Design und Entwicklung eines Systems
zur (teil-)automatisierten
Generierung von Zielführungslinks für Webseiten

Masterarbeit

zur Erlangung des akademischen Grades
Master of Engineering „M. Eng.“

TM06/03/SS2008

Technische Fachhochschule Wildau

Fachbereich Ingenieurwesen/Wirtschaftsingenieurwesen

Studiengang Telematik

Tag der Abgabe:	8. September 2008
Betreuer:	Prof. Dr. Ing. Stefan Brunthaler
Themensteller:	Verkehrsverbund Berlin-Brandenburg GmbH, Burkhard Gerken

Bibliografische Beschreibung und Referat

Dassow, Stefan

Design und Entwicklung eines Systems zur (teil-)automatisierten Generierung von Zielführungslinks für Webseiten

Masterarbeit, Technische Fachhochschule Wildau 2008, xx Seiten, 22 Abbildungen, 2 Anlagen, 1 Beilage

Ziel:
(Teil-)Automatische Ermittlung von Adressdaten aus heterogenen Datenbeständen und Konvertierung dieser in zielgerichtete Hyperlinks zu Fahrplanauskunftssystemen.

Inhalt:

- Strukturierung von unbekannten Datensystemen
- Entwicklung eines Verfahrens zur Ermittlung von Adressdaten aus heterogenen Datenbeständen
- informationstechnische Implementierung einer Adressermittlung
- Konvertierung von Adressdaten in verschiedenste Formate
- Generierung von Hyperlinks aufgrund von Adressen

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Hohen Neuendorf, den 25. August 2008

_____
Stefan Dassow

Inhaltsverzeichnis

Stichwortverzeichnis

ER-Modell
: Entity-Relationship-Modell

Freeware
: Kostenlose Software

GIS
: Geoinformationssystem

GUI
: engl.: Graphical User Interface - Benutzeroberfläche

Hyperlink
: Verweis zu einem anderen WWW-Dokument - kurz: Link

n-Gramm
: Folge von n Zeichen (Spez. Monogramm: 1 Zeichen)

ODBC
: standardisierte Datenbankschnittstelle

ÖPNV
: öffentlicher Personennahverkehr

ÖV
: Öffentlicher Verkehr

Phonetik
: hier: klangliche Analyse von Text

RDBMS
: Relationales Datenbank Managementsystem

SOAP
: Netzwerkprotokoll auf Basis von XML

TCP
: Transmission Control Protokoll (Netzwerkprotokoll)

URL
: Uniform Ressource Locator (Internetadresse)

VBB
: Verkehrsverbund Berlin-Brandenburg

XML
: Extended Markup Language

Kapitel 1 - Einführung

Die vorliegende Masterarbeit beschäftigt sich mit der Aufbereitung von Adressen für die Nutzung im Internet. Das Internet ist ein riesiger Datenspeicher mit vielen geografischen Daten. Diese Daten werden in den unterschiedlichsten Formaten angegeben und verarbeitet.

Galerien mit verorteten Bildern, Routenplaner mit digitalen Karten oder Kataloge mit Anschriften für alle möglichen Anlässe sind nur drei von vielen Anwendungen, welche mit geografischen Daten arbeiten. Vor allem im Bereich der Routenplanung hat sich in den letzten Jahren viel entwickelt. Die Kartendaten von Routenplanern des motorisierten Individualverkehrs wurden zum Beispiel um Zusatzinformationen für Tankstellen oder Restaurants erweitert. Im öffentlichen Verkehr wurde das intermodale Routing etabliert, welches verschiedenste Verkehrsmittel miteinander kombinieren kann. Viele gehen aber immer noch davon aus, dass man mit einer Fahrplanauskunft nur von einer Haltestelle zu einer anderen kommt. Dies belegen auch die Zugriffsstatistiken des Verkehrsverbundes Berlin-Brandenburg (VBB). Nur knapp 20% der Fahrplanauskünfte vom August 2008 waren mit Adressen oder Umgebungskarten verbunden. Dabei weiß die Fahrplanauskunft viel besser, welche Haltestelle in der Nähe eines Ortes am Besten zu einer bestimmten Zeit zu erreichen ist. So kann sich die Gesamtzeit eines Weges stark reduzieren, wenn man vielleicht mal eine Haltestelle 100m weiter entfernt nutzen würde.

Viele der Betreiber von Internetportalen, die Informationen über den öffentlichen Nahverkehr bereithalten, bieten ihren Kunden häufig nur die nächste Haltestelle an. Aber wird diese Haltestelle überhaupt noch bedient oder ist sie in der Nacht auch noch zu erreichen? Ziel dieser Arbeit ist es, dass Betreibern von Adresskatalogen ein Werkzeug gegeben wird, um automatisiert optimale Informationen über den öffentlichen Verkehr (ÖV) über die Adressen zu veröffentlichen. Diese Informationen sollen in einem Hyperlink bereitgestellt werden. Dieser Link führt zu einer Internetseite, die eine Fahrplanauskunft zu dieser Adresse erzeugt. Dazu müssen die bereitgestellten Adressen aufbereitet und verifiziert werden, damit der Nutzer der Fahrplanauskunft wirklich an das richtige Ziel geführt wird.

Die Software, welche die Erzeugung der Hyperlinks vornimmt, richtet sich hauptsächlich an Nutzer, die eine Vielzahl von Adressen haben, denen eine Auskunft für den öffentlichen Nahverkehr (ÖPNV) hinzugefügt werden soll. Eine einzelne Adresse kann derzeit über schon vorhandene Dienste in solch einen Hyperlink überführt werden. Diese Prozedur beansprucht aber viel Zeit. Nutzer mit einem großen Datenbestand an Adressen halten diese in Datenbanken vor. Dabei gibt es unterschiedlichste Formate von Datenbanken. Dokumente von Tabellenkalkulationsprogrammen sind eine Art, diese Daten zu speichern. Gebräuchlich sind relationale Datenbanksystems (RDBMS) die über standardisierte SQL-Schnittstellen angesprochen werden können. Damit ein möglichst hoher Nutzerkreis von dem Programm profitiert, sind keine Vorgaben zum Format der Adressen zulässig. Damit bei der Verifizierung der Adressen keine Fehler aufgrund von nicht erkannten Adressformaten auftreten, sind die Adressen in ein Format zu überführen, das die Verifizierungseinheit versteht.

