wird, eventuelle Abhängigkeiten zwischen den Eingabefeldern zu erkennen. Hinzu kommt das Problem, dass im Voraus nicht fest steht, nach wie vielen Dialogseiten die Anfrage vollständig ist.

Daher wird in der hier vorgestellten Lösung eine fixe Anzahl von Dialogseiten verwendet. Anhand des Aufbaus einer Anfrage und den erforderlichen Eingaben wurden fünf verschiedene Dialogseiten und die jeweils enthaltenen Eingabefelder festgelegt. Dank der fixen Anzahl von Dialogseiten kann zudem eine Übersicht der Dialogseiten eingeführt werden (**2** in Abbildung 3.2). In dieser Übersicht befindet sich für jede Dialogseite eine Schaltfläche, mit der der Benutzer zum einen direkt zu der jeweiligen Dialogseiten gelangen kann. Zum anderen kann er so stets erkennen, wo er sich befindet und wie viele Dialogseiten er noch vor sich hat.

Zwar könnten nun die Eingabefelder innerhalb der Seiten anhand ihres Gewichts angeordnet werden, jedoch würde dies noch immer zu einem unregelmäßigen Seitenaufbau führen. Stattdessen werden die Felder entsprechend ihres Gewichts hervorgehoben, wobei ihre Position auf der Dialogseite nicht verändert wird.

Aufgrund der nun fest einer Dialogseite zugeordneten Eingabefelder kann es vorkommen, dass auf einer Dialogseite nur verbotene Felder enthalten sind. In diesem Fall bestünde die Möglichkeit, die Seite zu überspringen und direkt mit der nächsten Seite des Dialogs fortzufahren. Dies wird in dieser Lösung jedoch nicht verwendet, da es den Benutzer

verwirren könnte. Der Benutzer kann nicht sehen, warum diese eine Seite übersprungen wurde und geht eventuell von einem Fehler der Anwendung aus. Stattdessen wird auf solch einer Seite ein Hinweis eingeblendet, der dem Benutzer erklärt, dass er auf dieser Seite kein Feld ausfüllen muss und direkt mit der nächsten Seite fortfahren kann.

Die fünf Dialogseiten werden unterhalb der Übersicht angezeigt (**3**) und entsprechen grob den Dialogen der alten Benutzungsoberfläche, wurden jedoch neu angeordnet:

Anfrageart Hier legt der Benutzer die Art seiner Anfrage fest.

Aktivität Hier wählt der Benutzer eine Aktivität aus.

Gebiet Hier wählt der Benutzer das für seine Anfrage relevante Gebiet aus.

Zeitraum Hier wählt der Benutzer den für seine Anfrage relevanten Zeitraum aus.

Persönliche Details Hier wählt der Benutzer unter anderem das für die Anfrage zu verwendende Benutzerprofil aus.

Diese Reihenfolge wurde gewählt, um die Eingabe mit dem größten Einfluss, also die meisten anderen Eingaben beeinflusst, als erste Eingabemöglichkeit für den Benutzer zu positionieren. Zudem sieht der Benutzer so direkt zu Beginn, welche Arten von Anfragen er formulieren kann, und muss nicht erst nach dem Ausfüllen aller anderen Felder feststellen, dass die Art von Anfrage, die er gerne gestellt hätte, gar nicht verfügbar ist. Da die Eingabe mit dem größten Einfluss am Anfang steht, sind die meisten nachfolgenden Eingaben bereits gewichtet und entsprechend hervorgehoben worden wenn der Benutzer sie erreicht. Das hat zum einen den Vorteil, dass sich der Benutzer leichter ein Model der internen Zusammenhänge bilden kann [DM02]. Zum anderen werden so die meisten Situationen vermieden, in denen durch eine Eingabe des Benutzers eine zuvor getätigte Eingabe ungültig wird.

Für die Auswahl eines Gebiets werden die drei Elemente (**1**, **2** und **3**) der Benutzungsoberfläche mit der Hauptnavigation, der Übersicht der Dialogseiten und der Dialogseite selbst ausgeblendet. Stattdessen wird ein kleines, verschiebbares Dialogfenster angezeigt. So steht dem Benutzer ausreichend Platz zur Verfügung, um ein Gebiet auf der Karte zu markieren. Sobald das kleinere Fenster geschlossen wird, werden die anderen Elemente wieder angezeigt.

Auf jeder Dialogseite wird zusätzlich ein kurzer Text eingebunden der dem Benutzer erklärt, welche Eingaben er auf dieser Seite machen kann und welchen Effekt sie auf die Anfrage und deren Ergebnis haben.

Beibehalten von der ersten Variante wird die ständig sichtbare Schaltfläche zum Senden der Anfrage an den Server. Die Schaltfläche wird jedoch so lange als nicht verfügbar dargestellt, bis eine vollständige und fehlerfreie Anfrage vorliegt. Sobald der Benutzer eine vollständige Anfrage an den Server gesendet hat, wechselt die Benutzungsoberfläche zur Ergebnisanzeige (zweites Element von rechts in der Hauptnavigation **1**). Hier wird das vom Server zurückgegebene Ergebnis für die Anfrage dargestellt. Da Teile des Ergebnisses auf der Karte dargestellt werden, können Elemente der Benutzungsoberfläche ausgeblendet werden, die sonst die Karte und das Ergebnis verdecken würden. Die vom Benutzer getätigten Eingaben bleiben dabei bestehen. Wechselt der Benutzer nun zurück in die Ansicht zur

Erstellung einer Anfrage hat er die Möglichkeit, entweder seine zuvor gesendete Anfrage weiter zu bearbeiten oder eine neue, leere Anfrage zu erstellen.

3.5.3 Gestaltung der Eingabefelder

Bei der Gestaltung der Eingabefelder ist das Ziel dieser Lösung, die Aufmerksamkeit des Benutzers möglichst gezielt auf die erforderlichen Eingabefelder zu lenken und ihn so bei seinen Eingaben zu unterstützen. Verbotene Eingabefelder sollen gleichzeitig in den Hintergrund gestellt werden. Ebenso soll der Benutzer auf falsch oder noch nicht ausgefüllte Felder hingewiesen werden.

Im Nachfolgenden werden die von dieser Lösung verwendeten Methoden zur Gestaltung und Hervorhebung von Eingabefeldern beschrieben. Hervorhebung schließt hierbei auch das Ausblenden oder Deaktivieren von Feldern mit ein.

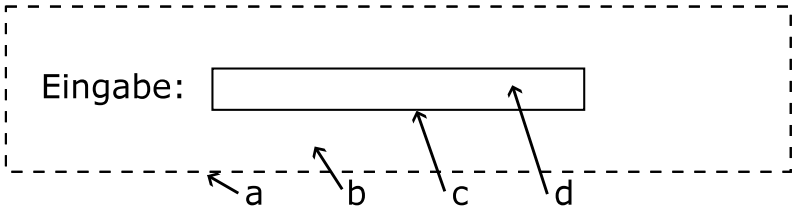


Abbildung 3.3: Schematische Darstellung eines Eingabefelds.
a) Rand der Umgebung des Eingabefelds
b) Umgebung des Eingabefelds
c) Rahmen des Eingabefelds
d) Hintergrund des Eingabefelds

Erforderliche Eingabefelder

Abbildung 3.4: Beispiel zur Darstellung eines erforderlichen Eingabefelds

Erforderliche Eingabefelder werden in dieser Lösung nicht gesondert hervorgehoben (siehe Abbildung 3.4), da in dieser Lösung bisher keine optionalen Felder gibt, die vom Benutzer ausgefüllt werden können und einen Einfluss auf das Ergebnis der Anfrage haben. Die meisten Felder wären daher als erforderlich markiert. Anstelle einer inflationären Hervorhebung erforderlicher Felder werden stattdessen die nicht erforderlichen Felder in den Hintergrund gestellt.

Verbotene Eingabefelder

Abbildung 3.5: Beispiel zur Darstellung eines verbotenen Eingabefelds

Verbotene Eingabefelder werden als deaktiviert angezeigt (siehe Abbildung 3.5). Beim Deaktivieren von Feldern werden diese weniger gut sichtbar dargestellt, so werden zum Beispiel schwarze Schriften oder Rahmen in Grau dargestellt. Bei Eingabefeldern kann die Farbe des Hintergrunds ebenfalls zu Grau wechseln. Zusätzlich kann die Eingabe für das Feld deaktiviert werden. Da solche „ausgegrauten" Felder von sehr vielen Anwendungen verwendet werden und deren Bedeutung im Umgang mit Computern allgemein bekannt ist, kann der Benutzer leicht erkennen, dass das deaktivierte Feld momentan nicht verfügbar ist.

Nicht ausgefüllte erforderliche Eingabefelder

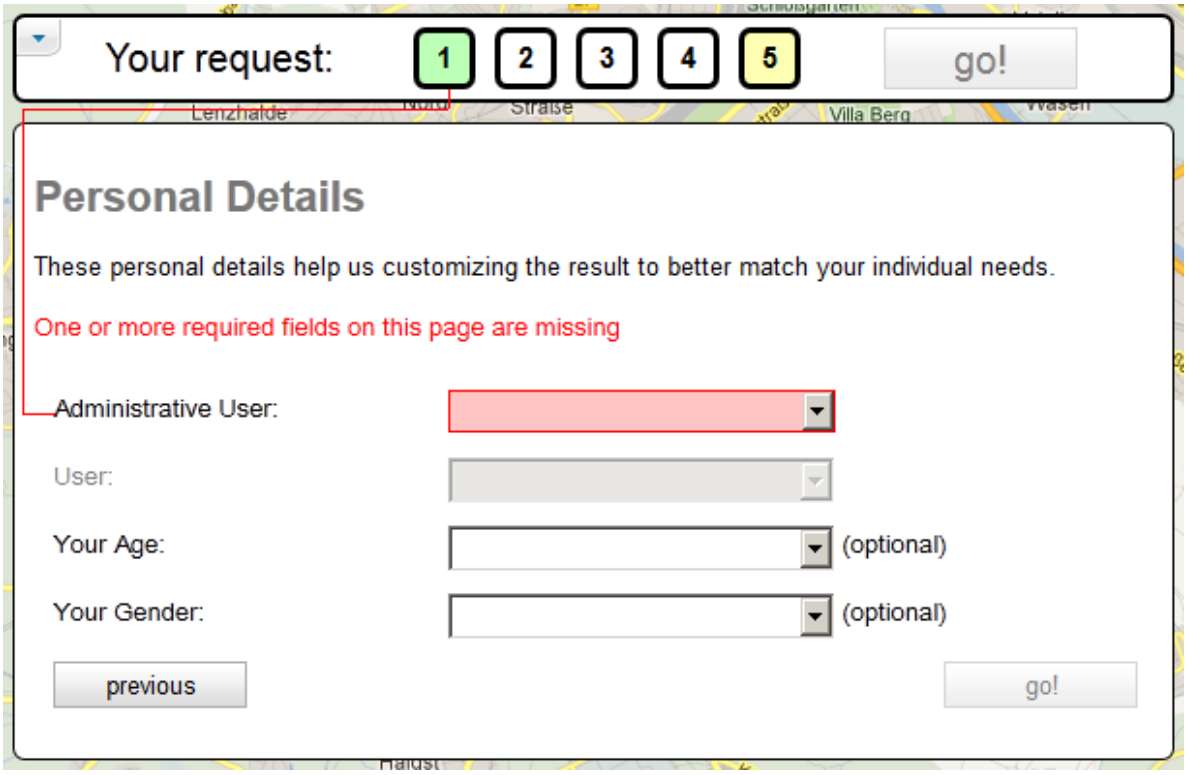


Abbildung 3.6: Beispiel zur Darstellung eines nicht ausgefüllten erforderlichen Eingabefelds

Erforderliche Felder, die der Benutzer noch nicht ausgefüllt hat, werden farbig hervorgehoben, sobald der Benutzer die Dialogseite zum zweiten Mal besucht (siehe Abbildung 3.6). Hierfür können verschiedene Teile des Eingabefelds gefärbt werden:

Hintergrund des Feldes Hierbei wird der standardmäßig weiße Hintergrund rot gefärbt

Rahmen des Feldes Hierbei wird das Feld rot umrandet

Umgebung des Feldes Hierbei wird der Bereich um das Feld, in dem sich zum Beispiel auch die zum Feld gehörender Beschriftung befindet, rot gefärbt

Rand der Umgebung des Feldes Hierbei wird der im vorherigen Punkt beschriebene Bereich um das Feld rot umrandet

Zusätzlich wird mit einer roten Linie angezeigt, weshalb dieses Feld erforderlich ist und ausgefüllt werden muss. Die rote Linie verbindet dabei das erforderliche Feld mit den Feldern, aufgrund denen das Feld erforderlich ist (die verursachenden Felder). Sollten sich diese Felder nicht auf der selben Dialogseite wie das erforderliche Feld befinden, so wird stattdessen eine rote Linie vom erforderlichen Feld zu den die jeweiligen Dialogseiten repräsentierenden Schaltflächen in der Übersicht gezeichnet. Auf den Dialogseiten mit den verursachenden Feldern wird dann ebenfalls eine rote Linie vom verursachenden Feld zu der Schaltfläche in der Übersicht gezeichnet, die die Dialogseite mit dem nicht ausgefüllten erforderlichen Feld darstellt.

Bei den Farben werden für flächige Hervorhebungen Pastelltöne verwendet, für kleinere Bereiche wie die Rahmen kräftige Volltöne. Sobald der Benutzer in einem farblich hervorgehobenen Feld eine Eingabe tätigt oder den bereits ausgewählten Wert ändert, werden die Farben dieses Felds auf die Standardfarben zurückgesetzt.

Neben der Hervorhebung des Feldes wird auf der Dialogseite zusätzlich ein entsprechender Hinweis angezeigt, der das aufgetretene Problem beschreibt.

Durch die Hervorhebung des Feldes in einer Farbe, die zum einen allgemein als Warnfarbe verwendet wird und zum anderen ausreichend Kontrast zur Umgebung des Feldes bietet, wird die Aufmerksamkeit des Benutzers gezielt auf dieses Feld gelenkt. Die Linien sowie der Hinweistext auf den betroffenen Dialogseiten sollen dem Benutzer helfen zu erkennen, wodurch das Problem verursacht wird. Dadurch soll es dem Benutzer erleichtert werden, das Problem selbstständig zu lösen. Gleichzeitig hilft die Linie, indem sie die Regelmäßigkeit des Formulars durchbricht, die Aufmerksamkeit auf das problematische Feld zu lenken.

Fehlerhaft ausgefüllte Eingabefelder

Als fehlerhaft ausgefüllte Felder werden hierbei Felder bezeichnet, bei denen der ausgewählte Wert zu einem Fehler führt, also zum Beispiel verbotene Felder für die ein Wert ausgewählt wurde. Sie werden wie die nicht ausgefüllten erforderlichen Felder farblich in Rot hervorgehoben. Ebenso wird mit einer roten Linie angezeigt, weshalb dieses Feld verboten ist und nicht ausgefüllt werden darf.

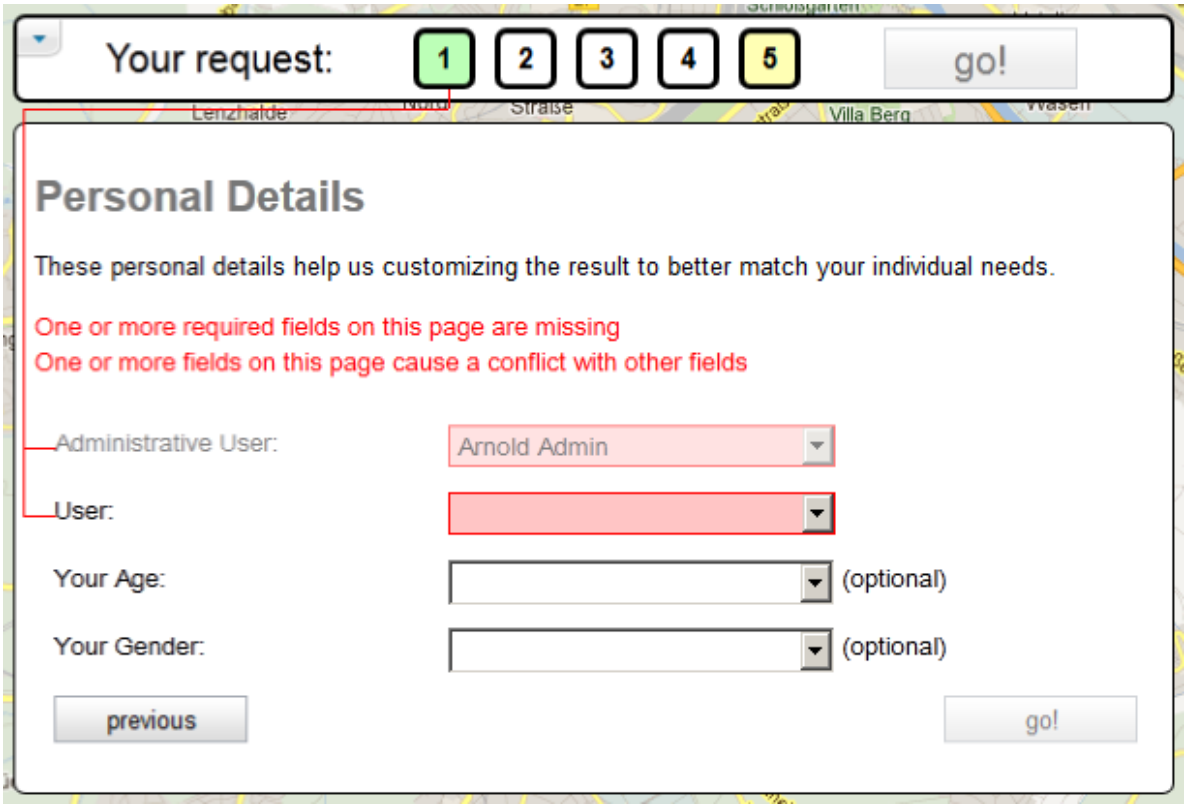


Abbildung 3.7: Beispiel zur Darstellung eines fehlerhaft ausgefüllten Eingabefelds (oben) im Vergleich zu einem nicht ausgefüllten erforderlichen Eingabefeld (unten)

Ungültige Eingaben, zum Beispiel Text in einem Datumsfeld, werden dagegen nicht explizit behandelt, da nur gültige Eingaben anerkannt werden. Das Datumsfeld würde daher solange als nicht ausgefüllt behandelt, bis ein gültiger Datumswert eingegeben wurde. Sofern das Datumsfeld nicht durch eine andere Regelung hervorgehoben wird, zum Beispiel weil es erforderlich ist und aufgrund der ungültigen Eingabe als nicht ausgefüllt gilt, findet keine Hervorhebung statt.

Zur Unterscheidung von nicht ausgefüllten erforderlichen Feldern wird das fehlerhaft ausgefüllte Feld in einem leicht helleren Farbton hervorgehoben (siehe Abbildung 3.7). Zudem wird ein anderer Hinweistext auf der Dialogseite eingeblendet.

Dass es trotz dem Deaktivieren verbotener Felder zu fehlerhaft ausgefüllten Eingabefeldern kommen kann liegt daran, dass ein bereits ausgefülltes und ehemals erforderliches Feld aufgrund der Eingabe in einem anderen Feld, zum Beispiel dem Anfragetyp, zu einem verbotenen Feld werden kann.

Optionale Eingabefelder

Optionale Eingabefelder werden durch einen Hinweis neben dem Eingabefeld als optional gekennzeichnet (siehe die beiden unteren Felder in Abbildung 3.7).