Der Nutzer der Software soll mit minimalem Wissen über den Aufbau seiner Datenbanken einen maximalen Erfolg bei der Erstellung der Hyperlinks erreichen. Dazu sollen die relevanten Informationen über den Speicherort der Adressen in der Datenbank automatisch ermittelt werden. Die eigentlichen Adressen aus der Datenbank sollen dann vollautomatisch für den Prozess der Link-Generierung in die Software importiert werden. Dafür sind Schnittstellen zu implementieren, die einen Datenimport ermöglichen. Damit die automatische Lokalisierung der Adressen auch zum Erfolg führt müssen bestehende Datenbanken und Adressformate analysiert werden, um die Suchalgorithmen an den Spezialfall der Adresssuche anzupassen.

Der VBB als Auftraggeber möchte mit den aus der Software erstellten Links erreichen, dass die Menschen in Berlin und Brandenburg besser über den sehr gut organisierten ÖV in der Region informiert werden. Studien haben gezeigt, dass in Berlin weniger als 40% der Haltestellen weiter als 500m von jeder Berliner Adresse entfernt sind [MID03, S. 146]. Diese und weitere Informationen aus den Fahrplanauskünften sollen die Attraktivität des ÖPNV herausstellen. Der Verkehrsverbund erhofft sich durch die zusätzlichen Informationen zum ÖV in Berlin und Brandenburg, dass mehr Menschen für den Umstieg auf Busse und Bahnen

begeistert werden können. Besonders in Zeiten des Klimawandels und hoher Ölpreise, wäre ein Erfolg der Software bei Betreibern von Adresskatalogen auch ein Erfolg für die Umwelt.

Kapitel 2 - Grundlagen

2.1 Organisation von Adressdaten

An der Verarbeitung von geografisch referenzierten Daten (Geodaten) in einem Geoinformationssystem (GIS) sind verschiedene Instanzen beteiligt. Ein System für die Verarbeitung von Geodaten besteht unter anderem aus Endnutzern und Regeln sowie Standards zur Kommunikation von verschiedenen GIS. Weiterhin sind Technologien enthalten, die die Erfassung, Verwaltung, Suche und Wiedergabe der Daten gewährleisten. Zur Kommunikation der einzelnen GIS werden entsprechende Netzwerke installiert. Die Daten werden dann über die Datenprovider mit Metadaten versehen und in den Datenbanken der GIS gespeichert. [NOGUERAS05, S. 3]

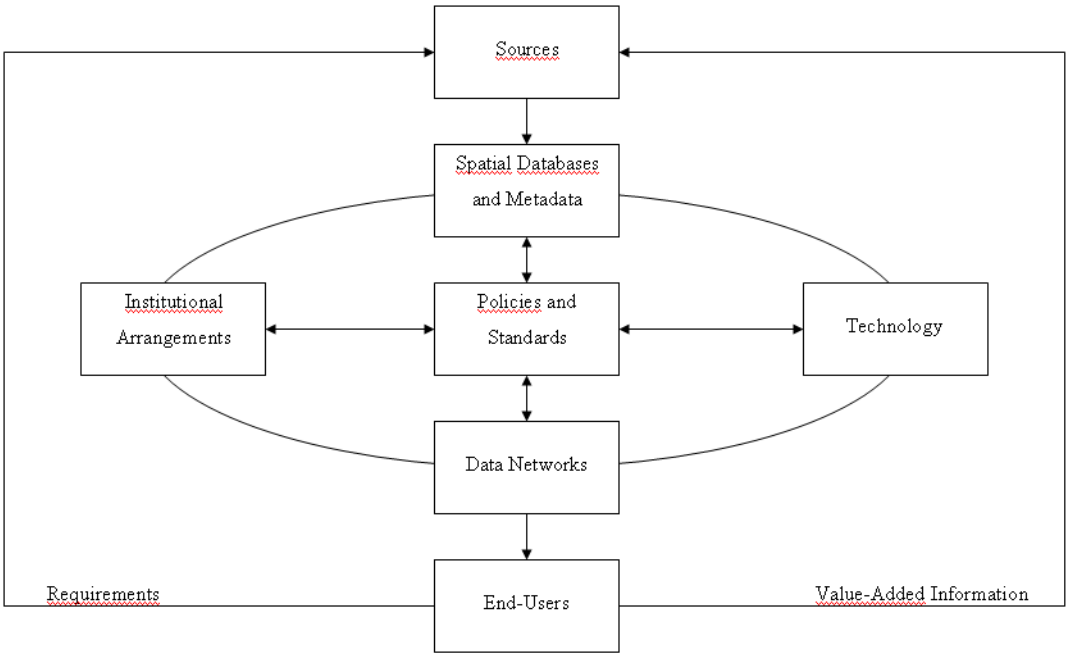


Abbildung 1: Schema eines GIS-Systems in Anlehnung an [NOGUERAS05, S. 4]

Bei Geodaten geht man grundsätzlich von Objekten aus, die einen räumlichen Bezug haben. Metadaten stellen den Bezug zwischen Objekteigenschaften und Raum dar. Der einfache Dublin Core Standard reicht zur Beschreibung nicht aus, da er nur grundlegende Metadaten wie zum Beispiel Autor und Thema kennt [NOGUERAS05, S. 7]. Ein Datenmodell, speziell zugeschnitten auf die Belange von Geodaten, findet

man in der ISO 19115. Der Standard beschreibt ausführlich die Geometrie eines Geoobjektes und dessen Beziehungen zu anderen Geoobjekten. Dabei können die Geoobjekte auch mit Metadaten versehen sein [ISO19115, S.109].

Im Gegensatz zu den geometrischen Informationen sind die Metadaten nicht genau spezifiziert, sondern nur in Themengruppen unterteilt. Eine der Themengruppen ist dabei der Ort. Diese Gruppe kann die Adresse des Objektes aufnehmen, aber auch Kontrollpunkte [ISO19115, S. 110]. Wie eine Adresse genau auszusehen hat, ist dabei nicht beschrieben. Auch in anderen Systemen zur Beschreibung von Geodaten finden sich keine Definitionen von Adressstrukturen.

Die Definitionen von Adressstrukturen findet man in Fachgebieten, die nichts mit Geoinformationssystemen zu tun haben; für Deutschland zum Beispiel in der Norm zu „Schreib- und Gestaltungsregeln für die Textverarbeitung“ [DIN5008]. Die Norm beschreibt, in welcher Reihenfolge die Adresselemente einer deutschen Anschrift im Anschriftfeld anzugeben sind. So sind in der Zeile für Straße und Hausnummer nach einem doppelten „/“ auch nähere Hinweise auf den Auslieferungsort möglich. Solche Hinweise können Etagenbezeichnungen oder Aufgänge sein. Die Postleitzahl wird immer gefolgt vom Ort. Der Ort darf aber keine Angaben zum Ortsteil enthalten. Dieser muss in einer Zeile oberhalb der Zustellangabe (Straße u. Hausnummer) angegeben werden.

```
Frau Erna Müller
Charlottenburg
Musterstraße 4 // 3. Etage
10178 Berlin
```

Text 1: Beispieladresse mit Zusatzangabe für Zustellung und Bezirk

Damit ist aber nur ein Layout der Adresselemente beschrieben. Es enthält keine Informationen über die Datenstrukturen und Datentypen der einzelnen Elemente.

2.1.1 Adressdaten in ISO 19773

Um den Missstand der fehlenden Adressspezifikation zu beheben, ist derzeit der ISO-Standard 19773 [ISO19773] in Arbeit. Die Vorlage zur Spezifikation enthält auch eine Beschreibung von Adresselementen wie sie den Vorgaben des Weltpostvereins entsprechen. Dabei wird eine Adresse in 4 Gruppen unterteilt:

1. Beschreibung des Absenders (Mailee Specification)
2. Beschreibung des Empfängers (Addressee Specification)
3. Beschreibung des Zustellortes (Delivery Point Specification)
4. Erweiterte Zustellinformationen (Mail Recipient Dispatching Information)

Die Daten der Adresse sind allein in der „Delivery Point Specification“ abgelegt. Die Adresse wird dabei über die Elemente „Locality“ und „Delivery Point Location“ näher definiert. Ersteres enthält den Ortsnamen und die Region (Bundesland) und Letzteres die Straße und die Hausnummer. Die Postleitzahl wird interessanterweise schon im Element „Delivery Point Specification“ definiert und könnte als Schlüssel für das „Locality“-Element verwendet werden.

Wie man der Beispielimplementierung in Abbildung 2 entnehmen kann, sind alle Felder als Zeichenkette definiert. Die zukünftige Norm eignet sich somit gut zum Speichern von Daten, da eine Struktur vorgegeben ist. Die internen Datentypen der einzelnen Elemente bleiben weiterhin unklar. Eine Validierung der Daten in solchen Datenkonstruktionen ist somit nicht möglich. Es bleibt dem Nutzer überlassen, wie er welche Daten in die Datenstruktur einpflegt. Ein Missbrauch von Feldern für Zusatzinformationen ist somit möglich.

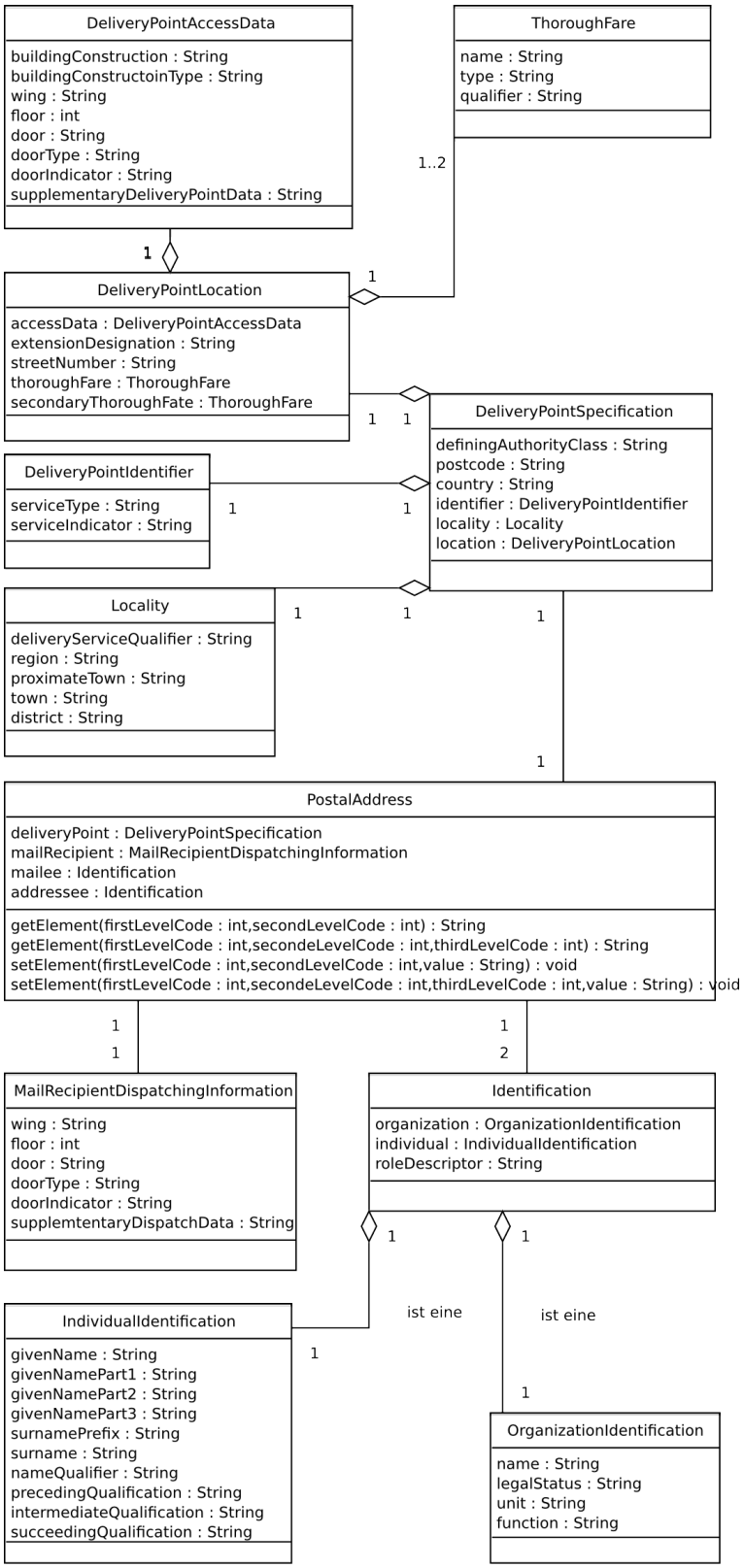


Abbildung 2: Beispielimplementierung der ISO 19773

2.1.2 XML Datenbanken und Relationale Datenbanksysteme

Sowohl XML-Datenbanken, als auch relationale Datenbanksysteme (RDBMS) eignen sich zur Speicherung von geografisch referenzierten Daten. Im Gegensatz zu relationalen Datenbanksystemen eignen sich Datenbanken auf Basis von XML auch als Medium zum Datenaustausch.