Dynamische Prüfung auf nicht ausgefüllte erforderliche Eingabefelder

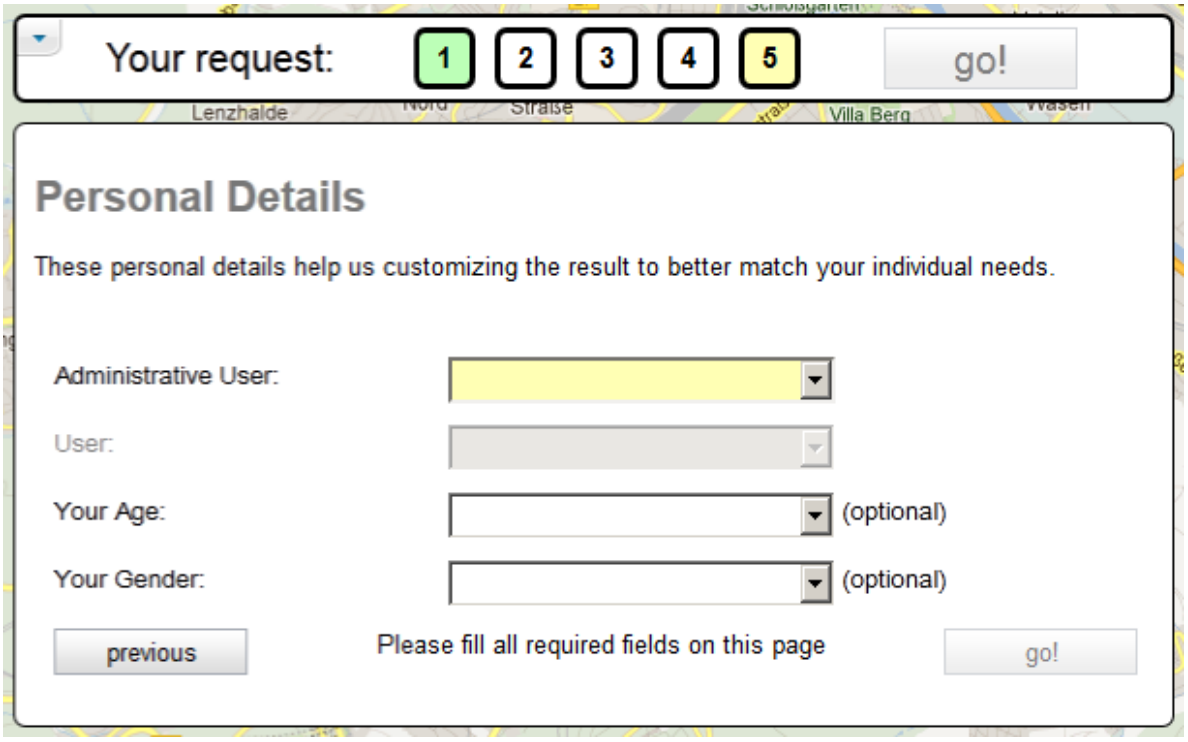
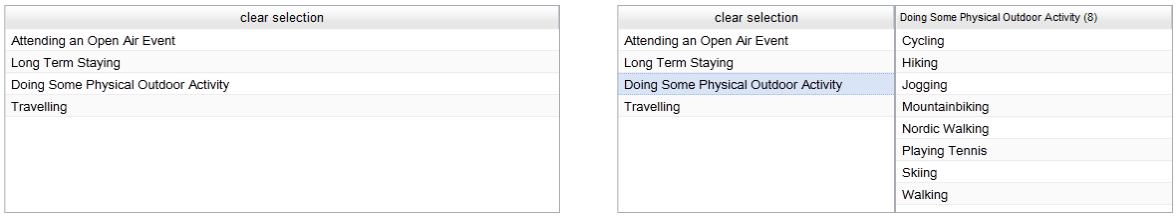


Abbildung 3.8: Beispiel zur Darstellung eines Hinweises auf noch nicht ausgefüllte erforderliche Eingabefelder

Nähert sich der Benutzer mit seinem Mauszeiger einer der Schaltflächen, mit der er zur nächsten oder vorherigen Dialogseite wechseln kann, so werden die auf der aktuellen Dialogseite befindlichen Eingabefelder überprüft. Wurde dabei ein erforderliches Feld nicht ausgefüllt, so wird ein entsprechender Hinweis in der Nähe der Schaltflächen (und damit in der Nähe des Mauszeigers und der Aufmerksamkeit des Benutzers) angezeigt. Zusätzlich wird der Hintergrund der betroffenen Felder gelb gefärbt, so dass der Benutzer die noch nicht ausgefüllten Felder erkennen kann (siehe Abbildung 3.8). Andere Hervorhebungen des Feldes werden dabei überschrieben. Die Farbe Gelb zur Hervorhebung wurde dabei zum einen gewählt, um eine Unterscheidbarkeit zu den in Rot hervorgehobenen Fehlern zu gewährleisten. Zum anderen ist das Gelb eine Anlehnung an eine Verkehrsampel, bei der die gelbe Phase auf die folgende rote Phase hinweist, in diesem Fall also auf den drohenden Fehler, falls das Feld nicht ausgefüllt wird.

Andere Gestaltungs- und Hervorhebungsmöglichkeiten



(a) **(b)**

Abbildung 3.9: Dynamisches Anzeigen abhängiger Felder in **(b)**, nachdem in **(a)** ein Wert ausgewählt wurde

Dynamisches Anzeigen abhängiger Felder Diese Methode kann nur bei Eingabefeldern eingesetzt werden, bei denen nach der Auswahl eines Eingabewertes dieser noch weiter spezifiziert werden muss. Sobald der Benutzer im ersten Eingabefeld eine Eingabe tätigt, wird in der direkten Nähe des Feldes ein zweites Feld angezeigt, in dem die gerade getroffene Eingabe weiter detailliert werden kann (siehe Abbildung 3.9). Ändert sich die Auswahl im ersten Feld, so ändert sich gegebenenfalls auch das zweite Feld. Dabei wird die Auswahl im zweiten Feld entfernt.

Hierbei wird die Aufmerksamkeit des Benutzers durch die visuelle Veränderung des Dialogs auf das neu angezeigte Feld gelenkt. Das Entfernen der Auswahl, falls das zweite Feld bereits sichtbar ist, erzeugt ebenfalls eine, wenngleich weniger auffällige visuelle Änderung. Der Benutzer kann so leicht den Zusammenhang zwischen den beiden Feldern erkennen.

Beschreibungstext neben dem Eingabefeld Sofern mehrere Eingabefelder auf einer Dialogseite sind und diese in der Beschreibung der Dialogseite nicht ausführlich erklärt werden, kann neben dem jeweiligen Feld ein kurzer Hinweistext angezeigt werden (siehe Abbildung 3.10).

Abbildung 3.10: Eingabefeld mit Beschreibungstext

Hinweis beim Senden einer fehlerhaften Anfrage Versucht der Benutzer eine fehlerhafte Anfrage an den Server zu senden, so wird ihm in einem Hinweisfenster erklärt, warum

er seine Anfrage momentan noch nicht an den Server senden kann. Zudem wird ihm dort die Möglichkeit geboten, seine derzeitige Anfrage in den letzten fehlerfreien Zustand zurückzusetzen.

Farbliche Anpassung der Schaltflächen in der Übersicht Die Schaltflächen in der Übersicht der Dialogseiten ändern ihre Farbe, ähnlich einer Ampel, abhängig vom Zustand der zugehörigen Dialogseite (siehe Abbildung 3.11). Hat der Benutzer die zugehörende Dialogseite noch nicht besucht, so ist die Schaltfläche weiß. Hat er die Dialogseite bereits besucht und alle Eingabefelder auf der Dialogseite wurden korrekt ausgefüllt, so ist die zugehörige Schaltfläche grün, bei einem Fehler dagegen rot. Die Farbe der Schaltfläche der Dialogseite, auf der sich der Benutzer gerade aufhält, wird immer in Gelb dargestellt.

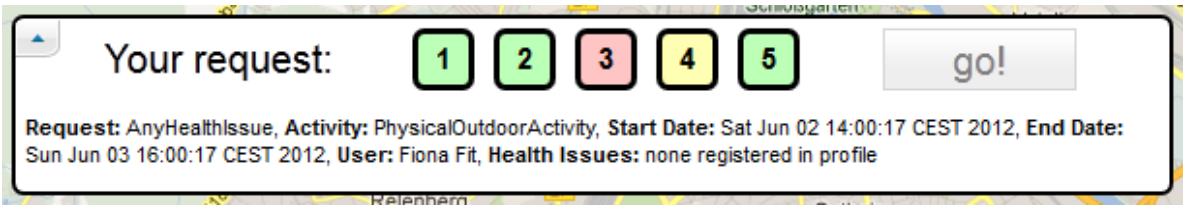


Abbildung 3.11: Hervorhebung der Schaltflächen in der Übersicht der Dialogseiten und Übersicht der bisher erstellten Anfrage

Übersicht der erstellten Anfrage Dem Benutzer wird, während er seine Anfrage erstellt, in einem kleinen Bereich eine Übersicht seiner bisher erstellten Anfrage angezeigt (siehe Abbildung 3.11). Macht der Benutzer eine Eingabe, so wird auch die Anfrage-Übersicht aktualisiert um die Änderung widerzuspiegeln. So erhält der Benutzer zum einen die Rückmeldung, dass seine Eingaben erfolgreich übernommen wurden. Zum anderen hilft es ihm, den Einfluss eines Eingabefelds auf die Anfrage und die Zusammenhänge zwischen den Eingabefeldern nachzuvollziehen.

Kontextsensitive Unterstützung Hierbei werden Informationen aus dem Kontext der Benutzerinteraktion verwendet, um den Benutzer beim Erstellen seiner Anfrage zu unterstützen. So wird zum Beispiel anhand seiner IP sein ungefährer Aufenthaltsort ermittelt und die Karte automatisch an dieser Stelle zentriert.

3.5.4 Nicht verwendete Methoden zur Gestaltung und Hervorhebung

Bei der Entwicklung dieser Lösung wurden noch weitere Varianten zur Hervorhebung von Eingabefeldern betrachtet, aus verschiedenen Gründen jedoch nicht eingesetzt. Diese Varianten sollen nicht unerwähnt bleiben und werden in diesem Abschnitt kurz erläutert.

Beziehungen zwischen Eingabefeldern Ursprünglich war angedacht, die Beziehungen zwischen den Eingabefeldern mit Hilfe von Linien, ähnlich den Linien bei fehlerhaften Eingaben, darzustellen. Standardmäßig sind diese Linien hellgrau, nur die Linie des gerade fokussierten Eingabefelds wird schwarz dargestellt. Die hierfür notwendigen Abhängigkeiten zwischen den Feldern können leicht aus den Regeln (siehe Kapitel 3.3) gewonnen werden, jedoch war die Darstellung der Linien nicht sehr übersichtlich. Zudem wurden die Felder meist mit Feldern auf allen anderen Dialogseiten verbunden, wodurch die daraus zusätzlich gewonnene Information für den Benutzer eher gering ausfiel.

Animationen zur Verdeutlichung von Zusammenhängen oder Hervorhebung Mittels Animationen könnte zum Beispiel der Zusammenhang zwischen einer Dialogseite und der zugehörigen Schaltfläche in der Übersicht verdeutlicht werden, in dem die Dialogseite beim Wechseln der Seite zur Position der Schaltfläche hin zusammenschrumpft. Da beim Einsatz von Animationen jedoch die Gefahr besteht, die Aufmerksamkeit des Benutzers auf eher nebensächliche Dinge zu lenken und die Animation den Benutzer bei seiner Arbeit eventuell behindert, da er erst auf das Ende der Animation warten muss um fortzufahren, wurden in dieser Lösung keine Animationen verwendet.

Erklärung für vorgeschlagene Eingabewerten Mit einer kleinen Schaltfläche „Warum?" bei vorgeschlagenen Eingabewerten kann der Benutzer erfahren, weshalb ihm dieser Wert vorgeschlagen wurde. Da momentan jedoch keine komplexen Mechanismen zum Vorschlagen von Eingabewerten verwendet und die verwendeten bereits an anderer Stelle beschrieben werden, wurde diese Methode nicht eingesetzt.

Gruppierung von Eingabefeldern Durch die Gruppierung (inhaltlich) zusammengehöriger Eingabefelder können lange Dialogseiten mit vielen Eingabefeldern übersichtlicher gestaltet werden. Da in dieser Lösung jedoch keine derart langen Dialogseiten existieren, wird diese Methode nicht verwendet.

Anpassung von Maßeinheiten und Datumsformat an Benutzer Anhand den Einstellungen des Browsers können die Sprach- und Ländereinstellungen des Benutzers ermittelt werden und gegebenenfalls die in den Eingabefeldern verwendeten Maßeinheiten und Datumsformate angepasst werden. Ebenso kann automatisch zu einer entsprechenden Übersetzung (falls vorhanden) der Benutzungsoberfläche gewechselt werden. Der Benutzer kann zudem in seinem Profil angeben, welche Formate er verwenden möchte und dadurch die automatische Anpassung überschreiben. Da in dieser Arbeit jedoch keine Lokalisierung realisiert wurde, wird diese Methode nicht verwendet.

Hervorhebung empfohlener Eingabefelder Eingabefelder, die nicht erforderlich, aufgrund ihres Gewichts jedoch empfohlen sind können ähnlich wie optionale Felder als empfohlen markiert werden. Zur besseren Erkennbarkeit wird Schrift jedoch farbig von den anderen

Beschriftungen abgehoben. Da die hierfür notwendige Funktionalität jedoch noch nicht verfügbar war und somit keine Empfehlungen generiert werden können, wird diese Methode nicht eingesetzt.

Minimieren optionaler Eingabefelder Um die Aufmerksamkeit des Benutzer noch besser auf die erforderlichen Felder zu lenken, können die optionalen Eingabefelder in einem minimierten Bereich zusammengefasst und nur bei Bedarf angezeigt werden. Die zuvor beschriebenen Probleme mit dem Ausblenden von Eingabefeldern sind hierbei weniger stark ausgeprägt, da sich die optionalen Felder (bisher) nicht in Abhängigkeit von anderen Eingaben ändern können. Aufgrund von Schwierigkeiten bei der Umsetzung dieser Funktionalität wird diese Methode jedoch nicht verwendet.

Hervorheben des Grundes eines Verbots Klickt der Benutzer auf ein verbotenes Feld, wird ihm ein Hinweis angezeigt, warum das Feld deaktiviert ist. Da vermutlich jedoch nur wenige Benutzer auf ein sichtbar deaktiviertes Feld klicken und zudem der Klick auf ein deaktiviertes Feld nur unzuverlässig registriert werden konnte, wurde diese Methode zu Gunsten anderer Methoden zurückgestellt.

Ausblenden von verbotenen Eingabefeldern Statt verbotene Felder zu deaktivieren und „ausgegraut" darzustellen, werden sie komplett ausgeblendet. Zusätzlich könnte eine Schaltfläche angezeigt werden, mit denen die ausgeblendeten Felder wieder sichtbar werden. Aufgrund der in Kapitel 3.5.2 beschriebenen Unregelmäßigkeit, die diese Methode in den Dialogen verursachen würde, wurde darauf jedoch verzichtet.

Zulassen verbotener Eingaben Statt durch das Deaktivieren von verbotenen Feldern eine Eingabe zu verhindern, werden bei dieser Methode die verbotenen Felder lediglich wie bereits beschrieben hervorgehoben, während eine Eingabe weiterhin möglich ist. Diese Methode wird nicht verwendet, da aufgrund der Abfolge der Eingabefelder (sofern der Benutzer der Struktur des Dialogs folgt) bereits nach der ersten Dialogseite alle verbotenen Felder und Werte feststehen. Tätigt der Benutzer anschließend eine verbotene Eingabe, wird diese stets zu einem Fehler in der Anfrage führen. Die Methode bringt dem Benutzer in diesem Fall keinen Nutzen. Andererseits hilft das Deaktivieren der Eingabe bei verbotenen Feldern dem Benutzer, eine vorherige falsche Auswahl zu erkennen. Da er das deaktivierte Feld stets angezeigt bekommt, ist er in dem Fall, dass das er das deaktivierte Feld unbedingt für seine Anfrage ausfüllen möchte, gezwungen im Dialog zurückzugehen und seine vorherigen Eingaben zu überprüfen und gegebenenfalls zu ändern. Dabei lernt er gleichzeitig die Zusammenhänge zwischen den Eingaben und kann sich so ein besseres Modell der Anwendung bilden.

4 Implementierung

In diesem Kapitel wird beschrieben, wie die zuvor in Kapitel 3 erläuterten Methoden im Rahmen dieser Arbeit realisiert wurden. Die Implementierung baut auf der bereits existierenden alten Benutzungsoberfläche auf. Dabei wurde darauf geachtet, bestehende Schnittstellen nach Möglichkeit beizubehalten und Neuerungen zu kapseln, um eine einfache Integration der neuen Benutzungsoberfläche in das System zu erlauben.

Die Benutzungsoberfläche wird, wie bereits die alte Benutzungsoberfläche, mit Hilfe des Google Web Toolkits[1] (GWT) sowie SmartGWT[2] realisiert. GWT erlaubt es, eine Webanwendung in Java zu entwickeln und diese später in JavaScript zu konvertieren. SmartGWT ist eine Sammlung von Elementen, wie zum Beispiel Buttons und Tables, die das GWT erweitern.

4.1 Architektur der Benutzungsoberfläche

Für die neue Benutzungsoberfläche wurde die Architektur der alten Benutzungsoberfläche beibehalten und um weitere Komponenten ergänzt. Abbildung 4.1 zeigt das Zusammenspiel dieser Komponenten, wobei neue Komponenten mit einem gestrichelten Rahmen und für diese Arbeit unwichtige Komponenten in Grau dargestellt werden. Die wichtigsten Komponenten dieser Architektur sind:

Presenter Der Presenter reagiert auf die Eingaben des Benutzers, steuert die Darstellung der View und kontrolliert den Ablauf der Anwendung.

View Die View ist für den Aufbau und die Aktualisierung der für den Benutzer sichtbaren Elemente verantwortlich.

EventBus Mit Hilfe des EventBus können Events an beliebiger Stelle erzeugt und an einer (oder mehreren) anderen Stellen verarbeitet werden, ohne dass der Empfänger eines Events den Erzeuger kennen muss.

QueryManager Der QueryManager verwaltet die anhand der Eingaben des Benutzers erzeugte Anfrage. Dazu werden zum einen die Eingaben gespeichert, zum anderen der sogenannte QueryString aus den Eingaben generiert, mit dem die Anfrage zur Bearbeitung an den Server gesendet wird.