2.1.2.1 Relationale Datenbankmanagement Systeme

Grundsätzlich lassen sich in RDBMS Daten aller Art abspeichern. Einige Datenbanken wie zum Beispiel SQLite haben aber Einschränkungen, was die Funktionalität und die Datentypen betrifft.

Die Struktur einer Datenbank wird in einem sogenannten Entity-Relationship-Modell (ER-Modell) [BUCHMANN05, S. 32ff] dargestellt. RDBMS werden als Tabellensammlung abgebildet. Eine Tabelle repräsentiert dabei eine Relation. Jede Spalte der Tabelle repräsentiert ein Attribut der Relation. Eine Zeile (Entität) steht für ein reales Objekt. Die Relationen können durch Verknüpfungen in Verbindung zueinander gebracht werden. Die Verbindungen werden durch Schlüssel sichergestellt. Die unterschiedlichen Kardinalitäten (Anzahl der möglichen Beziehungen) müssen unter Umständen durch zusätzliche Tabellen sichergestellt werden.

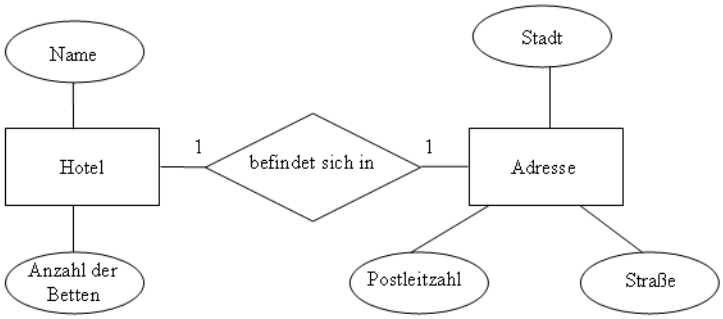


Abbildung 3: Einfaches ER-Modell mit einer 1:1 Beziehung zwischen Hotel und Adresse

Durch die Definitionen der Attribute lässt sich die Datenintegrität sicherstellen. Mit der „Data Manipulation Language“ [BUCHMANN05, S. 55ff], Teil des SQL Standards, lassen sich komplexe Abfragen erstellen, um Daten auftragsspezifisch abzufragen.

Mit Hilfe einer Rechteverwaltung lassen sich Zugriffe auf Teile von Datenbanken für jeden Benutzer begrenzen. Die Indexierung von Tabellen oder Attributen erlaubt einen schnellen Zugriff auf die gesuchten Entitäten.

2.1.2.2 XML Datenbanken

XML Datenbanken zeichnen sich durch ihre Flexibilität aus und sind nicht nur maschinenlesbar. XML Datenbanken sind hierarchisch organisiert. Alle Möglichkeiten des ER-Modells von RDBMS lassen sich auch in XML Datenbanken umsetzen. Die Integrität der Daten wird durch Schema-Dokumente oder DTD's überwacht. Diese Regeldokumente dokumentieren Datentypen, Referenzen und Reihenfolge der Elemente. Die Vorteile dieser Datenbankform liegen bei der Importmöglichkeit von mehreren Schemata und der Eingrenzung der Datentypen durch reguläre Ausdrücke. Mittels XPath und XQuery existieren mittlerweile einfache Arbeitsmittel zur Abfrage in XML-Dokumenten. Sie sind bei großen Datenbeständen vergleichsweise langsam. Mit XML lassen sich eigene Datentypen entwickeln. Weiterhin kann man heute mit einer ODBC-Schnittstelle auch mittels Datenbankprogrammen auf die Daten zugreifen. [SKULSCHUS04, S. 415]

2.1.2.3 Koexistenz

Sowohl RDBMS als auch XML-Datenbanken haben Vor- und Nachteile in der Datenverwaltung. Die Wandlung von XML in SQL und anders herum ist heute möglich. XML für den Datenaustausch ist weit verbreitet und RDBMS sorgen für die Datenspeicherung. Moderne RDBMS, die den SQL-Standard ISO/IEC 9075-14:2006 unterstützen, können XML sogar direkt verarbeiten.

2.2 Information Retrieval

Die Suche in unbekannten Datenbeständen wird unter dem Begriff „Information Retrieval“ zusammengefasst. Einer der größten unbekannten Datenbestände ist dabei das World Wide Web (WWW). Der Erfolg des Suchens ist bisher eher mäßig. Auch wenn Dienstleister wie Google® oder Yahoo!® für sich in Anspruch nehmen, alles zu finden, so ist die Frage der Genauigkeit der Suchergebnisse nicht dokumentiert. Grossmann [GROSSMANN04, S. XV] geht davon aus, dass heute bei einer Suchanfrage weniger als 50% der relevanten Dokumente gefunden werden.

Häufiges Problem ist, dass die Semantik der Texte von den Suchmaschinen nicht verstanden wird. Um eine Verbesserung des semantischen Verständnisses für Computer zu erreichen, müssen die Datenbeständen um Metainformationen erweitert werden, die den Inhalt der Dokumente maschinenlesbar beschreiben.

2.2.1 Retrieval-Strategien

Es gibt verschiedene Ansätze nach Inhalten zu suchen. In *Information Retrieval* [GROSSMANN04, S. 9] werden die Verfahren dabei nach ihren Suchstrategien unterschieden. Zum Beispiel sind das:

- Vektoranalyse
- Wahrscheinlichkeitsanalyse
- Wahrheitsanalyse

2.2.1.1 Vektoranalyse

Bei der Vektoranalyse werden Suchmaske und Dokumente in ihre Bestandteile zerlegt. Die Häufigkeit der einzelnen Bestandteile wird für jedes Dokument in einem Vektor abgebildet. Bei 1000 Begriffen in allen Dokumenten und Suchvektor ergibt das einen Vektor mit 1000 Elementen. Für jeden einzelnen Dokumentenvektor $D_i\left(d_{i1}, d_{i2}, \ldots, d_{it}\right)$ der Größe t wird das Skalarprodukt mit dem Suchmaskenvektor $Q\left(w_{q1}, w_{q2}, \ldots, w_{qt}\right)$ berechnet. Diese Werte lassen sich dann einfach mathematisch vergleichen. Das Produkt mit dem höchsten Wert hat die beste Übereinstimmung.

$$SC\left(Q, D_i\right) = \sum_{j=1}^{t} w_{qj} \times d_{ij}$$

Abbildung 4: Berechnung des Übereinstimmungsgrades (SC) beim Vektormodell [GROSSMANN04, S. 15]

Über invertierte Listen und Cosinus-Maß lassen sich die Datenmengen reduzieren und das Ergebnis verbessern.

2.2.1.2 Probabilistische Analyse

Bei der Ermittlung durch Wahrscheinlichkeiten werden folgende Kennzahlen verwendet:

- `t` – gesuchte Zeichenkette/Wort/Begriff
- `N` – Anzahl der Dokumente,
- `n` – Anzahl der Dokumente, die `t` enthalten,
- `R` – Anzahl relevante Dokumente für Suchanfrage,
- `r` – Anzahl relevante Dokumente, die `t` enthalten.

Zusätzlich kann noch die Häufigkeit/Anzahl an (gefundenen) Begriffen in einem Dokument eine Rolle spielen.

$$\omega = \log\left[\frac{\frac{r}{R}}{\frac{n}{N}}\right]$$

Der Quotient von relevanten Dokumenten zu der Gesamtheit der Dokumente ergibt dabei den Vergleichswert (ω). Die Formel kann durch Ergänzungen verfeinert werden [GROSSMANN04, S. 27]: a) Annahme, dass die Verteilung der Begriffe in den nicht relevanten Dokumenten unabhängig zur Suchmaske ist ($\omega_{2,}\omega_4$) und b) die Abwesenheit von Suchbestandteilen wird mit bewertet ($\omega_{3,}\omega_4$).

$$\omega_2 = \log\left[\frac{\frac{r}{R}}{\frac{(n-r)}{(N-R)}}\right]$$

$$\omega_3 = \log\left[\frac{\frac{r}{(R-r)}}{\frac{n}{(N-n)}}\right]$$

$$\omega_4 = \log\left[\frac{\frac{r}{(R-r)}}{\frac{(n-r)}{(N-n)-(R-r)}}\right]$$

Als Erweiterung von Wahrscheinlichkeitsmodellen lassen sich Folgenetzwerke für die Informationsrückgewinnung benutzen. Sprachanalysemodelle haben das probabilistische Modell häufig als Grundlage der Bewertung.

2.2.1.3 Extended Boolean Retrieval

Beim *Boolean Retrieval* lässt sich eine Sortierung der Ergebnisse nicht vornehmen, da ein Dokument entweder den Suchkriterien entspricht oder nicht. Die Suchkriterien können nur durch ein logisches `UND` bzw. `ODER` miteinander verknüpft werden. Durch die eingeschränkten Suchkriterien können (nur `ODER`-Verknüpfungen) sehr viele Dokumente in die Ergebnisliste aufgenommen werden, obwohl sie nicht relevant sind bzw. viele Dokumente unterschlagen werden (nur `UND`-Verknüpfungen) obwohl sie eine hohe Relevanz haben.

Um wenigstens eine gewisse Ordnung zu erzeugen, kann man beim *Extended Boolean Retrieval* [GROSSMANN04, S. 67f] jedem Suchkriterium zusätzlich eine prozentuale Gewichtung (ω) zuordnen. Zur Berechnung des Übereinstimmungsgrades kann man den euklidischen Abstand heranziehen. Für eine Suchmaske mit 2 Elementen kann man die euklidische Abstandsberechnung mit der 2-Norm durchführen. Dieser Wert wird dann über die Wurzel der Anzahl der Suchkriterien noch einmal normalisiert. Daraus folgt für zwei Suchbegriffe ein Wert von:

$$SC(Q,d_i)=\frac{\sqrt{(\omega_1)^2+(\omega_2)^2}}{\sqrt{2}}$$

2.2.2 Zeichenkettenverarbeitung

An vielen Stellen der Arbeit werden Zeichenketten verarbeitet; sei es bei der Suche nach Straßennamen oder dem Vergleich von Soll- und Ist-Daten.

Die Suche von Zeichenketten stellt dabei einen Spezialfall des Zeichenkettenvergleichs dar. Die Suche ist der zu 100% erfolgreiche Vergleich von zwei Zeichenketten, wobei die zu suchende Zeichenkette (Suchmaske) eine Teilmenge der zu durchsuchenden Zeichenkette (Text) abbildet. Die Suche gestaltet sich am Einfachsten bei statischen Zeichenketten. Hier kann durch einfache Schieberegister die betreffende Kette gefunden werden. Der Aufwand beträgt mit

dem Knuth-Morris-Pratt Algorithmus [LANG01] $O(m+n)$. Der lineare Aufwand lässt sich über andere Algorithmen wie Boyer-Moore [LANG02] im günstigsten Fall auf $O(m/n)$ reduzieren. Im schlechtesten Fall sind aber eine Vielzahl an Operationen im Vergleich zum Knuth-Morris-Pratt-Algorithmus erforderlich.

Beim Information Retrieval ist eine Suche nach regulären Ausdrücken anstatt nach einfachen Zeichenfolgen viel wahrscheinlicher. Sie geben ein Muster[WILKES01] vor, nach dem im Text gesucht wird. Der Aufwand, der sich durch eine solche Suchmaske ergibt, ist ungleich höher, als bei der einfachen Textsuche. Die Möglichkeiten der Suche sind dafür stark erweitert. So lassen sich durch logische und arithmetische Operatoren zum Beispiel E-Mail-Adressen finden, ohne eine bestimmte E-Mail Adresse angeben zu müssen. Mit regulären Ausdrücken werden also Zeichenketten gefunden, die sich aufgrund ihres Musters ähneln, aber nicht unbedingt identisch sind.