[1] https://developers.google.com/web-toolkit/
[2] http://www.smartclient.com/product/smartgwt.jsp

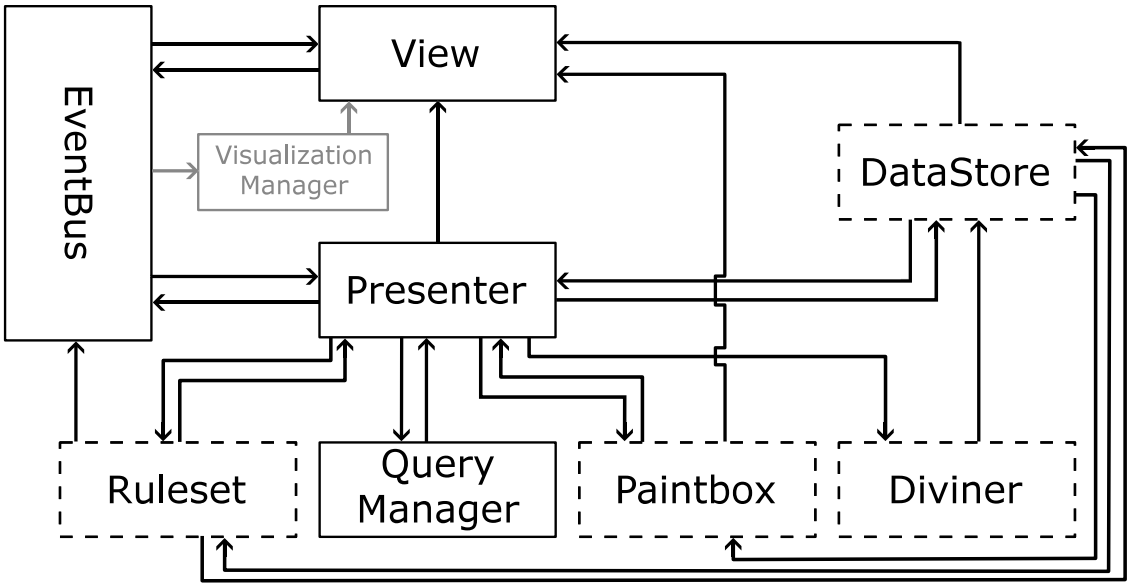


Abbildung 4.1: Architektur der neuen Benutzungsoberfläche von PESCaDO

VisualizationManager Der VisualizationManager kontrolliert die Visualisierung des Ergebnisses, das vom Server zurück geliefert wird.

Mit dieser Architektur wird der Großteil der Anwendung auf die Client-Seite in den Browser des Benutzers ausgelagert und die Kommunikation mit dem Server auf das notwendige Minimum reduziert.

Für die neue Benutzungsoberfläche wurden ein neuer *Presenter* und eine neue *View* implementiert und auch der *QueryManager* wurde angepasst. Alle anderen Komponenten wurden unverändert übernommen.

4.2 Implementierung der Benutzungsoberfläche

Zur Umsetzung der zuvor beschriebenen Lösung werden zusätzliche Komponenten und Klassen benötigt. Diese zusätzlichen Klassen sind im Java-Package „balance" enthalten und werden in diesem Abschnitt zusammen mit den Änderungen an den bestehenden Komponenten beschrieben.

4.2.1 Helferklassen

Für die Realisierung der Funktionen der Lösung werden eine Reihe von Helferklassen benötigt, deren überwiegende Aufgabe der Transport von Daten ist. Im Nachfolgenden werden die vorhandenen Helferklassen und ihr Verwendungszweck kurz erläutert.

BalanceRule Diese Klasse repräsentiert eine Regel (siehe Kapitel 3.3) und besitzt daher die Attribute *Rule*, *Factor* und *Affected*. Zusätzlich besitzt sie das Attribut *Name*, mit dem ein Name (beziehungsweise eine Beschreibung) für die Regel festgelegt werden kann, was die spätere Arbeit mit den Regeln wesentlich erleichtert.

BalanceWeight Diese Klasse repräsentiert das Gewicht einer Klasse der Ontologie. Sie besitzt die Attribute *Weight* und *OntologyReference*, in der der Name der zugehörigen Klasse aus der Ontologie gespeichert wird.

DialogError Diese Klasse repräsentiert einen Fehler, der innerhalb der Anfrage gefunden wurde (siehe Kapitel 3.4). Sie besitzt die Attribute

- *Reason*, in dem der Name der den Fehler verursachenden Klasse gespeichert wird.
- *Affected*, in dem die Namen aller von dem Fehler betroffenen Klassen gespeichert wird.
- *Type*, in dem die Art des Fehlers gespeichert wird.

DialogErrorTypeEnum Diese Enumeration listet die möglichen Fehlerarten auf, die in *DialogError* verwendet werden. Die Fehlerarten sind *REQUIRED_FIELD_MISSING* und *FORBIDDEN_FIELD*.

FieldValue Mit dieser Klasse wird ein möglicher Eingabewert für ein *FieldWidget* (siehe Kapitel 4.2.3) beschrieben. Die Klasse besitzt dazu die Attribute

- *OntologyReference*, in dem die zum Eingabewert gehörende Klasse in der Ontologie gespeichert wird.
- *Name*, in dem der innerhalb des *FieldWidgets* angezeigte Text für diesen Eingabewert gespeichert wird.
- *Description*, in dem eine Beschreibung zu dem Eingabewert gespeichert werden kann.
- *IsClass*, in dem gespeichert wird, ob es sich bei dem Eingabewert um eine Klasse der Ontologie (*IsClass* = *true*) oder eine Instanz einer Klasse (*IsClass* = *false*) handelt.
- *SubValues*, in dem weitere *FieldValues* gespeichert werden können, welche die eventuell zum aktuellen *FieldValue* gehörenden Unterauswahlen (Subklassen oder Instanzen) beschreiben.

HighlightTypeEnum Diese Enumeration listet die verschiedenen Hervorhebungsvarianten auf die verwendet werden können. Jede Variante hat drei verschiedene Stufen, *SUCCESS*, *INFO* und *WARN*. Jede Stufe wird dabei in einer anderen Farbe hervorgehoben. Die Varianten sind:

- *SIMPLE* spezifiziert keine weiteren Details.
- *BORDER* gibt an, dass der Rahmen des Widgets hervorgehoben werden soll.
- *BACKGROUND* gibt an, dass der Hintergrund des Widgets hervorgehoben werden soll.
- *FIELDBORDER* gibt an, dass der Rahmen des Eingabefelds innerhalb des Widgets hervorgehoben werden soll.
- *FIELDBACKGROUND* gibt an, dass der Hintergrund des Eingabefelds innerhalb des Widgets hervorgehoben werden soll.

Die einzelnen Varianten können miteinander kombiniert werden, zum Beispiel *BORDER_WARN + FIELDBACKGROUND_INFO*.

QueryRepresentationPart Diese Klasse wird verwendet, um den einzelnen Teilen des erzeugten QueryStrings ein oder mehrere Eingabefelder zuzuordnen. Die Klasse besitzt die folgenden Attribute:

- *RepresentationString*, in dem der angezeigte Teil des QueryStrings gespeichert wird.
- *RelatedClasses*, in dem die Namen der Klassen der Ontologie gespeichert werden, anhand derer der in *RepresentationString* gespeicherte Teil der Anfrage erzeugt wird.

StoredValue Diese Klasse wird verwendet, um die vom Benutzer getätigten Eingaben im lokalen Datenmodell zu speichern. Sie besitzt die folgenden Attribute:

- *OntologyClass*, in dem der Name der Klasse der Ontologie gespeichert wird.
- *Value*, in dem der für *OntologyClass* ausgewählte Wert gespeichert wird.
- *UserCreated* gibt an, ob der Wert durch die Eingabe des Benutzers ausgewählt wurde oder von der Anwendung bestimmt wurde.
- *Source*, in dem der Name der Ontologieklasse des *FieldWidgets* gespeichert wird, in dem die mit dieser *StoredValue*-Klasse repräsentierte Eingabe getätigt wurde.

UserInputValue Mit dieser Klasse wird die Information über eine vom Benutzer getätigte Eingabe transportiert. Sie besitzt die folgenden Attribute:

- *Source*, in dem die Ontologieklasse des *FieldWidgets* gespeichert wird, in dem der Benutzer die Eingabe getätigt hat.

- *Value*, in dem die ausgewählten Eingabewerte gespeichert werden. Die Eingabewerte werden jeweils durch einen eindeutigen Namen einer Klasse oder Instanz der Ontologie dargestellt. *Value* kann auch leer sein.
- *IsClass* gibt an, ob die Eingabewerte eine Klasse der Ontologie oder eine Instanz einer Klasse repräsentieren (siehe Kapitel 3.4).
- *SourceWidget* enthält einen Verweis auf das *FieldWidget*, in dem die Eingabe getätigt wurde.

4.2.2 Presenter

Der *Presenter* kontrolliert den Großteil der Reaktionen der Anwendung auf die Eingaben des Benutzers. Dazu gehören unter anderem das Anzeigen der verschiedenen Bereiche der Benutzungsoberfläche und das Anzeigen der einzelnen Dialogseiten.

Der *Presenter* behandelt zudem das von den einzelnen *FieldWidgets* bei einer Benutzereingabe ausgelöste Event. Wird ein solches Event aufgefangen, werden zunächst die enthaltenen Benutzereingaben im *DataStore* gespeichert und mit dem *QueryManager* die bestehende Anfrage aktualisiert. Danach werden mit dem *Ruleset* die Gewichte aller Widgets neu berechnet. Anschließend wird mit dem *Ruleset* geprüft, ob die aktuelle Anfrage Fehler enthält. Eventuell gefundene Fehler werden wiederum im *DataStore* gespeichert. Abschließend werden mit der *Paintbox* eventuell vorhandene Fehler hervorgehoben. Wurden keine Fehler hervorgehoben, wird das aktuelle Datenmodell vom *DataStore* als fehlerfreie Anfrage gespeichert. Wurden insgesamt keine Fehler gefunden, wird die Schaltfläche zum Senden der Anfrage aktiviert. Wurden dagegen Fehler gefunden, wird die Schaltfläche deaktiviert.

Nach jeder Eingabe durch den Benutzer wird zudem die Anfrageübersicht aktualisiert, in der der Benutzer die Details seiner bisher erstellten Anfrage sehen kann.

4.2.3 View

Die *View* erzeugt und verwaltet alle sichtbaren Elemente der Benutzungsoberfläche. Das umfasst neben der Webseite und deren einzelner Seiten auch die *DialogPages* und die enthaltenen *FieldWidgets*, aus denen der Dialog für den Benutzer aufgebaut wird.

Während die meisten Funktionen zur Reaktion auf Benutzereingaben im *Presenter* gebündelt sind, wurden einige Funktionen, die die Darstellung der Benutzungsoberfläche beeinflussen, der Einfachheit halber und um die Erzeugung unnötiger Events zu vermeiden in der *View* implementiert.

DialogWidget und FieldWidget

Als Widgets werden im GWT allgemein Steuerelemente, wie Schaltflächen oder Textfelder, bezeichnet. Im Rahmen dieser Arbeit werden zusätzlich sogenannte *DialogWidgets* eingeführt. Ein *DialogWidget* repräsentiert einen Teil des Dialogs der dem Benutzer angezeigt wird, zumeist also ein Eingabefeld und dessen Klasse aus der Ontologie. Da für die verschiedenen Eingaben unterschiedliche Eingabemethoden erforderlich sein können, werden von der abstrakten *DialogWidget* Klasse verschiedene andere Widgets, sogenannte *FieldWidgets*, mit jeweils angepassten Eingabemethoden, zum Beispiel einer Dropdown-Liste oder einer Miller Column, abgeleitet. Von *DialogWidget* erben diese Widgets ebenfalls eine abstrakte Methode zum Hervorheben des Widgets (wie in Kapitel 3.5 beschrieben). Die Methode zum Hervorheben des Widgets wird von jedem Widget in einer an die vom Widget verwendete Eingabemethode angepassten Art und Weise implementiert. So können später verschiedene Widgets einfach hervorgehoben werden, da nur die Art der Hervorhebung mitgeteilt werden muss. Die eigentliche Hervorhebung realisiert jedes Widget für sich selbst.

Beim Erstellen eines Widgets wird für dieses festgelegt, welche Klasse der Ontologie es repräsentiert, welche Eingabewerte möglich sind und welcher Text als Beschriftung des Widgets angezeigt wird. So kann ein einzelnes Widget auch für mehrere Klassen der Ontologie verwendet werden.

Wählt der Benutzer in einem *FieldWidget* einen Wert aus, so wird ein Event ausgelöst, das den eingegebenen Wert mit Hilfe der Klasse *UserInputValue* (siehe Abschnitt Helferklassen 4.2.1) transportiert.

Eine Ausnahme bildet hierbei das *PageFooterWidget*. Dieses Widget wird verwendet, um am Seitenende die „zurück" und „weiter" Schaltflächen anzuzeigen.

DialogPage

Eine *DialogPage* repräsentiert eine Seite des Dialogs mit dem der Benutzer seine Anfrage erstellt und kann dazu beliebig viele *FieldWidgets* enthalten. Die *DialogPage* verwaltet die enthaltenen Widgets und kann diese zum Beispiel deaktivieren, falls das Widget eine verbotene Eingabe darstellt. Ebenso sorgt die *DialogPage* dafür, dass die enthaltenen Widgets beim Öffnen der entsprechenden Dialogseite aktualisiert werden.

Beim Erstellen der *DialogPage* wird neben den enthaltenen Widgets auch der Beschreibungstext der *DialogPage* festgelegt. Dieser Text wird oberhalb der auf der Seite enthaltenen Widgets angezeigt. Zwischen Beschreibungstext und Widgets können zusätzliche Hinweise, zum Beispiel im Fall eines Fehlers, angezeigt werden.

Die *DialogPage* kann anschließend in die Benutzungsoberfläche eingebunden und angezeigt werden. Da *DialogPage* ebenfalls von *DialogWidget* erbt, besitzt auch *DialogPage* die Möglichkeit zur Hervorhebung. Statt einer Klasse der Ontologie werden den Dialogseiten eindeutige IDs zugeordnet.

4.2.4 QueryManager

Der *QueryManager* wurde im Vergleich zur alten Benutzungsoberfläche nur gering erweitert, so dass bei Änderungen der Anfragedaten automatisch die entsprechenden *QueryRepresentationParts* erstellt werden. Die *QueryRepresentationParts* werden verwendet, um dem Benutzer eine Übersicht über seine bisher erstellte Anfrage anzeigen zu können.

4.2.5 DataStore

Der *DataStore* implementiert das Singleton-Muster und ist somit global verfügbar. Im *DataStore* wird das aktuelle Datenmodell vorgehalten. Dazu gehören:

- Die vom Benutzer getätigten Eingaben. Alle Eingabe werden in einer Liste von *StoredValue* gespeichert.
- Die aktuellen Gewichte aller Eingabefelder, die in einer Liste von *BalanceWeight* gespeichert werden.
- Alle in der Anfrage momentan vorhandenen Fehler, die in einer Liste von *DialogError* gespeichert werden.
- Eine Liste der vom Benutzer bereits besuchten Dialogseiten.
- Die Information, ob die Anfrage bereits an den Server gesendet wurde.
- Eine die Hierarchie der Ontologieklassen beschreibende Klassenstruktur.
- Ein Verweis auf den *QueryManager*.

Neben Funktionen zum Hinzufügen, Abrufen oder Löschen dieser Informationen besitzt der *DataStore* eine Methode, mit der das gesamte aktuelle Datenmodell gespeichert werden kann. Mit einer zweiten Methode kann es dann später wieder geladen werden, wobei alle zwischenzeitlichen Änderungen rückgängig gemacht werden.

4.2.6 Diviner

Der *Diviner* ist für die Umsetzung von Funktionen zuständig, die dem Benutzer Arbeit abnehmen sollen. Hierzu gehört zum einen, dass der *Diviner* anhand der IP des Benutzers versucht, seinen ungefähren Standort zu ermitteln. Gelingt dies, wird die Karte der Benutzungsoberfläche auf den gefundenen Standort zentriert, ansonsten wird sie auf einen Standardpunkt zentriert. Zum anderen verwendet der *Diviner* im Profil gespeicherte Informationen, um im Voraus die zugehörigen Eingabefelder auszufüllen und die entsprechenden Werte im *DataStore* zu hinterlegen. Diese Werte werden dabei als nicht benutzergeneriert (*StoredValue.UserCreated = false*) markiert. Sollten die Werte später zu einem Fehler führen, werden sie ohne einen *DialogError* zu erzeugen wieder entfernt.

4.2.7 Paintbox

Die *Paintbox* verwaltet die verwendeten Widgets und ist für das Hervorheben einzelner Widgets verantwortlich. Bei der Erstellung eines neuen *FieldWidgets* oder einer neuen *DialogPage* werden diese bei der *Paintbox* registriert. So kann später schnell anhand einer Klasse der Ontologie, beziehungsweise der ID der Dialogseite, das dazugehörende Widget gefunden werden. Ebenso werden in der *Paintbox* die verschiedenen Farben zu Hervorhebung definiert und festgelegt, welche Fehlerart wie hervorgehoben werden soll (siehe *HighlightTypeEnum*).

Für das Hervorheben von Feldern implementiert die *Paintbox* eine Reihe von Funktionen, um

- alle fehlerhaft ausgefüllten Felder auf vom Benutzer bereits besuchten Dialogseiten in Rot hervorzuheben,
- nicht ausgefüllte erforderliche Felder auf der aktuellen Dialogseite in Gelb hervorzuheben,
- die Schaltflächen in der Übersicht der Dialogseiten entsprechend dem Zustand der zugehörigen Dialogseite hervorzuheben,
- die Linien zwischen an einem Fehler beteiligten Widgets zu zeichnen (siehe Kapitel 3.5.3).

Zur Hervorhebung der Fehler werden die im *DataStore* gespeicherten Fehler verwendet. Für jeden gespeicherten Fehler wird das verursachende Widget hervorgehoben und eine Linie zwischen dem verursachenden Widget und allen betroffenen Widgets gezeichnet.

4.2.8 Ruleset

Das *Ruleset* lädt zu Beginn die im Voraus generierten Regeln (siehe Kapitel 3.3) und erstellt dabei für jede in den Regeln verwendete Klasse der Ontologie eine Liste mit allen Regeln, die diese Klasse betreffen. Dies wird verwendet, um später leichter die indirekt von einer Regel betroffenen Klassen finden zu können.

Die beiden Aufgaben des *Rulesets* sind das Berechnen der aktuellen Gewichte für alle Eingabefelder und -werte sowie das Auswerten der aktuellen Eingaben (siehe Kapitel 3.4). Dazu werden die im *DataStore* gespeicherten Gewichte und Eingaben des Benutzers verwendet. Wird beim Auswerten der aktuellen Eingaben ein Fehler entdeckt, so wird ein neuer Fehler mittels *DialogError* erzeugt. Nachdem alle Eingaben überprüft wurden, wird eine Liste der gefundenen Fehler zurückgegeben.

4.3 Generierung der Regeln

Da die Generierung der Regeln nur einmalig erforderlich ist und nur bei einer Änderung der PESCaDO Ontologie wiederholt werden muss, wurde der *RuleGenerator* als eine von der Benutzungsoberfläche getrennte, eigenständige Java-Anwendung realisiert.

Für die Arbeit mit einer Ontologie werden die OWL API[3] sowie ein Reasoner benötigt. Mit einem Reasoner kann die Ontologie auf semantischer Ebene bearbeitet werden, es kann zum Beispiel leicht nach Klassen gesucht werden, die in einer bestimmten Beziehung zu einer anderen Klasse stehen. Für den *RuleGenerator* wurde HermiT[4] als Reasoner verwendet.

Zur Erstellung der Regeln arbeitet der *RuleGenerator* eine Liste von relevanten Klassen der Ontologie ab. Diese Liste ist wird im Voraus definiert und ändert sich während der Ausführung der Anwendung nicht. Nur wenn alle an einer Beziehung beteiligten Klassen in dieser Liste enthalten sind, wird die Art der Beziehung untersucht und entsprechend Kapitel 3.3 eine Regel erzeugt. Auf diese Weise wird die Anzahl der erzeugten Regeln auf die Zahl der für die Benutzungsoberfläche relevanten Regeln reduziert. Andernfalls würden viele Regeln generiert, die Klassen betreffen, welche in der Benutzungsoberfläche durch kein Widget repräsentiert werden.