Findet man mehrere Zeichenketten die sich aufgrund eines Musters ähnlich sind, stellt sich die Frage, wie ähnlich sich diese Zeichenketten sind. Dazu müssen diese Zeichenketten miteinander verglichen werden. Um zwei Zeichenketten zu vergleichen, kann man entweder die Ähnlichkeit oder den Abstand berechnen. Kondrak kommt in [NAVARRO05, S.125f] zu dem Ergebnis, dass sich durch Abstandsberechnungen die Gleichheit genauer bestimmen lässt. Der schlechteste Ansatz ist die Berechnung der Hamming-Distanz [MEREI07], da sie nur auf gleich lange Zeichenketten angewandt werden kann und nur die Unterschiede der Zeichen des Textes mit dem gleichen Index berücksichtigt. Sulzberger beschreibt mit dem Levenshtein-Algorithmus einen Ansatz, der nicht nur Tauschoperationen, wie bei der Hamming-Distanz, kennt, sondern auch Einfüge- und Lösch-Operationen mit einbezieht [SULZBERGER08]. Somit können sowohl Zeichenketten unterschiedlicher Länge behandelt als auch bessere Ergebnisse erzielt werden. Die Verbesserungen ergeben sich durch das Erkennen von Verschiebungen aufgrund von Einfügeoperationen. Damerau hat das Verfahren noch um das Vertauschen von Zeichen [WILZ05, S.24] erweitert, um Tippfehler bei der Eingabe nicht zu hoch zu bewerten. Die Standardverfahren beziehen sich jeweils auf Monogramme, also den Vergleich von Einzelbuchstaben. In [NAVARRO05, S.124f] untersucht Kondrak das Verhalten der Algorithmen bei der Verwendung von n-Grammen. Er kommt dabei zu dem Schluss, dass man mit Bi-Grammen eine Verbesserung des Ergebnisses erreichen kann, bei Tri-Grammen, die Genauigkeit aber wieder abnimmt. Je größer

die Anzahl der Zeichen in den n-Grammen ist, die zum Vergleich herangezogen werden, desto größer werden auch die Aufwände für die einzelnen Verfahren. Für jedes einzelne Zeichen werden jetzt n Vergleiche notwendig. Alle diese Verfahren berücksichtigen aber nur den lexikalischen Vergleich der Texte. Die Phonetik und die Sprachfamilien (Landessprache) werden bei all diesen Verfahren nicht berücksichtigt.

Eine Phonetische Analyse von geschriebenen Texten ist grundsätzlich möglich. Dabei werden einzelnen Buchstabenkombinationen Zahlenwerte zugeordnet. Während mit SOUNDEX [WILZ05, S12ff] ein Verfahren existiert, welches für die englische Sprache ausreichend analysiert und weiterentwickelt ist, sieht es für die deutsche Sprache nicht so gut aus. Eine der wenigen Phonetik-Alphabete für die deutsche Sprache ist die „Köln Phonetik“ [WILZ05, S.17f]. Sie ist besonders in Ausschreibungen von Behörden weit verbreitet. Durch die begrenzte Anzahl an Regeln lässt sie sich relativ einfach in Software implementieren. Die Berücksichtigung des Einflusses der deutschen Vokale ist aber nicht in einem ausreichenden Maße gegeben. Die phonetische Analyse basiert auf der Ersetzung von Buchstabenkombinationen durch andere Zeichenkombinationen. Sie werden in einem oder mehreren Durchläufen zu einer endgültigen Zeichenkette zusammengefasst. Diese Zeichenkette muss mit den oben beschriebenen Verfahren aber wieder verglichen werden.

Für alle Verfahren gilt, dass die Distanzen noch einmal normalisiert werden müssen. Die häufigste Normalisierung findet über die größere Länge der beiden Texte statt.

Da bei dieser Arbeit nur deutsche Begriffe gesucht werden, ist eine Erkennung von Begriffen in unterschiedlichen Landessprachen nicht notwendig.

Die Verfahren berücksichtigen nicht den Kontext, in dem sie stehen. Da die zu suchenden Begriffe meist allein in ihrer Struktur stehen und die Länge der Suchmaske meist größer als die Hälfte der Textlänge ist, ist eine Betrachtung des Kontexts nicht notwendig.

Lösung für den Homepageservice

2.3 Analyse von Beispieldatenbanken

2.3.1 Datenstruktur

Zur Strukturanalyse von Adressdatenbanken sind 4 Beispiele vorhanden. Alle Beispiele setzen auf einem RDBMS auf. Dabei kommen die weit verbreiteten Datenbankensysteme von Oracle®, MySql® und PostgreSql® zum Einsatz. Alle diese Datenbanksysteme zeichnen sich dadurch aus, dass sie mindestens einen Großteil des SQL-92 Standard unterstützen.

Allen Datenbanken ist gemeinsam, dass der Datentyp über alle Adresselemente hinweg als Zeichenkette (VARCHAR) definiert ist.

Keine der Datenbanken erfüllt die Strukturanforderungen der „Dritten Normalform“, bei der „keine funktionalen Abhängigkeiten zwischen Nicht-Schlüsselfeldern existieren dürfen. Es dürfen keine transitiven (rückbezüglichen) Abhängigkeiten in einer Tabelle vorhanden sein. Alle Spalten dürfen also nur von dem Schlüsselattribut und nicht von anderen Werten abhängig sein.“ [BUCHMANN05, S. 49]. Keine der Datenbanken arbeitet mit zusammengesetzten Primärschlüsseln, wobei sich die Prüfung auf die „Zweite Normalform“ erübrigt. Die „Erste Normalform“, bei der alle Attribute nur einen Wert enthalten dürfen, wird nur dann eingehalten, wenn man annimmt, dass die Zustellangabe aus Straße und Hausnummer als ein zusammenhängender Wert angesehen wird. Nur eine der Datenbanken trennt die Angaben von Hausnummer und Straße in zwei Attribute auf. Eine andere Datenbank gliedert die Postleitzahlen und den Ortsnamen in eine eigene Tabelle aus. Diese Datenbank nutzt für die Beziehungen von Zustellangabe zu Ort aber keine Fremdschlüssel, da sie vom RDBMS nicht unterstützt werden.

Alle Datenbanken nutzen für die Attributbezeichnungen deutsche Begriffe, wobei keine deutschen Umlaute dafür verwendet werden. Die Attribute beschreiben den Inhalt hinreichend. So werden die Spalten für die Straße auch als solche („`strasse`“) bezeichnet. In dem Fall, in dem die Ortsnamen und Postleitzahlen in eine andere Tabelle ausgegliedert werden, wird dem Attributnamen („`id_stadt`“) ein „`id_`“

vorangestellt, um es als Fremdschlüssel zu kennzeichnen. Es wird sich auch an die Konvention gehalten, dass bei der Namensvergabe der Attribute immer Begriffe im Singular zu verwenden sind.