Listing 4.1 Die Basisregel

```
<BalanceRule>
	<Rule>
		true
	</Rule>
	<Affected>
		pescado:Request
	</Affected>
	<Factor>
		1024
	</Factor>
	<Name>
		Base Rule
	</Name>
</BalanceRule>
```

Nachdem alle Klassen dieser Liste bearbeitet wurden, werden zunächst eventuell doppelt erzeugte Regeln entfernt. Dann wird die sogenannte „Basisregel" (siehe Listing 4.1) hinzugefügt. Diese Regel wird immer als wahr ausgewertet und sorgt dafür, dass die Klasse *Request* erforderlich ist. Ohne die Basisregel würde bei der Auswertung der Eingaben auch eine leere Anfrage als fehlerfrei akzeptiert werden. Anschließend werden die gefundenen Regeln in einer XML-Datei gespeichert. Diese XML-Datei wird später vom *Ruleset* der Benutzungsoberfläche geladen.

[3]http://owlapi.sourceforge.net/
[4]http://www.hermit-reasoner.com/

Anhand der für diese Arbeit verwendeten Version der PESCaDO Ontologie werden so insgesamt 56 Regeln generiert. Eine dieser Regeln ist in Listing 4.2 zu sehen. Wie dort gut zu erkennen ist, beinhalten die erzeugten Regeln das in Kapitel 4.2.1 beschriebene Attribut *Name*. Es wird verwendet, um die Funktion der generierten Regel kurz zu beschreiben. In der Beispielregel sorgt die Klasse *WarningRequest* aus der Ontologie dafür, dass die Klasse *AnyHealthIssue* erforderlich wird, sobald *WarningRequest* ausgewählt wird und weder *AnyRestrictionForPrivateTransport* noch *WarningDueToEnvironmentalConditions* ausgewählt wurden, da *AnyHealthIssue* eine Subklasse von *WarningRequest* ist.

Listing 4.2 Eine der insgesamt 56 vom *RuleGenerator* erzeugten Regeln

```
<BalanceRule>
       <Rule>
              pescado:WarningRequest && ! ( pescado:AnyRestrictionForPrivateTransport ||
                  pescado:WarningDueToEnvironmentalConditions )
       </Rule>
       <Affected>
              pescado:AnyHealthIssue
       </Affected>
       <Factor>
              1024
       </Factor>
       <Name>
              pescado:WarningRequest causes pescado:AnyHealthIssue to be required (subclass)
       </Name>
</BalanceRule>
```

5 Evaluation

Mit der in dieser Arbeit entwickelten Lösung wurde eine Benutzerstudie durchgeführt, die zum einen klären sollte, ob die entwickelten Methoden zur intelligenteren Benutzerführung effektiv sind. Zum anderen wurde die Effizienz der neuen Benutzungsoberfläche im Vergleich zur alten Benutzungsoberfläche untersucht.

In diesem Kapitel wird zunächst die Vorgehensweise der Benutzerstudie beschrieben. Anschließend werden die Ergebnisse der Studie vorgestellt und diskutiert.

5.1 Geplanter Ablauf der Evaluation

Mit der Evaluation der entwickelten Lösung sollen die folgenden Thesen überprüft werden:

1. Die dynamische Hervorhebung einzelner Felder hilft dem Benutzer, alle erforderlichen Felder auszufüllen, fehlerhafte Eingaben zu vermeiden, fehlerhafte Eingaben zu erkennen und zu beheben und somit den Dialog schneller abzuschließen.
2. Die Struktur des Eingabedialogs hilft dem Benutzer dabei, alle erforderlichen Felder auszufüllen und zu erkennen, wie er mit dem Dialog fortfahren kann.
3. Die Hervorhebung von Abhängigkeiten zwischen den Feldern im Fehlerfall hilft dem Benutzer, die Fehlerursache zu erkennen und den Fehler zu beheben.
4. Die Beschreibungstexte helfen dem Benutzer zu entscheiden, wie er das Eingabefeld ausfüllen soll.
5. Die neue Benutzungsoberfläche unterstützt den Benutzer beim Erstellen einer Anfrage besser als die alte Benutzungsoberfläche.

Um diese Thesen zu überprüfen, müssen mehrere Varianten der neuen Benutzungsoberfläche untereinander und mit der alten Oberfläche verglichen werden. Da die entwickelte Lösung webbasiert ist, bietet es sich an eine Online-Benutzerstudie durchzuführen. Dies hat zudem den Vorteil, dass mit relativ geringem Aufwand eine größere Zahl von Testpersonen erreicht werden kann. Während der Studie wird dem Tester zu jeder Benutzungsoberfläche eine Aufgabe gestellt, die er anschließend eigenständig lösen soll. Neben verschiedenen Messungen, die während der Benutzung der alten und neuen Oberfläche vorgenommen werden, werden dem Tester nach Abschluss jeder Aufgabe Fragen zu der gerade verwendeten Benutzungsoberfläche gestellt.

Um die Motivation des Testers möglichst hoch zu halten und so möglichst viele vollständige Durchläufe zu erhalten, wird die für die Studie benötigte Zeit so kurz wie möglich gehalten. Des Weiteren erhält der Tester bei Abschluss der Studie die Möglichkeit an einem Gewinnspiel teilzunehmen.

5.1.1 Varianten und Aufgaben

Um die Thesen 1 und 3 überprüfen zu können, werden zwei Versionen der neuen Benutzungsoberfläche verwendet. Eine Version, in der alle in dieser Lösung entwickelten Methoden zur Hervorhebung deaktiviert sind (als Version *N* bezeichnet), sowie eine Version, die der entwickelten Lösung dieser Arbeit entspricht (als Version *N+* bezeichnet). Von der alten Benutzungsoberfläche wird nur eine Version *O* verwendet. Da die Dauer der Studie kurz gehalten werden soll, erhält der Tester immer nur zwei Versionen der Benutzungsoberflächen zum testen.

Zu jeder dieser Versionen erhält der Benutzer eine dieser drei Aufgaben:

A Stellen Sie sich vor, Sie sind Fiona Fit und wollen am nächsten Wochenende in der Umgebung von Böblingen wandern gehen. Benutzen Sie PESCaDO um festzustellen, ob dabei mit gesundheitlichen Problemen zu rechnen ist.

B Stellen Sie sich vor, Sie sind Arnold Admin und wollen das nächste Wochenende in der Umgebung von Paris verbringen. Benutzen Sie PESCaDO, um die dortige Luftqualität zu überprüfen.

C Ihr Kollege Arnold Admin möchte sich Ihnen anschließen, hat jedoch Bedenken wegen möglicherweise erhöhten Ozonwerten. Benutzen Sie PESCaDO und verändern Sie ihre bestehende Anfrage, um die Luftqualität am nächsten Wochenende in der Umgebung von Böblingen zu prüfen.

Die Aufgaben sind dabei so formuliert, dass die Namen der Personen den in PESCaDO vorhandenen Profilen entsprechen. Zudem wurde darauf geachtet, dass die verwendeten Begriffe den in PESCaDO verwendeten Begriffen nach Möglichkeit gleichen. So kann der Tester schnell den Zusammenhang zwischen den in der Aufgabe geforderten Bedingungen und den entsprechenden Eingabefeldern erkennen, wodurch die Gefahr einer falsch verstandenen Aufgabe und daher abweichender Anfragen, die eventuell nicht mehr vergleichbar sind, verringert wird.

Während Aufgabe A das Ausfüllen aller Eingabefelder erfordert, führt bei Aufgabe B das Auswählen einer Aktivität zu einem Fehler. Aufgabe C wird stets als zweite Aufgabe nach Aufgabe A gestellt. Aufgabe C ist dabei so gestaltet, dass die in der ersten Aufgabe erstellte Anfrage erhalten bleibt und vom Benutzer angepasst werden muss. Durch die Anpassung werden sich dabei zwangsläufig Fehler ergeben, die vom Benutzer behoben werden sollen.

Aus diesen Versionen der Benutzungsoberflächen und den Aufgaben ergeben sich eine Reihe von Varianten. Um eine Verfälschung der Ergebnisse durch Übungseffekte bei der Bearbeitung der zweiten Aufgabe zu minimieren, müssen zudem die Varianten in umgekehrter

Reihenfolge durchgeführt werden. Da jedoch Zeit und Mittel für diese Benutzerstudie relativ beschränkt sind, muss die Anzahl von Varianten, die an die Tester ausgegeben werden, möglichst gering gehalten werden, um eine ausreichend große Zahl von Testern für jede Variante erreichen zu können.

Es werden daher die folgenden sechs Varianten an die Testpersonen ausgegeben:

1. Version *N* mit Aufgabe A, anschließend Version *N+* mit Aufgabe B
2. Version *N+* mit Aufgabe A, anschließend Version *N+* mit Aufgabe C
3. Version *O* mit Aufgabe A, anschließend Version *N+* mit Aufgabe B
4. Version *N* mit Aufgabe A, anschließend Version *O* mit Aufgabe B
5. Version *N+* mit Aufgabe A, anschließend Version *O* mit Aufgabe B
6. Version *O* mit Aufgabe A, anschließend Version *N* mit Aufgabe B

Die folgenden Varianten werden nicht ausgegeben:

- Version *N* mit Aufgabe A, anschließend Version *N+* mit Aufgabe C, da dieser Fall mit Hervorhebung bereits durch Variante 1 abgedeckt ist und Aufgabe C als Sonderfall zu betrachten ist, der einen Fehler erzeugen soll.
- Version *N+* mit Aufgabe A, anschließend Version *N* mit Aufgabe B, da die Ergebnisse durch den Carry-over-Effekt [Hub05] beeinflusst würden, wenn die „angenehmere" Variante zuerst präsentiert wird. Stattdessen werden die Ergebnisse von verschiedenen Durchläufen betrachtet, die jeweils mit Version *N+* und Aufgabe A, beziehungsweise Version *N* und Aufgabe A beginnen.
- Version *O* mit Aufgabe A, anschließend Version *N+* mit Aufgabe C, da dieser Fall bereits durch Variante 3 mit abgedeckt ist. Aufgabe C ist lediglich eine Abwandlung von Aufgabe B die zu Fehlern führen soll, um so die Hervorhebungen der neuen Oberfläche gezielter testen zu können.
- Version *N/N+/O* mit Aufgabe C, anschließend Version *N/N+/O* mit Aufgabe A/B, da dies Aufgaben bedingt nicht möglich ist.

Um eine gleichmäßige Verteilung der Varianten zu garantieren, wird ein mit dem Wert 1 initialisierter Zähler x verwendet. Der aktuelle Tester erhält dabei Variante x, anschließend wird der Zähler um 1 erhöht. Überschreitet der Zähler die Zahl der Varianten, wird er auf 1 zurückgesetzt.

Sollte die Testperson Schwierigkeiten haben, die Aufgabe erfolgreich abzuschließen, so kann der Dialog jederzeit durch den Tester abgebrochen werden. In diesem Fall wird der Tester zunächst gebeten, den Grund für den Abbruch zu beschreiben. Anschließend wird er zum Fragebogen weitergeleitet und kann die Evaluation fortsetzen.

Eine Aufgabe endet mit dem Absenden der erstellten Anfrage oder wenn sie von der Testperson abgebrochen wird.

5.1.2 Fragebogen

Um die Erfahrungen der Testpersonen mit den Benutzungsoberflächen auswerten zu können, wird der Tester nach jeder bearbeiteten Aufgabe (im Verlauf einer Variante also zweimal) gebeten, eine Reihe von Fragen zu beantworten. Die Fragen sind in vier Fragebogenseiten F1 bis F4 unterteilt und werden in Abhängigkeit von der Variante, die der Tester bearbeitet hat, angezeigt.

Die einzelnen Fragen sind als Aussagen formuliert, denen der Tester zustimmen oder nicht zustimmen kann. Hierfür wird eine 5-Punkte Likert-Skala verwendet, was zusätzlich eine neutrale Haltung zur jeweiligen Aussage erlaubt. Ein Punkt auf der Skala entspricht „nicht zutreffend", fünf Punkte auf der Skala entsprechen „zutreffend". Im Nachfolgenden sind die Fragen zur einfacheren Referenzierung durchgehend nummeriert.

Die Fragebogenseite F1 enthält die folgenden Fragen:

1. Die Webseite ist klar strukturiert
2. Die Webseite ist einfach zu verstehen
3. Die Webseite ist einfach zu bedienen
4. Die Webseite verhält sich erwartungsgemäß
5. Die Webseite macht ersichtlich, welcher Schritt als nächstes ausgeführt werden muss
6. Die Abfolge der Schritte ist nachvollziehbar
7. Die Abfolge der Schritte hat mir geholfen, alle erforderlichen Felder auszufüllen
8. Die Eingabemöglichkeiten passen zu den einzugebenden Werten
9. Die Beschreibungstexte auf den einzelnen Seiten sind verständlich
10. Die Beschreibungstexte haben mir geholfen, falsche Eingaben zu vermeiden

Die Fragebogenseite F2 enthält die folgenden Fragen:

11. Die farbige Hervorhebung von Feldern ist nachvollziehbar
12. Die farbige Hervorhebung von Feldern ist übersichtlich
13. Die farbige Hervorhebung von Feldern hat mir geholfen, falsche Eingaben zu vermeiden
14. Die farbig hervorgehobenen Felder haben mir geholfen, das damit verknüpfte Problem zu erkennen
15. Die farbig hervorgehobenen Felder haben mir geholfen, das bestehende Problem zu beheben
16. Es wurden zu viele Felder hervorgehoben
17. Es wurden zu wenige Felder hervorgehoben

18. Die farbige Hervorhebung der Teilschritte neben der „senden" Schaltfläche ist hilfreich
19. Die Funktion der roten Linien zur Hervorhebung von Fehlern ist ersichtlich
20. Die roten Linien zur Hervorhebung von Fehlern sind hilfreich
21. Das Deaktivieren einzelner Felder ist nachvollziehbar
22. Das Deaktivieren einzelner Felder ist hilfreich

Die Fragebogenseite F3 enthält die folgenden Fragen:

23. Die zuerst getestete Webseite ist einfacher zu verstehen als die zuletzt getestete
24. Die zuerst getestete Webseite ist einfacher zu benutzen als die zuletzt getestete
25. An der zuerst getesteten Webseite hat mir gut gefallen: [Platz für Freitext]
26. An der zuerst getesteten Webseite hat mir nicht gefallen: [Platz für Freitext]
27. An der zuletzt getesteten Webseite hat mir gut gefallen: [Platz für Freitext]
28. An der zuletzt getesteten Webseite hat mir nicht gefallen: [Platz für Freitext]
29. In Zukunft am ehesten verwenden würde ich die..., hierbei wurden auf der Skala „nicht zutreffend" durch „erste Webseite" und „zutreffend" durch „zweite Webseite" ersetzt

Die Fragebogenseite F4 enthält die folgenden Fragen:

30. Was mir positiv aufgefallen ist: [Platz für Freitext]
31. Was mir negativ aufgefallen ist: [Platz für Freitext]
32. Sonstige Anmerkungen: [Platz für Freitext]

Auf den Seite F1 bis F3 steht dem Tester am Ende der Seite jeweils ein Freitextfeld für Anmerkungen zur Verfügung.

Nicht für jede Variante werden alle Seiten des Fragebogens benötigt. Für die einzelnen Varianten werden die folgenden Fragebogenseiten angezeigt:

Variante 1 Erste Aufgabe: F1 - zweite Aufgabe: F2, F4

Variante 2 Erste Aufgabe: F1, F2 - zweite Aufgabe: F2, F4

Variante 3 Erste Aufgabe: F1 - zweite Aufgabe: F1, F2, F3, F4

Variante 4 Erste Aufgabe: F1 - zweite Aufgabe: F1, F3, F4

Variante 5 Erste Aufgabe: F1, F2 - zweite Aufgabe: F1, F3, F4

Variante 6 Erste Aufgabe: F1 - zweite Aufgabe: F1, F3, F4

Vor Beginn der Evaluation werden das Alter der Testperson und seine eigene Einschätzung seiner Computerkenntnisse (keine, gering, mittel, gut oder Experte) abgefragt. Mit Hilfe dieser Basisfragen können die späteren Ergebnisse besser eingeordnet werden.

5.1.3 Messwerte

Während die Testperson die Benutzungsoberfläche zur Lösung der gestellten Aufgabe verwendet, werden die folgenden Werte ermittelt und gespeichert:

- Zeitpunkt, zu dem mit der Aufgabe begonnen wurde
- Zeitpunkt, zu dem die Aufgabe abgeschlossen oder abgebrochen wurde
- Die erzeugte Anfrage
- Zeitpunkt, wann welche Dialogseite (neue UI), beziehungsweise welches Dialogfenster (alte UI), angezeigt wurde
- Zeitpunkt, Eingabefeld und Eingabewert zu jeder getätigten Eingabe
- Zeitpunkt, Dialogseite und Ursache der in der neuen UI angezeigten Fehler

Aus diesen Werten können unter anderem die folgenden Messwerte gewonnen werden:

a) Zeit vom Anzeigen der Benutzungsoberfläche bis zum Senden der Anfrage

b) Wie oft jede Dialogseite und jedes Dialogfenster angezeigt wurden

c) Wie viele Fehler hervorgehoben wurden oder, bei Version *N*, hervorgehoben worden wären

d) Ob und wann der Dialog durch den Benutzer abgebrochen wurde

Zusätzlich werden der User Agent des von der Testperson verwendeten Browsers und die bei der Testperson eingestellte Bildschirmauflösung ermittelt und gespeichert. So können später vom Browser oder der Auflösung abhängige Probleme leichter identifiziert werden.

5.2 Anpassungen der Benutzungsoberfläche

Für die Durchführung der Evaluation wurde die neu entwickelte Benutzungsoberfläche angepasst und die im Nachfolgenden als Evaluationsclient bezeichnete Variante der Benutzungsoberfläche erstellt. Das Grundgerüst der Benutzungsoberfläche wurde erweitert, so dass während der Benutzung jederzeit zwischen der neuen und der alten Benutzungsoberfläche gewechselt werden kann. Ebenfalls wurde ein Framework geschaffen, mit dem die zuvor beschriebenen Fragen und Aufgaben angezeigt werden können.

Der neu hinzugefügte *EvaluationManager* steuert dabei den Ablauf der Evaluation und das Speichern der Antworten des Benutzers sowie der Messwerte. Der Ablauf der Evaluation folgt dabei stets dem Muster Begrüßung, Basisfragen, erste Aufgabe mit anschließenden Fragen, zweite Aufgabe mit anschließenden Fragen, Abschlussseite und Möglichkeit, am Gewinnspiel teilzunehmen sowie Interesse an weiteren Benutzerstudien anzumelden. Die Aufgaben ändern sich dabei abhängig von der Variante (siehe Kapitel 5.1.1).

Bei beiden Benutzungsoberflächen wurden diverse Funktionen so erweitert, dass sie beim Aufrufen der Funktion ein Event auslösen, das relevante Messwerte transportiert und vom *EvaluationManager* aufgefangen wird. Ebenso wurden alle Textstellen ins Deutsche übersetzt und noch nicht funktionsfähige Bestandteile der Benutzungsoberflächen deaktiviert oder entfernt, damit sie die Evaluation nicht beeinflussen können. In der neuen Benutzungsoberfläche wurden zudem Funktionen integriert, mit denen die dynamischen Hervorhebungen ein- und ausgeschaltet werden können.