Das ER-Modell enthält in keinem Fall mehr als eine Adresse. Da aber bei keiner Datenbank das komplette ER-Modell verfügbar ist, ist nicht auszuschließen, dass in anderen Tabellen noch weitere Adressen abgebildet werden.

2.3.2 Datenbankinhalte

Bei zwei von den vier Beispieldatenbanken sind auch die Inhalte der Datenbanken für eine Analyse verfügbar. Dabei sind bei den einzelnen Adresselementen die unterschiedlichsten Formate aufgetreten. Folgende Analyse bezieht sich nur auf die zwei Datenbanken von denen auch die Daten mit verfügbar sind. Die Datenbanken enthalten ausschließlich Adressen, die dem Raum Berlin/Brandenburg zugeordnet werden können.

Postleitzahlen, sollte man denken, lassen sich auch durch Zahlen abbilden. In Deutschland werden aber führende „Nullen“ mit ausgeschrieben. Deshalb werden auch alle Postleitzahlen als Zeichenkette abgespeichert. Die Deutsche Industrie Norm 5008 [DIN5008] gibt keine diesbezügliche Angabe, ob eine Postleitzahl mit Zusätzen versehen werden darf. Die *Deutsche Post AG* gibt in ihren Richtlinien „Richtig adressieren“ [DPOST01] an, dass Zusätze wie „D-“ oder „W-“/„O-“ zu unterlassen sind. In beiden Datenbanken kommen keine Zusätze vor, da das Postleitzahlenattribut auf fünf Zeichen beschränkt ist. In beiden Datenbanken, ist das Element „Postleitzahl“ obligatorisch. In einer Datenbank kommt es vor, dass unbekannte Postleitzahlen durch „00000“ ersetzt werden. Diese Auffüllung ist möglich, da es keine Postleitzahlen in Deutschland gibt, die mit einer doppelten „0“ beginnen. Denkbar wäre auch, dass das Feld auch „`null`“ oder eine leere Zeichenkette enthalten darf.

Eine anderes Problem findet sich bei den Ortsnamen. Eines der komplizierteren Beispiele ist die Großstadt Berlin. Sie besteht aus vielen Stadtbezirken. Um eine räumliche Einordnung für den Nutzer der Adresse zu geben, wird häufig der Stadtbezirk zusätzlich zum Ort mit angegeben. Die Bezirke sind im Anschriftfeld in einer eigenen Zeile aufzuführen und nicht an den Ort mit anzuhängen [DIN5008]. Es

widerspräche auch der „Ersten Normalform“ bei den Datenbankschemata. Beide Datenbanken haben für Stadt- und Ortsteile ein eigenes Attribut erzeugt. Im Zuge der Gemeindegebietsreform in Brandenburg wurden viele Orte zu größeren Ämtern oder ähnlichen Vereinigungen zusammengefasst. Die Orte sind somit zum Teil zu Ortsteilen geworden. Trotzdem wird in einer der Adressdatenbanken der Ortsteil als Ort geführt und das Feld für den Stadtteil wird für den Landkreis genutzt.

In beiden zur Verfügung stehenden Datenbanken werden Straße und Hausnummer in einem Feld gespeichert. Bei den Straßen wird der Begriff „Straße“ stellenweise abgekürzt. Die Angabe der Hausnummer bzw. des Hausnummernbereiches ist höchst unterschiedlich. So werden bei Nummernbereichen zwischen dem Bindestrich und den Zahlen einmal Leerzeichen gesetzt, aber in einem anderen Fall weggelassen. Stellenweise werden Nummernzusätze wie „a“ klein geschrieben, in anderen Fällen groß. Manchmal wird bei Nummernzusätzen zwischen Nummer und Zusatz ein Freizeichen eingefügt. In wenigen Adressen finden sich Angaben wie Straßenecken. Noch seltener sind Adresszusätze wie „Tor 2“ oder Ähnliches zu finden.

2.4 Informationstechnisches Lösungskonzept

Die Software muss laut Pflichtenheft auf unterschiedlichen Plattformen lauffähig sein. Damit eine Übersetzung für jede einzelne Plattform entfällt, kommt JAVA® als Programmiersprache und Laufzeitumgebung zum Einsatz.

Der Link-Generator soll eine Vielzahl von verschiedensten Datenformaten für das Einlesen von Adressen unterstützen. So sind neben RDBMS auch Daten aus Dokumenten von Tabellenkalkulationsprogrammen zu unterstützen. Der Import der Daten muss über geeignete Schnittstellen erfolgen. Mit ODBC, respektive JDBC, steht eine mächtige Schnittstelle zur Verfügung, die den Zugang zu vielen RDBMS sicherstellt. Somit entfällt eine Implementierung der Schnittstelle für jedes einzelne RDBMS. Auf dem kostenpflichtigen Softwaremarkt sind Pseudo-Treiber erhältlich, die auch andere Formate, wie Tabellenkalkulationsdokumente oder Textdateien über einen solchen Treiber zugänglich machen. Somit wird JDBC die einzige Schnittstelle zum Datenimport darstellen.

Um der Vielfältigkeit der Datenformate gerecht zu werden, wird keine Datenstruktur für den Datenimport vorgegeben. Damit ist eine Datenstrukturanalyse während des Transformationsprozesses von Adresse zu Hyperlink zwingend notwendig. Eine Suche nach Stichwörtern ist nicht hinreichend genau wie man dem Abschnitt „Adressdaten in der ISO19773“ entnehmen kann. Die Datenstrukturanalyse wird zusätzlich ein Stichwortverzeichnis verwenden, um die Ergebnisse zu optimieren.

Um die Adresse im ER-Modell zu visualisieren soll, OpenSource-Software verwendet werden. Viele der frei verfügbaren SQL-Clients können die Relationen als Liste darstellen. Wenn man das ER-Modell grafisch abbilden möchte, bleibt im kostenfreien Bereich nur der SQuirreL SQL-Client [SQUIRREL01] übrig. Leider sind die grafischen Darstellungen des ER-Modells so stark mit der restlichen GUI verwoben, dass man nur die Fachkonzeptklassen verwenden kann, die die Verknüpfungen der Relationen aufnehmen. Die Darstellung muss somit selbst implementiert werden.