Da die Evaluation online durchgeführt wird, wurde der Evaluationsclient in der Google App Engine[1] installiert. Diese bietet zudem eine mit GWT einfach zu nutzende Speichermöglichkeit und ist in beschränktem Rahmen kostenfrei nutzbar. Da der Speicher der App Engine nicht tabellenbasiert ist und stattdessen JavaScript Objekte speichert, wurde ein zusätzliches Tool entwickelt, mit dem die Daten zur Auswertung aus dem Speicher heruntergeladen und in je einer XML-Datei pro Testperson gespeichert werden können.

Vor Beginn der Evaluation wurde der Evaluationsclient mehrfach auch von nicht an PESCaDO beteiligten Personen getestet, um möglichst viele Fehler erkennen und beheben zu können. Insbesondere wurde hierbei auch geprüft, ob die Texte und Aufgaben für nicht mit PESCaDO vertraute Personen verständlich sind.

5.3 Durchführung und Ergebnisse der Evaluation

Die Evaluation wurde über einen Zeitraum von 14 Tagen durchgeführt. In diesem Zeitraum haben 142 Personen die Evaluation begonnen, davon haben 56 Personen die Evaluation vollständig abgeschlossen.

In diesem Kapitel werden die Ergebnisse der 56 vollständigen Durchläufe der Evaluation vorgestellt, im anschließenden Kapitel werden die Ergebnisse ausgewertet.

5.3.1 Verteilung der Evaluationsvarianten

Die nachfolgenden Tabellen beschreiben, wie häufig die einzelnen Varianten der Evaluation (siehe Kapitel 5.1.1) ausgegeben wurden und wie oft die jeweiligen Aufgaben (Aufgabe 1 = A1, Aufgabe 2 = A2) von der Testperson abgebrochen wurden. Zu Tabelle 5.1 sei angemerkt, dass die Häufigkeiten aufgrund der beschriebenen Ausgabemethode der Varianten mit Hilfe eines Zählers (siehe Kapitel 5.1.1) gleichverteilt sein sollten. Die Unregelmäßigkeiten wurden vermutlich durch Latenzprobleme verursacht, haben auf die Ergebnisse der Evaluation jedoch keine Auswirkung.

[1] https://developers.google.com/appengine/

Variante	1	2	3	4	5	6	Gesamt
Häufigkeit	23	25	24	25	23	22	142

Tabelle 5.1: Gesamtverteilung der ausgegebenen Varianten

Variante	Häufigkeit	A1 abgebrochen	A2 abgebrochen	A1 & A2 abgebrochen
1	12	1	0	0
2	9	0	5	0
3	7	1	1	0
4	11	1	0	0
5	6	0	0	0
6	11	1	11	1

Tabelle 5.2: Verteilung der ausgegebenen Varianten und abgebrochene Aufgaben der 56 vollständigen Evaluationsdurchläufe

5.3.2 Basisfragen

Die beiden nachfolgenden Tabellen beschreiben die Altersverteilung und die vorhandenen Computerkenntnisse der Testpersonen.

Alter	Anzahl	Alter	Anzahl	Alter	Anzahl
17	1	25	2	33	0
18	1	26	2	34	1
19	4	27	1	35	1
20	4	28	4	...	...
21	3	29	5	52	1
22	7	30	1	53	0
23	6	31	3	54	1
24	6	32	2		
Gesamt					56

Tabelle 5.3: Altersverteilung der Teilnehmer

Wissen	Anzahl
kein	0
gering	3
mittel	7
gut	16
Experte	30
Gesamt	56

Tabelle 5.4: Computerkenntnisse der Teilnehmer

5.3.3 Fragebogen

Die nachfolgenden Tabellen enthalten die zusammengezählten Ergebnisse der Fragen der Evaluation. Die Ergebnisse sind aufgegliedert in alte Benutzungsoberfläche, neue Benutzungsoberfläche ohne Hervorhebungen und neue Benutzungsoberfläche (mit Hervorhebungen). Zusätzlich wird unterschieden, ob die jeweilige Benutzungsoberfläche für die erste oder zweite Aufgabe der Evaluation verwendet wurde. Bei der grafischen Darstellung der Ergebnisse wurden die Werte aus der zweiten Aufgabe grau gefärbt, die Beschriftung *a.b* gibt an, dass es sich um die Frage *a* aus der *b*-ten Aufgabe handelt. Der Median der Boxplot-Diagramme wurde zur besseren Erkennbarkeit rot gefärbt. In den Tabellen steht der erste Wert *X* jeder Zelle *X/Y* für das Ergebnis aus der ersten Aufgabe, der zweite Wert *Y* für das Ergebnis aus der zweiten Aufgabe. Die jeweiligen Fragen können in Kapitel 5.1.2 betrachtet werden.

Allgemein gilt, dass eine höhere Punktzahl bei einer Frage ein besseres Abschneiden der jeweiligen Oberfläche bei dieser Frage bedeutet als eine niedrige Punktzahl. Ausgenommen hiervon sind die Fragen 16 und 17, bei denen eine niedrige Punktzahl für ein besseres Ergebnis steht.

Die letzten beiden Tabellen 5.8 und 5.9 befassen sich mit den vergleichenden Fragen 23, 24 und 29. Da diese von der Reihenfolge der verwendeten Benutzungsoberflächen in der Evaluation abhängig sind, wurden die Werte normalisiert. Die erste Spalte mit Punkten entspricht dabei fünf Punkten für die alte und einem Punkt für die neue Benutzungsoberfläche, bei der letzten Spalte mit Punkten ist es umgekehrt.

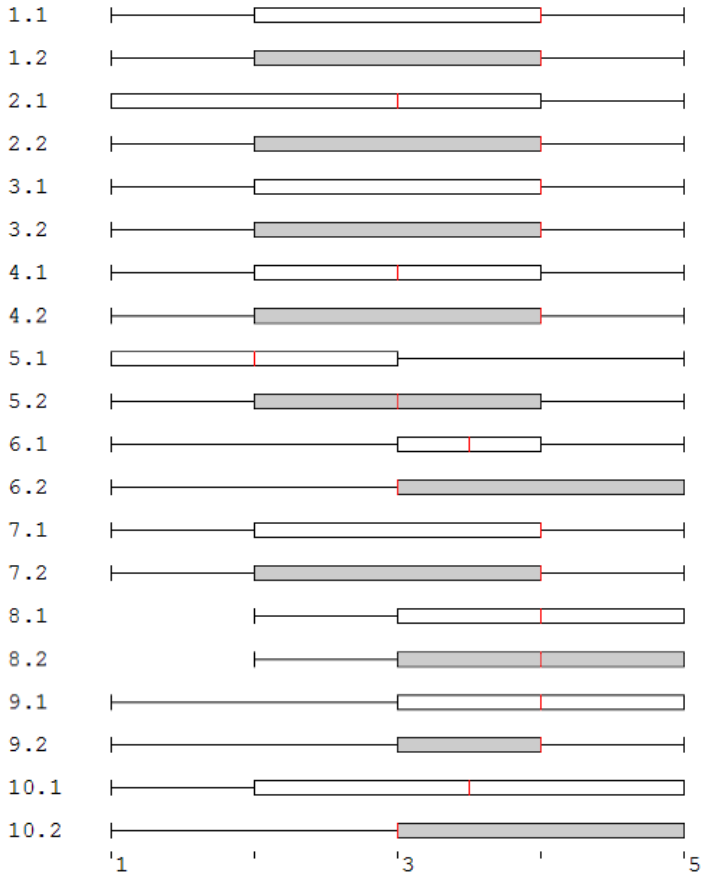


Abbildung 5.1: Punkteverteilung der Fragen 1 bis 10 für die alte Benutzungsoberfläche

Alte Benutzungsoberfläche						
Frage	1 Punkt	2 Punkte	3 Punkte	4 Punkte	5 Punkte	Gesamt
1	3 / 2	5 / 3	1 / 4	6 / 5	3 / 2	18 / 17
2	5 / 1	4 / 4	2 / 2	5 / 7	2 / 3	18 / 17
3	4 / 2	1 / 3	3 / 4	7 / 5	3 / 3	18 / 17
4	3 / 0	4 / 1	5 / 5	4 / 7	2 / 4	18 / 17
5	5 / 2	6 / 4	5 / 5	1 / 3	1 / 2	18 / 16
6	3 / 2	0 / 2	7 / 6	5 / 1	3 / 6	18 / 17
7	4 / 2	2 / 3	3 / 4	5 / 6	4 / 2	18 / 17
8	0 / 0	4 / 1	1 / 6	5 / 5	7 / 5	17 / 17
9	1 / 1	2 / 1	5 / 5	4 / 6	6 / 4	18 / 17
10	1 / 1	5 / 3	4 / 6	2 / 2	6 / 4	18 / 16

Tabelle 5.5: Punkteverteilung der Fragen 1 bis 10 für die alte Benutzungsoberfläche

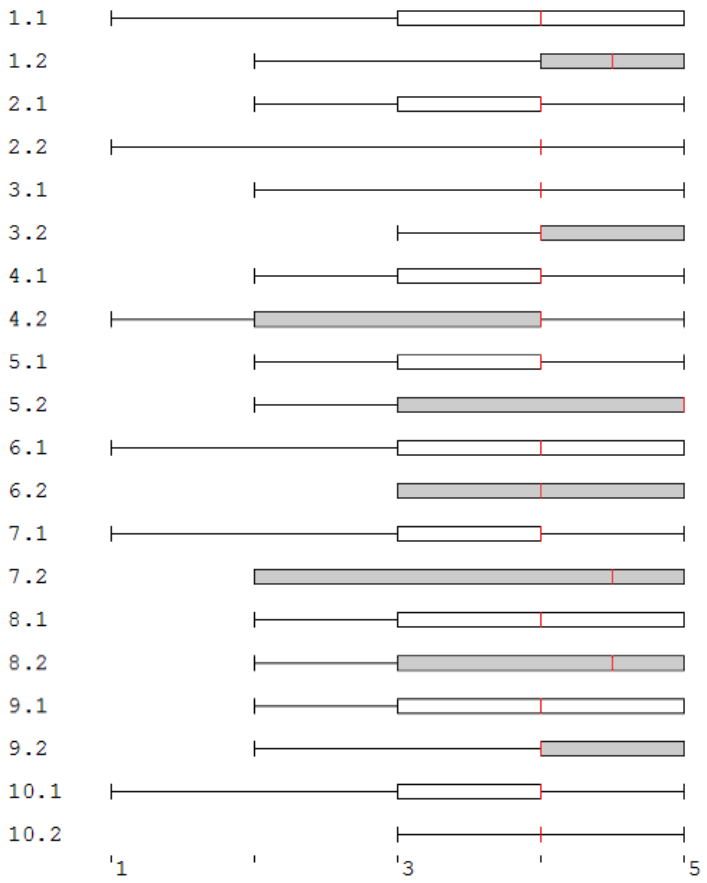


Abbildung 5.2: Punkteverteilung der Fragen 1 bis 10 für die neue Benutzungsoberfläche ohne Hervorhebung

Neue Benutzungsoberfläche (ohne Hervorhebung)						
Frage	1 Punkt	2 Punkte	3 Punkte	4 Punkte	5 Punkte	Gesamt
1	1 / 0	3 / 1	6 / 1	7 / 4	6 / 4	23 / 10
2	0 / 1	2 / 0	4 / 1	12 / 6	5 / 2	23 / 10
3	0 / 0	3 / 0	2 / 1	13 / 6	5 / 3	23 / 10
4	0 / 2	2 / 2	7 / 1	10 / 3	4 / 2	23 / 10
5	0 / 0	3 / 1	4 / 2	10 / 2	5 / 5	22 / 10
6	2 / 0	2 / 0	4 / 5	8 / 2	7 / 3	23 / 10
7	2 / 0	1 / 4	3 / 0	12 / 2	5 / 4	23 / 10
8	0 / 0	3 / 2	4 / 1	9 / 2	7 / 3	23 / 8
9	0 / 0	1 / 1	6 / 1	9 / 5	7 / 3	23 / 10
10	1 / 0	4 / 0	2 / 1	11 / 7	3 / 2	21 / 10

Tabelle 5.6: Punkteverteilung der Fragen 1 bis 10 für die neue Benutzungsoberfläche ohne Hervorhebungen

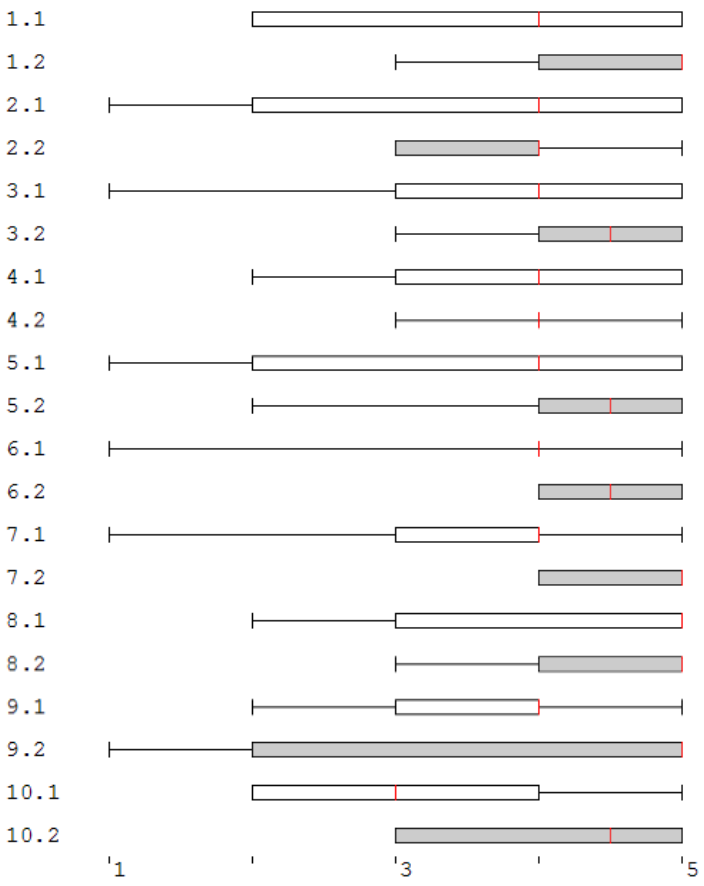


Abbildung 5.3: Punkteverteilung der Fragen 1 bis 10 für die neue Benutzungsoberfläche

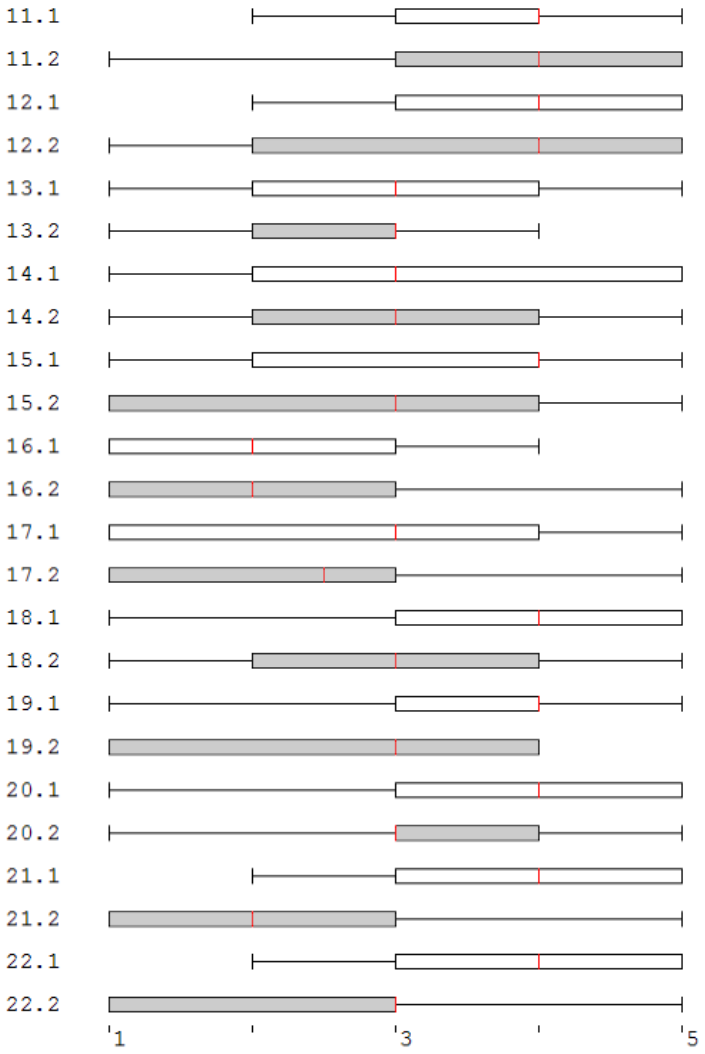


Abbildung 5.4: Punkteverteilung der Fragen 11 bis 22 für die neue Benutzungsoberfläche

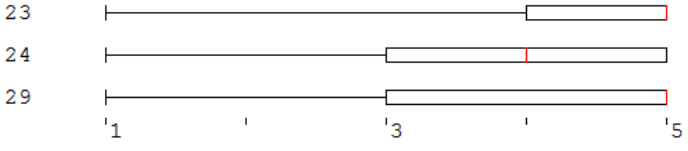


Abbildung 5.5: Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche ohne Hervorhebungen

Neue Benutzungsoberfläche						
Frage	1 Punkt	2 Punkte	3 Punkte	4 Punkte	5 Punkte	Gesamt
1	0 / 0	4 / 0	4 / 1	3 / 2	4 / 3	15 / 6
2	1 / 0	5 / 0	2 / 3	3 / 2	4 / 1	15 / 6
3	1 / 0	1 / 0	2 / 1	7 / 3	4 / 2	15 / 6
4	0 / 0	2 / 0	5 / 1	4 / 4	4 / 1	15 / 6
5	1 / 0	3 / 1	4 / 0	3 / 3	4 / 2	15 / 6
6	1 / 0	1 / 0	1 / 0	9 / 4	3 / 2	15 / 6
7	1 / 0	1 / 0	5 / 0	5 / 2	3 / 4	15 / 6
8	0 / 0	1 / 0	3 / 1	4 / 1	7 / 4	15 / 6
9	0 / 1	3 / 0	4 / 2	3 / 0	3 / 3	13 / 6
10	0 / 0	4 / 0	5 / 2	4 / 2	2 / 2	15 / 6
11	0 / 4	2 / 2	5 / 4	5 / 7	3 / 7	15 / 24
12	0 / 3	2 / 3	3 / 4	4 / 7	6 / 6	15 / 23
13	1 / 6	4 / 5	5 / 8	2 / 5	3 / 0	15 / 24
14	1 / 5	3 / 3	5 / 10	1 / 4	5 / 2	15 / 24
15	1 / 6	3 / 6	3 / 3	5 / 6	3 / 2	15 / 23
16	6 / 8	4 / 8	4 / 5	1 / 1	0 / 2	15 / 24
17	4 / 9	2 / 4	4 / 7	4 / 3	1 / 1	15 / 24
18	1 / 5	2 / 2	3 / 6	3 / 8	6 / 2	15 / 23
19	2 / 5	1 / 3	4 / 5	4 / 8	3 / 1	14 / 22
20	2 / 4	1 / 1	5 / 9	3 / 6	4 / 2	15 / 22
21	0 / 8	3 / 6	3 / 7	5 / 2	4 / 1	15 / 24
22	0 / 6	2 / 3	2 / 9	5 / 4	6 / 1	15 / 23

Tabelle 5.7: Punkteverteilung der Fragen 1 bis 22 für die neue Benutzungsoberfläche

Vergleich: Alt gegen Neu (ohne HV)						
Frage	Alt				Neu	Gesamt
23	2	3	0	4	13	22
24	2	2	2	6	9	21
29	4	1	1	4	11	21

Tabelle 5.8: Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche ohne Hervorhebungen

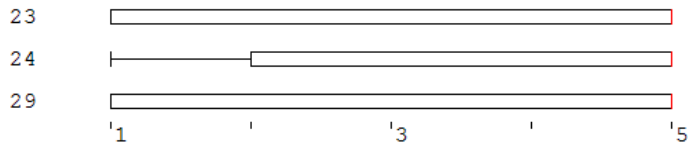


Abbildung 5.6: Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche

Vergleich: Alt gegen Neu						
Frage	Alt				Neu	Gesamt
23	4	0	0	0	8	12
24	3	2	1	1	6	13
29	4	1	2	0	6	13

Tabelle 5.9: Punkteverteilung der vergleichenden Fragen (normiert), alte gegen neue Benutzungsoberfläche

5.3.4 Für Anfrage benötigte Zeit

Die nachfolgenden Tabellen beschreiben, wie häufig mit welcher Benutzungsoberfläche innerhalb welchen Zeitraums eine Anfrage erstellt wurde. Die Zeitangabe ist im Format *Minuten:Sekunden*, die obere Zeitgrenze jeder Zeile ist exklusiv. Die Häufigkeit der Anfrage wird im Format *X/Y* dargestellt, wobei *X* die Anzahl der korrekten und *Y* die Anzahl der nicht der Aufgabenstellung entsprechenden Anfragen ist. Als nicht der Aufgabenstellung entsprechend werden dabei Anfragen betrachtet, die deutlich von der Aufgabenstellung abweichen und zum Beispiel „Arnold Admin" statt „Fiona Fit" verwenden. Anfragen, die von der für die Aufgabe gedachten Anfrage abweichen, weil die Formulierung der Aufgabenstellung nicht eindeutig war, werden als korrekt anerkannt.

Wird zusätzlich eine Zahl *(Z)* in Klammern angegeben, so ist dies die Anzahl der abgebrochenen Anfragen. Die abgebrochenen Anfragen wurden nicht mit in die grafische Darstellungen aufgenommen, um eine klare Übersicht der zum Erstellen einer Anfrage benötigten Zeiten zu gewähren.

Anfragen der ersten Aufgabe			
Zeit	Alt	Neu (ohne HV)	Neu
0:00 - 0:30	0 / 0	0 / 0	0 / 0
0:30 - 1:00	0 / 0	0 / 0	0 / 0
1:00 - 1:30	0 / 0	0 / 0	0 / 0
1:30 - 2:00	4 / 0	1 / 0	1 / 0
2:00 - 2:30	0 / 3	5 / 0	0 / 1
2:30 - 3:00	4 / 0	2 / 0	3 / 1
3:00 - 3:30	2 / 1	3 / 0	1 / 1
3:30 - 4:00	0 / 0	4 / 0 (1)	2 / 1
4:00 - 4:30	1 / 1	2 / 0	0 / 0
4:30 - 5:00	0 / 0	1 / 0	1 / 1
5:00 - 5:30	0 / 0	1 / 0	0 / 0
5:30 - 6:00	0 / 0	0 / 0 (1)	0 / 0
6:00 - 6:30	0 / 0 (1)	0 / 0	0 / 0
...			
9:00 - 9:30	0 / 0 (1)	0 / 0	0 / 0
...			
10:30 - 11:00	0 / 0	0 / 1	0 / 0
...			
13:30 - 14:00	0 / 0	1 / 0	0 / 0

Tabelle 5.10: Für Anfragen benötigte Zeit bei der ersten Aufgabe

Anfragen der zweiten Aufgabe			
Zeit	Alt	Neu (ohne HV)	Neu
0:00 - 0:30	0 / 0	0 / 0	0 / 1
0:30 - 1:00	0 / 0	0 / 0	0 / 0 (2)
1:00 - 1:30	1 / 0	0 / 0	9 / 1
1:30 - 2:00	3 / 3	0 / 0	5 / 1
2:00 - 2:30	2 / 1	0 / 0	4 / 0 (1)
2:30 - 3:00	4 / 1	0 / 0 (1)	0 / 0
3:00 - 3:30	1 / 0	0 / 0 (1)	0 / 0 (1)
3:30 - 4:00	1 / 0	0 / 0 (4)	1 / 0
4:00 - 4:30	0 / 0	0 / 0 (1)	0 / 0
4:30 - 5:00	0 / 0	0 / 0 (1)	0 / 0
5:00 - 5:30	0 / 0	0 / 0	0 / 0
5:30 - 6:00	0 / 0	0 / 0 (1)	0 / 0
6:00 - 6:30	0 / 0	0 / 0	0 / 0 (2)
6:30 - 7:00	0 / 0	0 / 0	0 / 0
7:00 - 7:30	0 / 0	0 / 0 (1)	0 / 0
7:30 - 8:00	0 / 0	0 / 0 (1)	0 / 0

Tabelle 5.11: Für Anfragen benötigte Zeit bei der zweiten Aufgabe

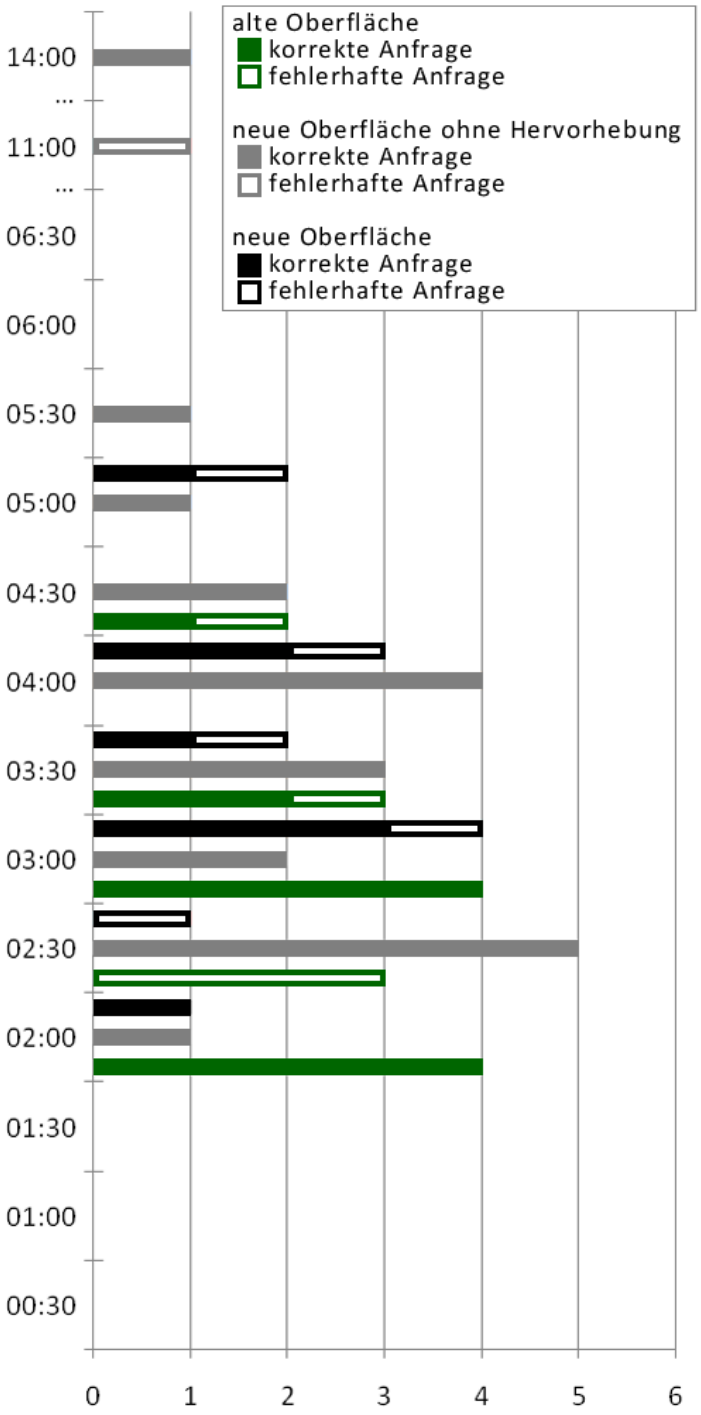


Abbildung 5.7: Für Anfragen benötigte Zeit bei der ersten Aufgabe. Auf der x-Achse wird die Anzahl der Anfragen, auf der y-Achse die jeweils exklusive obere Zeitgrenze der 30 Sekunden umfassenden Intervalle angezeigt.

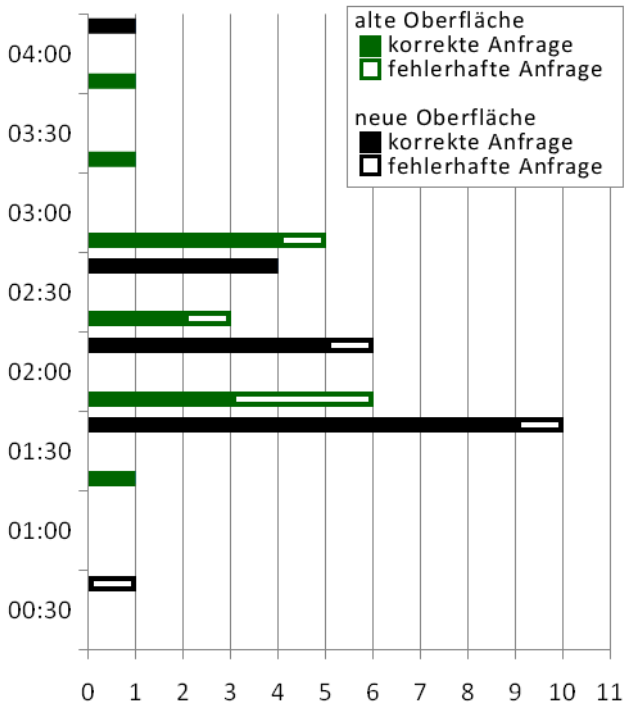


Abbildung 5.8: Für Anfragen benötigte Zeit bei der zweiten Aufgabe. Auf der x-Achse wird die Anzahl der Anfragen, auf der y-Achse die jeweils exklusive obere Zeitgrenze der 30 Sekunden umfassenden Intervalle angezeigt. Für die neue Benutzungsoberfläche ohne Hervorhebung liegen in diesem Fall keine Daten vor.

5.3.5 Häufigkeit der angezeigten Dialoge

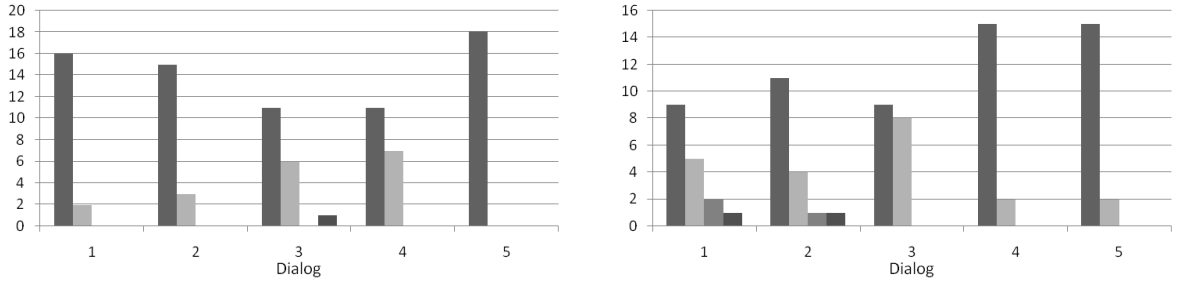
Die nachfolgenden Tabellen beschreiben, wie häufig eine bestimmte Dialogseite, beziehungsweise Dialogfenster in der alten Benutzungsoberfläche, angezeigt wurde. Hierbei wurde für jede Testperson separat gezählt, wie oft die einzelnen Seiten oder Fenster angezeigt wurden. Zusätzlich wird unterschieden, ob die Benutzungsoberfläche für die erste oder zweite Aufgabe der Evaluation verwendet wurde. Der erste Zahl in jeder Zelle *X/Y* gibt dabei den Wert für die erste Aufgabe an, die zweite Zahl den Wert für die zweite Aufgabe. Aus Tabelle 5.12 kann man zum Beispiel erkennen, das bei 16 Testpersonen in der ersten Aufgabe Dialog 1 nur einmal und bei zwei Testpersonen zweimal angezeigt wurde.

Dialog 1 bis Dialog 5 entspricht dabei der Reihenfolge der Dialogseiten in der neuen Benutzungsoberfläche, also Anfragetyp, Aktivität, Gebiet, Zeitraum und persönliche Details. Die Dialogfenster der alten Benutzungsoberfläche werden ebenfalls als Dialog 1 bis 5 bezeichnet. Hierbei ist zu beachten, dass die Reihenfolge der Dialogfenster in der alten Oberfläche nicht der Reihenfolge der neuen Oberfläche entspricht. Dialog 5 bezeichnet so beispielsweise in der alten und neuen Oberfläche den Dialog für persönliche Details, in der alten Oberfläche steht die zum Dialogfenster gehörende Schaltfläche jedoch an erster Position. Bei den Diagrammen wird auf der y-Achse die Häufigkeit einer Anzahl von Einblendungen angegeben. Auf der x-Achse wird für jeden der Dialoge 1 bis 5 mit mehreren Balken die Häufigkeit der Einblendungen des jeweiligen Dialogs dargestellt. Der erste Balken von links gibt dabei an, in wie viele Fällen der Dialog einmal angezeigt wurde, der zweite Balken in wie vielen Fällen zweimal, der dritte Balken in wie vielen Fällen dreimal, et cetera.

Die aus den Rohdaten extrahierten Werte weichen von den hier gezeigte Werten für den ersten Dialog bei der neuen Benutzungsoberfläche ab, da durch die Art und Weise wie die neue Oberfläche in den Evaluationsclient eingebunden ist, das Event zum anzeigen der ersten Seite bereits zweimal ausgelöst wird, bevor der Benutzer mit seinen Eingaben beginnen kann. Die hier dargestellten Werte für die Anzeigehäufigkeit des ersten Dialogs bei der neuen Benutzungsoberfläche (mit und ohne Hervorhebung) wurden daher jeweils um den Wert 1 reduziert.

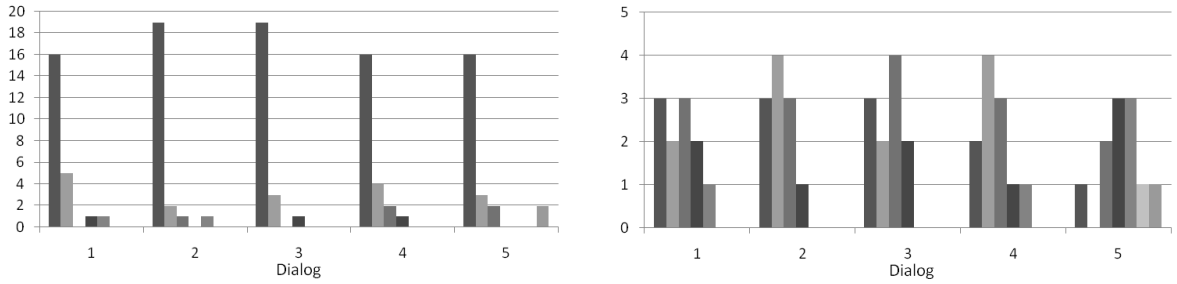
Alte Benutzungsoberfläche					
Häufigkeit	Dialog 1	Dialog 2	Dialog 3	Dialog 4	Dialog 5
1	16 / 9	15 / 11	11 / 9	11 / 15	18 / 15
2	2 / 5	3 / 4	6 / 8	7 / 2	0 / 2
3	0 / 2	0 / 1	0 / 0	0 / 0	0 / 0
4	0 / 1	0 / 1	1 / 0	0 / 0	0 / 0

Tabelle 5.12: Häufigkeit der Dialoge bei alter Benutzungsoberfläche



(a) Erste Aufgabe **(b)** Zweite Aufgabe

Abbildung 5.9: Häufigkeit der Dialoge bei alter Benutzungsoberfläche für die erste Aufgabe **(a)** und für die zweite Aufgabe **(b)**

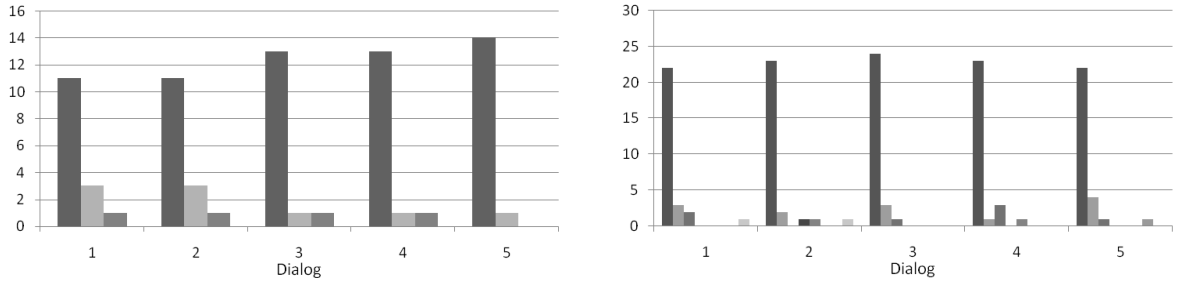


(a) Erste Aufgabe **(b)** Zweite Aufgabe

Abbildung 5.10: Häufigkeit der Dialoge bei neuer Benutzungsoberfläche ohne Hervorhebung für die erste Aufgabe **(a)** und für die zweite Aufgabe **(b)**

Neue Benutzungsoberfläche (ohne Hervorhebung)					
Häufigkeit	Dialog 1	Dialog 2	Dialog 3	Dialog 4	Dialog 5
1	16 / 3	19 / 3	19 / 3	16 / 2	16 / 1
2	5 / 2	2 / 4	3 / 2	4 / 4	3 / 0
3	0 / 3	1 / 3	0 / 4	2 / 3	2 / 2
4	1 / 2	0 / 1	1 / 2	1 / 1	0 / 3
5	1 / 1	1 / 0	0 / 0	0 / 1	0 / 3
6	0 / 0	0 / 0	0 / 0	0 / 0	0 / 1
7	0 / 0	0 / 0	0 / 0	0 / 0	2 / 1

Tabelle 5.13: Häufigkeit der Dialoge bei neuer Benutzungsoberfläche ohne Hervorhebungen



(a) Erste Aufgabe **(b)** Zweite Aufgabe

Abbildung 5.11: Häufigkeit der Dialoge bei neuer Benutzungsoberfläche für die erste Aufgabe **(a)** und für die zweite Aufgabe **(b)**

Neue Benutzungsoberfläche					
Häufigkeit	Dialog 1	Dialog 2	Dialog 3	Dialog 4	Dialog 5
1	11 / 22	11 / 23	13 / 24	13 / 23	14 / 22
2	3 / 3	3 / 2	1 / 3	1 / 1	1 / 4
3	1 / 2	1 / 0	1 / 1	1 / 3	0 / 1
4	0 / 0	0 / 1	0 / 0	0 / 0	0 / 0
5	0 / 0	0 / 1	0 / 0	0 / 1	0 / 0
6	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
7	0 / 0	0 / 0	0 / 0	0 / 0	0 / 1
8	0 / 1	0 / 1	0 / 0	0 / 0	0 / 0

Tabelle 5.14: Häufigkeit der Dialoge bei neuer Benutzungsoberfläche

5.3.6 Häufigkeit von hervorgehobenen Fehlern

Die nachfolgende Tabelle beschreibt, bei wie vielen Testpersonen welche Anzahl an Fehlern hervorgehoben wurde, beziehungsweise worden wäre (bei deaktivierter Hervorhebung). Diese Daten sind nur zur neuen Benutzungsoberfläche verfügbar, da die alte Benutzungsoberfläche keine Fehler in der Eingabe erkennt. Es wird zusätzlich die Art des Fehlers unterschieden. „Missing" steht für ein erforderliches, aber nicht ausgefülltes Feld, „Forbidden" ist ein verbotenes, aber ausgefülltes Feld.

Es ist zu beachten, dass die Zahl der Fehler nicht mit der Zahl falsch ausgefüllter Felder gleichgesetzt werden kann, da ein fehlerhaftes Feld mehrere Fehler verursachen kann und diese zudem mehrfach hervorgehoben werden können. Nicht enthalten sind die Fehlerzahlen aus der zweiten Aufgabe von Variante 2, da in diesem Fall die korrekte Befolgung der Aufgabe zu Fehlern führt. Diese Fehler können nicht von durch andere Ursachen verursachten Fehlern unterschieden werden und würden die Messwerte verfälschen.

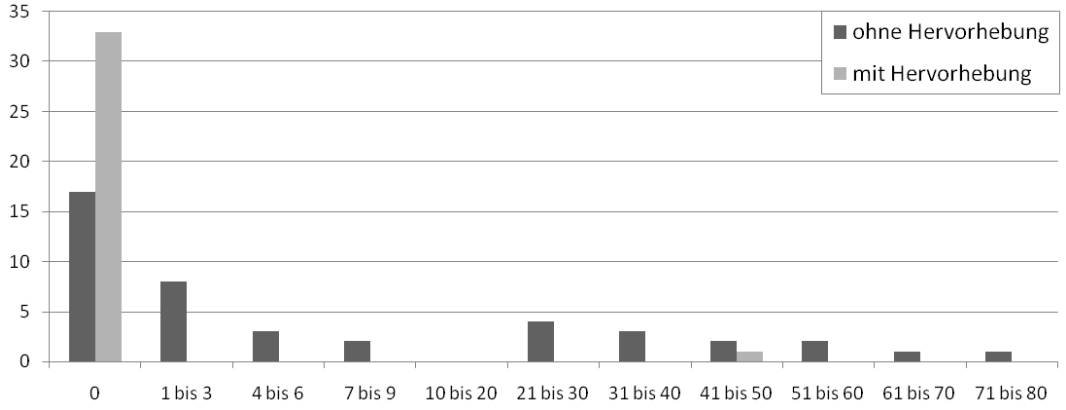


Abbildung 5.12: Häufigkeit der insgesamt hervorgehobenen Fehler bei der neuen Benutzungsoberfläche mit und ohne Hervorhebung. Die Zahl der hervorgehobenen Fehler wird dabei auf der x-Achse abgebildet, die Häufigkeit auf der y-Achse.

	ohne Hervorhebung			mit Hervorhebung		
Fehlerzahl	Forbidden	Missing	Gesamt	Forbidden	Missing	Gesamt
0	17		17	33		33
1 bis 3	3	5	8	0	0	0
4 bis 6	0	3	3	0	0	0
7 bis 9	2	0	2	0	0	0
10 bis 20	0	0	0	0	0	0
21 bis 30	4	0	4	0	0	0
31 bis 40	3	0	3	0	0	0
41 bis 50	2	0	2	0	1	1
51 bis 60	2	0	2	0	0	0
61 bis 70	1	0	1	0	0	0
71 bis 80	1	0	1	0	0	0

Tabelle 5.15: Anzahl von Benutzern der neuen Oberfläche mit und ohne Hervorhebung, bei denen eine bestimmte Fehlerzahl hervorgehoben wurde

5.3.7 Anmerkungen der Testpersonen

Auf den einzelnen Fragebogenseiten und bei den Fragen 25 bis 28 sowie 30 bis 32 wurden viele Anmerkungen und Kommentare der Testpersonen gesammelt, die Aufgrund des Umfangs nicht alle hier wiedergegeben werden. Im Nachfolgenden werden die häufigsten Anmerkungen und Kommentare der Testpersonen aus allen Varianten zusammengefasst, sortiert nach der Benutzungsoberfläche auf die sie sich beziehen. Bei jedem Kommentar ist in Klammern angemerkt, wie häufig dieser vorkam. Sofern es zu einem Thema gegensätzliche Aussagen gab, werden beide angegeben, unabhängig von ihrer Häufigkeit. Selbiges gilt für Kommentare, die auf allgemeine Verbesserungsmöglichkeiten hinweisen.

Allgemein

- (12) Die Methode zur Auswahl eines Gebiets ist nicht intuitiv. Viele Benutzer versuchen, bei gedrückter Maustaste ein Rechteck zu ziehen.
- (1) Bei der Auswahl des Zeitraums kann nur ein Datum, und keine Uhrzeit gewählt werden. Im Datumsfeld erscheint anschließend trotzdem eine Uhrzeit.
- (1) Auswahl mehrerer Aktivitäten wünschenswert.
- (1) Die Suche nach einer Region in gleicher Weise, wie nach einem Ort gesucht wird, ist wünschenswert.
- (1) Bei der Suche nach einem Ort wäre es wünschenswert, die Zoomstufe der Karte zu verringern, da es sonst bei großen Städten wie Paris sehr unübersichtlich ist.

Alte Benutzungsoberfläche
positiv:

- (3) Freie Auswahl der Schritte, insbesondere bei wiederholter Nutzung vereinfacht dies die
- (2) Dreiteiliger Aufbau der Seite aus anderen Anwendungen bekannt.
- (2) Anzeige der Anfragedetails.
- (2) Kurze, übersichtliche Anleitung. Arbeit.

negativ:

- (8) Die Anleitung ist zu kurz und kompliziert.
- (7) Struktur und Reihenfolge, in der die Dialoge bearbeitet werden sollen, ist unklar.
- (6) Die Ortssuche ist ungewöhnlich positioniert und wird leicht übersehen, zudem sollte der Anleitungstext im Feld verschwinden wenn der Benutzer eine Eingabe macht.
- (4) Die Icons besitzen keine Aussagekraft.
- (4) Unbrauchbare Angaben in den Anfragedetails, zum Beispiel Zeiten in GMT oder Koordinaten mit vielen Nachkommastellen.
- (4) Es ist nicht ersichtlich, warum bei manchen Eingabekombinationen der Dialog „Aktivität" keine Eingaben mehr ermöglicht.
- (3) Mehrere Benutzer fanden es schwierig, eine bestimmte Eingabe (meist „Luftqualität") in den Dialogfenstern zu finden und mussten sich so lange durch die Dialoge klicken, bis sie es gefunden hatten.
- (3) Mehreren Benutzern fehlte eine Möglichkeit, die Zoomstufe der Karte zu ändern.
- (2) Einige Benutzer haben die sich ändernden Anfragedetails auf der linken Seite mit dem Ergebnis verwechselt.
- (2) Es ist kein Zusammenhang zwischen den Eingaben erkennbar.
- (2) Die Schriftgröße der Texte ist zu klein.
- (2) Erfordert hohe Einarbeitungszeit und vieles wird nur durch Ausprobieren verständlich.
- (2) Bei Eingaben gibt es so gut wie kein Feedback.
- (2) Die Position der Anfragedetails ist ungewöhnlich, da Formulare im allgemeinen von links oben nach rechts unten abgearbeitet werden, und die Zusammenfassung am Ende stehen sollte.

Neue Benutzungsoberfläche mit Hervorhebungen
positiv:

- (10) Die Schritt-für-Schritt Anleitung des Benutzers durch den Dialog ist hilfreich.

- (7) Übersichtliche Darstellung und klare Reihenfolge, in der die Eingabefelder bearbeitet werden.
- (3) Verständliche Anleitung.
- (3) Wirkt modern.
- (2) Der aktuelle Fortschritt ist ersichtlich.
- (2) Zentrale Platzierung des Dialogs vor der Karte.
- (1) Eingabefehler können leicht erkannt werden.
- (1) Trotz der Schritt-für-Schritt Anleitung kann frei zwischen den einzelnen Dialogschritten gewählt werden.

negativ:

- (6) Der Unterschied zwischen „Administrativen Profilen" und „Personalisierten Profilen" ist unklar.
- (5) Die Texte zur Erklärung sind zu lang, Ausprobieren erscheint die schnellere Methode zu sein.
- (4) Der Text auf der Willkommensseite ist wenig aussagekräftig.
- (4) Schritte, bei denen keine Eingabe notwendig ist, sollten übersprungen werden.
- (3) Zentrale Platzierung des Dialogs verdeckt die Karte.
- (3) Die Ursache eines Fehlers ist nicht immer erkennbar.
- (3) Die erzwungene Reihenfolge der Eingaben stört.
- (2) Eingabefeld für Ortssuche ist zu unauffällig und manchmal schwierig zu finden.
- (2) Das Mausrad scrollt den Dialog und zoomt gleichzeitig die Karte.
- (2) Deaktivierte Eingabefelder sollten ausgeblendet werden.
- (2) Die Fehlermeldungen sind nicht aussagekräftig.
- (1) Es ist umständlich, dass durch eine Eingabe ungültig gewordene andere Eingaben von Hand gelöscht werden müssen.

Neue Benutzungsoberfläche ohne Hervorhebungen
Hierbei werden nur die Unterschiede zur neuen Benutzungsoberfläche mit Hervorhebung aufgelistet.

positiv:

- -

negativ:

- (12) Es ist nicht ersichtlich, warum die „senden" Schaltfläche nicht verfügbar ist, obwohl alle Felder ausgefüllt wurden.

- (3) Es gibt keine Hervorhebung des aktuellen Fortschritts.
- (2) Es ist unklar, welche Eingabe wann und wo gemacht werden kann oder nötig ist.

5.4 Auswertung und Erkenntnisse der Evaluation

In diesem Kapitel werden die zuvor vorgestellten Ergebnisse der Evaluation ausgewertet. Hierzu wird zunächst betrachtet, ob und in wie weit die in Kapitel 5.1 aufgestellten Thesen bestätigt werden können. Anschließend wird betrachtet, welche Ideen der in dieser Arbeit vorgestellten Lösung sich als brauchbar erwiesen haben und wie sie eventuell noch weiter entwickelt werden können.

5.4.1 Überprüfung der Thesen

These 1

> Die dynamische Hervorhebung einzelner Felder hilft dem Benutzer, alle erforderlichen Felder auszufüllen, fehlerhafte Eingaben zu vermeiden, fehlerhafte Eingaben zu erkennen und zu beheben und somit den Dialog schneller abzuschließen.

Zur Überprüfung dieser These werden zunächst die einzelnen Bestandteile der These betrachtet.

...alle erforderlichen Felder auszufüllen und fehlerhafte Eingaben zu vermeiden Betrachtet man hierzu die Tabellen 5.10 und 5.11, so fällt zum einen auf, dass insbesondere wenn der Benutzer bei der zweiten Aufgabe bereits etwas mit dem System vertraut war, die Anzahl nicht der Aufgabe entsprechender Anfragen im Vergleich zu den korrekten Anfragen bei der neuen Benutzungsoberfläche sehr gering ist, während sie bei der alten UI relativ konstant bleibt. Zum anderen fällt auf, dass es bei der neuen UI ohne Hervorhebung nur eine fehlerhafte Anfrage gab, sie dafür jedoch eine sehr hohe Zahl abgebrochener Anfragen aufweist. So haben zum Beispiel bei Variante 6 alle Benutzer die Aufgabe mit der neuen UI ohne Hervorhebung abgebrochen (siehe Tabelle 5.2). Aus den Anmerkungen dieser Benutzer ist zu erkennen, dass der Abbruch dabei stets erfolgte, weil die „senden" Schaltfläche infolge einer fehlerhaften Eingabe nicht verfügbar war.

Dies spiegelt sich auch in Tabelle 5.15 wieder. Bei insgesamt 34 ausgegebenen Aufgaben mit der neuen UI ohne Hervorhebung war die Hälfte aller Durchgänge ohne Fehler. Diese fehlerlosen Durchgänge traten zudem nur bei der ersten Aufgabe auf, die aufgrund der Aufgabenstellung das Ausfüllen aller Felder erfordert und einen fehlerfreien Durchlauf begünstigt. Die neue UI mit Hervorhebungen hat bei ebenfalls 34 ausgegeben Aufgaben 33 Durchläufe ohne einen Fehler und einen Durchlauf, bei dem erforderliche aber nicht ausgefüllte Felder zu Fehlern geführt haben.

Zieht man nun die Tabelle 5.7 hinzu und betrachtet hier die Fragen 11 bis 22, so decken sich die Ergebnisse der Messungen in den meisten Punkten mit den Einschätzungen der Testpersonen. Jedoch sind einige der in den Fragen besprochenen Punkte verbesserungswürdig, so ist insbesondere das Deaktivieren von Feldern für den Benutzer oft nicht nachvollziehbar. Auffällig hierbei ist die gute Bewertung des Deaktivierens (Fragen 21 und 22) in der ersten Aufgabe, während das Deaktivieren von Eingabefeldern in der zweiten Aufgabe wesentlich schlechter bewertet wurde. Erklären lässt sich dies mit der Aufgabenstellung in der zweiten Aufgabe, die dazu führt, dass mehr Felder als in der ersten Aufgabe deaktiviert werden und dies so vom Benutzer deutlicher wahrgenommen wird.

...fehlerhafte Eingaben zu erkennen und zu beheben Auch wenn die Benutzer die mit dem Erkennen und Beheben verknüpften Fragen 14 und 15 insgesamt eher neutral bewerten (siehe Tabelle 5.7), so zeigen doch insbesondere die Zahl der abgebrochenen Aufgaben (Tabelle 5.2) und die Anmerkungen der Testpersonen, dass dabei die Fehlerursache nicht erkannt werden konnte, die Wirksamkeit der Hervorhebungen.

...den Dialog schneller abzuschließen Betrachtet man hierzu Tabelle 5.10 und 5.11, so lassen sich bei der ersten Tabelle keine nennenswerte Unterschiede zwischen der neuen UI mit und ohne Hervorhebung feststellen. Bei der zweiten Tabelle lässt sich aufgrund fehlender Werte für die neue UI ohne Hervorhebung kein direkter Vergleich anstellen, da alle Aufgaben mit der neuen UI ohne Hervorhebung abgebrochen wurden.

Insgesamt kann man diese These als bestätigt ansehen.

These 2

> Die Struktur des Eingabedialogs hilft dem Benutzer dabei, alle erforderlichen Felder auszufüllen und zu erkennen, wie er mit dem Dialog fortfahren kann.

Da in der zum Vergleich der Dialogstruktur notwendigen alten UI keine Möglichkeit zum erkennen von Fehlern besteht, betrachtet man hierzu zunächst die Tabellen 5.2 und 5.10. Bei vergleichbarer Zahl abgebrochener Aufgaben 1 (je zwei mit der neuen und alten UI), ist die Zahl fehlerhafter Anfragen mit der alten UI deutlich höher als bei der neuen UI. Zieht man Tabelle 5.11 hinzu zeigt sich zudem, dass bei der alten UI die Zahl der fehlerhaften Anfragen im Verhältnis relativ gleich bleibt, während sie bei der neuen UI deutlich sinkt.

Aus den auf die Struktur bezogenen Fragen 1 bis 7 zeigt sich in den Tabellen 5.5 bis 5.7, dass die neue UI hier in allen Fragen bessere Werte erhalten hat als die alte UI. In den Tabellen 5.12 bis 5.14 zeigt sich jedoch keine deutlich erhöhte Zahl von Aufrufen der selben Dialogseiten, die auf wiederholte Änderungen der entsprechenden Eingabefelder hindeuten würden.

Betrachtet man zusätzlich die Anmerkungen der Benutzer, so war die mangelnde Struktur und daraus resultierende Orientierungslosigkeit ein häufiger Kritikpunkt an der alten UI. Die These kann daher als bestätigt angesehen werden.

These 3

> Die Hervorhebung von Abhängigkeiten zwischen den Feldern im Fehlerfall hilft dem Benutzer, die Fehlerursache zu erkennen und den Fehler zu beheben.

Hierauf gehen insbesondere die Fragen 19 und 20 ein. Die Tabelle 5.7 zeigt hierbei eine überwiegend neutrale bis gute Wahrnehmung durch den Benutzer. Auch die geringe Anzahl der abgebrochenen Aufgaben, bei denen diese Art der Hervorhebung aktiv war, (siehe Tabelle 5.2) spricht für die These. Es fällt jedoch auf, dass bei Variante 2 die Aufgabe 2, die durch die Aufgabenstellung für einen Fehler sorgt und so das finden und beheben eines Fehlers provozieren soll, in fünf der neun Durchläufe abgebrochen wurde. Aus den Anmerkungen dieser Durchläufe geht hervor, dass der Abbruch meist aufgrund der nicht aufzufindenden Fehlerursache gewählt wurde. In diesem Fall haben jedoch offenbar auch alle anderen Hervorhebungsmethoden versagt, der Effekt der hervorgehobenen Abhängigkeiten kann daher nicht isoliert betrachtet werden.

Zudem kam es bei der Beantwortung der Fragen 19 und 20 offenbar zu einem Missverständnis, wodurch das Ergebnis dieser Fragen verfälscht wurde. Aufgrund der ausgewerteten Fragen 19 und 20, die eine ähnlich hohe Beteiligung aufweisen wie die übrigen Fragen, müssten deutlich mehr Fehler aufgetreten sein, als in Tabelle 5.15 verzeichnet sind. Um Probleme bei der Aufzeichnung der Fehler auszuschließen, wurden die Eingaben der entsprechenden Varianten anhand der Aufzeichnungen über Benutzereingaben während der Evaluation wiederholt, wobei die niedrige Fehlerzahl bestätigt wurde. Es können daher nicht alle Benutzer, die bei den Fragen 19 und 20 eine Auswahl getroffen haben, auch die in den Fragen angesprochenen Verbindungslinien angezeigt bekommen haben. Eventuell wurden hier aufgrund der nicht präzise genug formulierten Frage andere „rote Linien" mit einbezogen. Die Testpersonen mit Variante 2 haben die Fragen 19 und 20 bei der zweiten Aufgabe im Durchschnitt neutral bewertet.

Über die Korrektheit dieser These kann daher keine Aussage getroffen werden.

These 4

> Die Beschreibungstexte helfen dem Benutzer zu entscheiden, wie er das Eingabefeld ausfüllen soll.

Zur Überprüfung dieser These werden die Fragen 9 und 10 bei der alten und neuen Benutzungsoberfläche betrachtet. Es zeigt sich, dass die neue Benutzungsoberfläche, insbesondere bei Frage 10, insgesamt besser bewertet wird als die alte Benutzungsoberfläche, wenngleich der Unterschied gering ausfällt. In beiden Fällen ist die Bewertung neutral bis gut. Aus den Anmerkungen der Testpersonen wird ebenfalls deutlich, dass die Beschreibungstexte der neuen Oberfläche hilfreich sind. Die These kann daher als bestätigt angesehen werden.

These 5

> Die neue Benutzungsoberfläche unterstützt den Benutzer beim Erstellen einer Anfrage besser als die alte Benutzungsoberfläche.

Zur Überprüfung dieser These werden die Ergebnisse der Evaluation der alten Benutzungsoberfläche mit der neuen Oberfläche (mit Hervorhebung) verglichen, sofern dies möglich ist. Bei den Fragen 1 bis 10 (siehe Tabellen 5.5 und 5.7) schneidet die neue Benutzungsoberfläche dabei bei allen Fragen ähnlich, meist jedoch besser als die alte Oberfläche ab. Vergleicht man die für das Erstellen einer Anfrage benötigte Zeit (Tabelle 5.10 und 5.11), so zeigen sich bei der ersten Aufgabe keine nennenswerten Unterschiede. Erst bei der zweiten Aufgabe ist die neue UI in wesentlich mehr Fällen schneller als die alte UI, deren Zeiten relativ konstant bleiben. Eine mögliche Erklärung hierfür wäre der Übungseffekt bei Benutzern, die bereits in der ersten Aufgabe mit der neuen Benutzungsoberfläche (mit oder ohne Hervorhebung) gearbeitet haben. Da sich jedoch die Zeiten von Testpersonen mit Variante 3, bei der zunächst die alte Oberfläche verwendet wird, ähnlich verteilen, wie die von Testpersonen mit anderen Varianten, kann der Übungseffekt hier vernachlässigt werden.

Betrachtet man die Häufigkeiten mit denen die einzelnen Dialogseiten und -fenster (Tabelle 5.12 und 5.14) angezeigt wurden, so schneidet die neue Benutzungsoberfläche leicht besser ab, insbesondere wenn man die Zahl der angezeigten Dialogseiten und -fenster bei der zweiten Aufgabe vergleicht. Bezieht man zudem die Fragen 23, 24 und 29 mit ein (Tabelle 5.8 und 5.9), in denen die neue und die alte Benutzungsoberfläche direkt miteinander verglichen werden, so sprechen die Ergebnisse deutlich für die neue Benutzungsoberfläche. Die These kann daher als bestätigt angesehen werden.

5.4.2 Signifikanz des Ergebnisses

Um die Signifikanz des Ergebnisses der Evaluation zu berechnen, werden die Fragen 1 bis 24 und 29 für die neue Benutzungsoberfläche betrachtet. Aufgrund der Formulierung der Fragen werden die aufgestellten Hypothesen erfüllt, sofern den Fragen nicht „nicht zugestimmt" wird, also zumindest eine neutrale Position (3 Punkte) erreicht wird. Ausgenommen hiervon sind die Fragen 16 und 17, bei denen es genau umgekehrt ist. Da sich die Fragen 23, 24 und 29 auf die Reihenfolge der Benutzeroberflächen beziehen, wurden ihre Ergebnisse entsprechend normiert. Für die Berechnung der Signifikanz wurden die Ergebnisse der Fragen aus der ersten und zweiten Aufgabe addiert.

Die entsprechende Nullhypothese H_o ist folglich erfüllt, wenn den Fragen nicht zugestimmt wird. Mit Hilfe des t-Tests [BD06] kann die Signifikanz der Hypothesen berechnet werden. Da es sich um gerichtete Thesen handelt ist ein einseitiger t-Test ausreichend. Aufgrund der bei den Fragen verwendeten Likert-Skala wird der Median zur Mittelwertbestimmung verwendet.

Die Nullhypothese ist in diesem Fall H_o: $\mu < \mu_o$, die Hypothese H_1: $\mu >= \mu_o$ mit $\mu_o = 3$. Die Nullhypothese H_o wird folglich abgelehnt, wenn t größer als das 95%-Quantil der t-Verteilung t_o ist. t_o hat für eine Stichprobengröße von n=18 und α=0,05 den Wert t_o=1,734 [BS05]. Da t_o für größere n kleiner wird, genügt dieser eine t_o-Wert. In der nachfolgenden Tabelle 5.16 sind für die Fragen 1 bis 24 und 29 die jeweils berechneten t-Werte (auf drei Nachkommastellen gerundet) angegeben. Man kann erkennen, dass alle Fragen das Signifikanzkriterium für α=0,05 erfüllen. Die Ergebnisse der Evaluation sind dementsprechend signifikant.

Frage	Anzahl beantwortet	Median	t-Wert
1	21	4	7,746
2	21	3	3,515
3	21	4	8,739
4	21	4	9,403
5	21	4	6,928
6	21	4	9,403
7	21	4	9,403
8	21	5	11,225
9	19	3	3,172
10	21	3	4,019
11	39	4	9,137
12	38	4	9,231
13	39	3	5,144
14	39	3	4,850
15	38	3	4,547
16	39	4	10,999
17	39	3	4,506
18	38	3	4,450
19	36	3	4,583
20	37	3	4,751
21	39	3	4,668
22	38	3	4,612
23	34	3	3,124
24	34	3	3,511
29	34	3	3,243

Tabelle 5.16: t-Werte der Fragen 1 bis 24 und 29

5.4.3 Relevanz des Ergebnisses

Wie aus den Tabellen 5.3 und 5.4 erkennbar ist, konnten für die Evaluation überwiegend Personen im Alter von 20 bis 30 gewonnen werden, die zudem über ein im Durchschnitt sehr gutes Wissen im Umgang mit Computern verfügen. Die Aussagekraft dieser Evaluation ist daher auf einen recht kleinen Teil der Bevölkerung beschränkt, die Relevanz ist insgesamt als niedrig einzustufen.

5.4.4 Erkenntnisse der Evaluation

Die Evaluation hat gezeigt, dass die Ideen und Methoden der in dieser Arbeit vorgestellten Lösung auf dem richtigen Weg sind. Sie hat jedoch auch gezeigt, dass an diversen Stellen noch Raum für Verbesserungen ist. In diesem Kapitel werden die größten Problemstellen, die die Evaluation aufgezeigt hat, kurz beschrieben und mögliche Verbesserungen vorgeschlagen.

Auswahl eines Gebiets auf der Karte Die Auswahl eines Gebiets auf der Karte ist für viele Benutzer nicht intuitiv zu benutzen, da sie es von anderen Anwendungen gewohnt sind, bei gedrückter Maustaste ein Rechteck aufzuziehen oder eine Linie zu zeichnen.

Eine Möglichkeit diesem Problem zu begegnen, ist die zusätzliche Einführung einer Schaltfläche, mit der in gewohnter Weise ein solches Rechteck aufgezogen werden kann. Zudem werden in den Schaltflächen Grafiken angezeigt, die zum einen ein Rechteck zeigen, zum anderen ein unregelmäßiges Polygon. Ähnliche Grafiken werden zum Beispiel auf in Zeichenanwendungen verwendet.

Beschreibungstexte zu lang Bei den zu langen Beschreibungstexten muss geprüft werden, ob Teile davon, die sich nicht direkt auf die auf der selben Dialogseite vorhandenen Eingabefelder beziehen, entfernt oder auf eine separate Seite mit weiterführenden Erklärungen verschoben werden können. Auf diese Erklärungsseite kann anschließend verlinkt werden. Zudem bietet es sich an, vorhandene Eingabefelder oder andere Elemente der Webseite mit Grafiken zu referenzieren. Sofern das Element keine Grafik verwendet, kann auch über eine dem eigentlichen Element nachempfundene Abbildung darauf verwiesen werden, zum Beispiel bei den Schritten 1 bis 5 in der Dialogseitenübersicht.

Fehlermeldungen nicht aussagekräftig Um die Aussagekraft der Fehlermeldungen zu erhöhen, können diese mit Hilfe der bereits in den *DialogError* Klassen vorhandenen Informationen über verursachende und betroffene Eingabefelder individualisiert werden. Statt der Meldung „Ein Feld auf dieser Seite verursacht einen Konflikt mit einem anderen Feld" könnte dann zum Beispiel „Feld X verursacht einen Konflikt mit Feld Y auf Seite Z" ausgegeben werden.

Erzwungene Reihenfolge der Eingaben Einigen Benutzern war die durch die Abfolge der Dialogseiten festgelegte Reihenfolge zur Eingabe von Daten zu restriktiv. Zwar kann bereits jetzt jederzeit mit den Schaltflächen in der Dialogseitenübersicht frei zwischen den einzelnen Seiten gewechselt werden, jedoch scheint die Möglichkeit dafür nicht gut genug erkennbar zu sein. Abhilfe kann zum Beispiel geschaffen werden, indem die übrigen verwendeten Schaltflächen (insbesondere „zurück" und „weiter") von der Darstellung her an die Darstellung der Schaltflächen in der Dialogseitenübersicht angepasst werden.

Anpassung der Farben In einigen Fällen der Evaluation berichteten die Benutzer von fehlenden Hervorhebungen, obwohl sie eine Variante bearbeiteten, bei der die Hervorhebung aktiv war. Geht man davon aus, dass in diesem Fall kein Fehler vorlag und die Hervorhebung tatsächlich stattgefunden hat, müssen eventuell die zur Hervorhebung verwendeten Farben angepasst werden. Insbesondere die Pastelltöne können auf schlecht konfigurierten Bildschirmen, bei ungünstigem Blickwinkel oder ungünstiger Beleuchtung auf Flachbildschirmen nur schwer zu erkennen sein. Ebenso muss geprüft werden, wie gut diese Farben von Personen mit einer Farbsehschwäche erkannt werden können.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurden verschiedene Methoden zur intelligenten Dialogführung mit dem Benutzer untersucht und in einer Benutzungsoberfläche für PESCaDO implementiert. Mit der Benutzungsoberfläche kann eine neue Anfrage für PESCaDO generiert werden. Da PESCaDO eine Ontologie zur Beschreibung seines Datenmodells verwendet und in dieser auch der mögliche Aufbau korrekter Anfragen beschrieben ist, wurde zunächst eine Anwendung entwickelt, mit der eine Reihe von Regeln zur Beschreibung korrekter Anfragen aus der Ontologie generiert werden. Mit Hilfe dieser Regeln können dem Benutzer zum einen Eingaben vorgeschlagen werden, zum anderen können seine Eingaben überprüft und er auf eventuelle Fehler in der Eingabe hingewiesen werden. Hierzu werden den einzelnen Eingabefeldern Gewichte zugewiesen und diese anhand der Regeln und Eingaben des Benutzers verändert.

Wie sich im Verlauf der Diplomarbeit zeigte, ist der schwierigste Part hierbei, wie die gewonnenen Informationen mit dem Benutzer kommuniziert werden können. Hierfür wurden ebenfalls verschiedene Lösungen entwickelt und in der Benutzungsoberfläche zur Anwendung gebracht.

Mit der so entwickelten Benutzungsoberfläche werden alle in der Problemstellung (Kapitel 3.1) beschriebenen Teilprobleme erfolgreich gelöst:
Da der Benutzer seine Anfrage erst nach Ausfüllen aller erforderlichen Felder absenden kann, wird sichergestellt, dass keine unvollständigen Anfragen an den Server gesendet werden. Durch die Hervorhebung von Eingabefeldern wird der Benutzer zum einen auf fehlerhafte Eingaben hingewiesen, zum anderen werden ihm, insbesondere durch das Deaktivieren verbotener Eingaben, weitere sinnvolle Eingabefelder vorgeschlagen. Mit den hinzugefügten Bereichen der Benutzungsoberfläche zur Verwaltung seines Profils und zur Anzeige des Ergebnisses der Anfrage können dem Benutzer zudem gezielt Eingabewerte vorgeschlagen werden und die Anfrage, auch nachdem das Ergebnis bereits angezeigt wurde, noch weiter bearbeitet werden.

In einer anschließenden Benutzerstudie wurden zum einen die im Rahmen dieser Arbeit entwickelten Methoden überprüft, zum anderen die neu entwickelte Benutzungsoberfläche mit der bisher verwendeten alten Oberfläche verglichen. Die Benutzerstudie zeigte dabei, dass die neue Oberfläche die meisten Probleme der alten Oberfläche beheben konnte und auch die neu entwickelten Methoden großteils wie erwartet funktionierten. Es zeigte sich jedoch auch, dass es an verschiedenen Stellen noch Potential für weitere Verbesserungen der Methoden gibt.

Ähnlich wie die Methoden zur Interaktion mit dem Benutzer bietet auch das Grundgerüst, die Regeln und generierten Gewichte der Eingabefelder, noch Raum für Verbesserungen.

So können hier zum Beispiel mit Methoden zum maschinellen Lernen die Rückmeldungen des Benutzers, seien sie explizit durch das Klicken einer Schaltfläche oder implizit durch das Ändern und erneute Senden seiner Anfrage, verwendet werden, um die Gewichte der Eingabefelder weiter zu beeinflussen und somit die Benutzungsoberfläche besser an den Benutzer anzupassen.

In Zukunft werden derartige Benutzungsoberflächen noch weitere Verbreitung finden, da ein immer größerer Teil des alltäglichen Lebens im Umgang mit Computern verbracht wird und auch andere Alltagsgegenstände, wie zum Beispiel Fernseher, immer mehr über die Möglichkeiten eines Computers verfügen werden. Intelligente Benutzungsoberflächen sind dabei eine gute Möglichkeit, eine einfache Bedienbarkeit für den Benutzer zu erreichen. Durch die stetig steigende Vernetzung und Verknüpfung von Informationen wird es zudem möglich werden, die Benutzungsoberflächen immer besser an den Benutzer anpassen zu können.

Literaturverzeichnis

[AGA06] M. Armentano, D. Godoy, A. Amandi. Personal assistants: Direct manipulation vs. mixed initiative interfaces. *International Journal of Human-Computer Studies*, 64(1):27–35, 2006. doi:10.1016/j.ijhcs.2005.06.001. URL http://dx.doi.org/10.1016/j.ijhcs.2005.06.001. (Zitiert auf Seite 22)

[BD06] J. Bortz, N. Döring. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*. Springer Lehrbuch, 2006. (Zitiert auf Seite 86)

[BS91] H. Burkhardt, B. Smith. *Handbook of metaphysics and ontology*. Nummer Bd. 1 in Analytica (Philosophia Verlag). Philosophia Verlag, 1991. URL http://books.google.de/books?id=hYkYAAAAIAAJ. (Zitiert auf Seite 16)

[BS05] J. Bortz, C. Schuster. *Statistik für Human- und Sozialwissenschaftler*. Springer Lehrbuch, 2005. (Zitiert auf Seite 86)

[BTE11] H. Bosch, D. Thom, T. Ertl. Das Web als personalisierte Entscheidungsplatform – Die PESCaDO Idee. In *Lecture Notes in Informatics (LNI) – Proceedings of Informatik 2011: Informatik schafft Communities*. 2011. URL http://pescado-project.eu/results-topic3.php. (Zitiert auf Seite 11)

[DIN08] Ergonomie der Mensch-System-Interaktion - Teil 110: Grundsätze der Dialoggestaltung (ISO 9241-110:2006); Deutsche Fassung EN ISO 9241-110:2006, 2008. (Zitiert auf Seite 22)

[DM02] S. Delisle, B. Moulin. User interfaces and help systems: from helplessness to intelligent assistance. *Artificial Intelligence Review*, 18(2):117–157, 2002. doi:10.1023/A:1015179704819. URL http://dx.doi.org/10.1023/A:1015179704819. (Zitiert auf den Seiten 20 und 37)

[Dry97] D. C. Dryer. Wizards, guides, and beyond: rational and empirical methods for selecting optimal intelligent user interface agents. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, IUI '97, S. 265–268. ACM, New York, NY, USA, 1997. doi:10.1145/238218.238347. URL http://doi.acm.org/10.1145/238218.238347. (Zitiert auf Seite 20)

[FMO+01] M. Frank, M. Muslea, J. Oh, S. Minton, C. Knoblock. An intelligent user interface for mixed-initiative multi-source travel planning. In *Proceedings of the 6th international conference on Intelligent user interfaces*, IUI '01, S. 85–86. ACM, New York, NY, USA, 2001. doi:10.1145/359784.360287. URL http://doi.acm.org/10.1145/359784.360287. (Zitiert auf Seite 22)

[Höoo] K. Höök. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12(4):409 – 426, 2000. doi:10.1016/S0953-5438(99)00006-5. URL http://www.sciencedirect.com/science/article/pii/S0953543899000065. (Zitiert auf Seite 9)

[HC07] C.-C. Hsu, W.-M. Cheng. An Intelligent Transaction Assistant System for Electronic Commerce to Recommend Personalized Transaction Behavior. In *Machine Learning and Cybernetics, 2007 International Conference on*, Band 3, S. 1802 –1807. 2007. doi:10.1109/ICMLC.2007.4370440. (Zitiert auf Seite 22)

[HKRS08] P. Hitzler, M. Krötzsch, S. Rudolph, Y. Sure. *Semantic Web: Grundlagen*. Springer, Berlin, 2008. doi:10.1007/978-3-540-33994-6. (Zitiert auf Seite 16)

[Hub05] O. Huber. *Das psychologische Experiment: Eine Einführung*. Verlag Hand Huber, 2005. (Zitiert auf Seite 59)

[LCW09] C.-S. Lee, Y.-C. Chang, M.-H. Wang. Ontological recommendation multi-agent for Tainan City travel. *Expert Systems with Applications*, 36(3):6740–6753, 2009. doi:10.1016/j.eswa.2008.08.016. URL http://dx.doi.org/10.1016/j.eswa.2008.08.016. (Zitiert auf Seite 22)

[MHW] M. Myllynen, D. Hilbring, L. Wanner. PESCaDO - First Prototype Evaluation Report. (Zitiert auf den Seiten 34 und 35)

[OWL] Beschreibung der Web Ontology Language OWL. URL http://www.w3.org/TR/owl-semantics/. (Zitiert auf Seite 17)

[Ros12] M. Rospocher. PESCaDO Ontology Documentation. 2012. URL http://www.pescado-project.eu/Pages/Pdfs-pages/PESCaDO_Ontology_Documentation_2.0.pdf. (Zitiert auf Seite 17)

[RSW+10] M. Rospocher, L. Serafini, L. Wanner, H. Bosch, M. Myllynen, A. Karppinen. D5.1 Decision support request and problem specification language. 2010. URL http://www.pescado-project.eu/Pages/Pdfs-pages/D5.1.pdf. (Zitiert auf Seite 17)

[Sel94] T. Selker. COACH: a teaching agent that learns. *Communications of the ACM*, 37(7):92–99, 1994. doi:10.1145/176789.176799. URL http://doi.acm.org/10.1145/176789.176799. (Zitiert auf Seite 21)

[SV04] J. Stocq, J. Vanderdonckt. WOLD: a mixed-initiative wizard for producing multi-platform user interfaces. In *Proceedings of the 9th international conference on Intelligent user interfaces*, IUI '04, S. 331–333. ACM, New York, NY, USA, 2004. doi:10.1145/964442.964522. URL http://doi.acm.org/10.1145/964442.964522. (Zitiert auf Seite 21)

[Woo02] M. Wooldridge. Intelligent Agents: The Key Concepts. In *Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications II-Selected Revised Papers*, S. 3–43. Springer-Verlag, London, UK, UK, 2002. URL http://dl.acm.org/citation.cfm?id=645699.665762. (Zitiert auf Seite 22)

[WVT+11] L. Wanner, S. Vrochidis, S. Tonelli, H. B. J. Mossgraber, A. Karppinen, M. Myllynen, M. Rospocher, N. Bouayad-Agha, U. Bügel, G. Casamayor, T. Ertl, I. Kompatsiaris, T. Koskentalo, S. Mille, A. Moumtzidou, E. Pianta, H. Saggion, L. Serafini, V. Tarvainen. Building an Environmental Information System for Personalized Content Delivery. 2011. (Zitiert auf Seite 11)

[Yan09] S.-Y. Yang. Developing of an ontological interface agent with template-based linguistic processing technique for FAQ services. *Expert Systems with Applications*, 36(2):4049–4060, 2009. doi:10.1016/j.eswa.2008.03.011. URL http://dx.doi.org/10.1016/j.eswa.2008.03.011. (Zitiert auf Seite 22)

Alle URLs wurden zuletzt am 03.06.2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Geoffrey-Alexeij Heinze)