Da das Internetportal, welches die Fahrauskünfte erzeugt, auch mit einem Adresskatalog arbeitet, muss die beste Adresse in diesem System gefunden werden, die zur Quelladresse passt. Das Internetportal stellt eine Schnittstelle bereit, die auf eine Anfrage nach einer Adresse eine Liste mit den am besten passenden Adressen zurückgibt. Diese Abfrage erfolgt als einfache `HTTP-GET` Anfrage, die die gesuchte Adresse im Format „`PLZ Ort, Straße Nummer`“ als Parameter in der URL enthält. Die Antwort erfolgt im strukturierten XML-Format.

Ein Export der Daten mittels Serialisierung der Datenobjekte kommt nicht in Betracht, da Programme, die die Daten dann nutzen wollen, erst eine Schnittstelle implementieren müssten, um die Daten verarbeiten zu können. Ein Export in ein standardisiertes Textformat, wie ein SQL-Skript oder XML, erleichtert die Weiterverarbeitung der Daten.

Die genauen Lösungsansätze ergeben sich aus dem Pflichtenheft im Anhang.

2.5 Implementierung

2.5.1 Datenanalyse/-beschaffung

Um die Adressdaten importieren zu können, müssen die Adressen in der Datenquelle erst einmal lokalisiert werden. Die Adresse wird dafür in vier Elemente unterteilt: Postleitzahl, Ortsname, Straßenname und Hausnummer. Für die Lokalisierung wird ein mehrstufiges *Extended Boolean Retrieval* Verfahren angewandt. Das Verfahren geht nach dem Bottom-Up Prinzip vor. Zu Beginn steht die Analyse der kleinsten Struktureinheit einer relationalen Datenbank (Feld), danach folgen Attribut und Relation. Zum Schluss wird die Gesamtheit der Relationen betrachtet.

2.5.1.1 Feldanalyse

Die Analyse geht von einem Datenbanksystem als Datenquelle aus. Die unterste Ebene ist die Analyse eines Feldes (Zelle in einer Tabelle). Es kommen nur Zellen in Betracht, die vom Format „Text“ sind. Dies liegt darin begründet, dass keine Binärdaten für Adressen verwendet werden und dass es keine Adresselemente gibt, die allein durch Zahlen repräsentiert werden können. Die Zelle wird auf das Vorhandensein der Elemente `Postleitzahl`, `Ort`, `Straße` und `Hausnummer` untersucht. Für jedes einzelne Element wird dabei das Prüfungsergebnis separat abgespeichert. Die Prüfung erfolgt auf Grundlage von Musterbeschreibungen der einzelnen Elemente.

Die Postleitzahl hat ein vergleichsweise einfaches Muster. Es besteht aus fünf hintereinander geschriebenen Ziffern. Der Algorithmus prüft auf dieses Muster. Hausnummern überschreiten selten die Höhe von 1000, womit eine Hausnummer nicht fälschlich als Postleitzahl interpretiert wird. Postfachangaben können aber sehr wohl diesen Zahlenbereich erreichen, wodurch Postfachangaben fälschlich als Postleitzahl eingeordnet werden können.

Es gibt kein Muster mit dem sich Ortsnamen beschreiben lassen. Aus diesem Grund wird die Zelle in einem ersten Schritt auf Stichwörter überprüft. Die Software findet ihr Einsatzgebiet im Raum Berlin/Brandenburg, so dass sich als Stichworte die

größten Städte der beiden Bundesländer anbieten. Konkret wird auf die Städte: *Berlin*, *Potsdam*, *Frankfurt*, *Cottbus*, *Schwedt* und *Neuruppin* geprüft. Einen starken Hinweis auf einen Ortsnamen bilden auch Endungen.

Tabelle 1: Häufige Endungen von Ortsnamen

Endung
-dorf
-burg
-hausen
-berg
-walde
-städt

Wichtig ist, dass es sich hierbei wirklich um Wortendungen handelt, da diese Zeichenfolgen auch am Anfang von Straßennamen oder in anderen Begriffen vorkommen können. Einige Ortsnamen beinhalten zusätzlich ihre Rechtsform oder verliehene Titel. Diese Begriffe sind wiederum eigenständig im Ortsnamen enthalten (Bsp: Amt Oberkrämer) oder als Endung dem Namen angehangen (Bsp.: Eisenhüttenstadt), wobei Bezeichnungen wie Amt oder Gemeinde häufig weggelassen werden.

Tabelle 2: Auswahl an Gemeindetypen und -bezeichnungen

Bezeichnung	**Bedeutung**
Gemeinde	
Amt	Gemeindeverbund
Stadt	Ort mit größerer Anzahl an Einwohnern, verliehener Titel
Bad	Kurort, Kurbad o.Ä., verliehener Titel

Findet der Algorithmus eine der vordefinierten Städte, eine der Endungen oder eine der Gemeindebezeichnungen, so wird das Feld in die Liste der Felder mit Ortsnamen einsortiert.

Ein Großteil der Straßennamen lässt sich über Stichwörter und Muster finden. Häufig findet sich ein Synonym für den Begriff „*Straße*“ im Namen, der auf die Bedeutung der Straße hinweisen soll.

Tabelle 3: Bezeichnungen von Straßen nach ihrer Bedeutung

Synonym
Allee
Weg
Gasse
Damm
Promenade
Chaussee

Häufig ist auch die Abkürzung „str.“ für „Straße“ anzutreffen. In einigen Fällen findet man auch noch Hinweise auf die Umgebung im Straßennamen. So werden Straßen als „Hain“ (Bsp: Eichenhain) bezeichnet, wenn ein solcher in unmittelbarer Umgebung ist. Gleiches gilt für „*Platz*“ (Herrmannplatz), „*Hof*“ (Juliushof) und „*Kolonie*“ (Kolonie Berg). Mit „*Ring*“ und „*Wall*“ im Straßennamen gibt es Hinweise auf den Straßenverlauf oder -bauart. Wenn einer dieser Begriffe als Endung oder allein stehend im Feld vorkommt, wird dieses Feld als Straßenname erkannt.

Die Hausnummern können, genauso wie Postleitzahlen, über ein Muster beschrieben werden. Eine Hausnummer ist fast immer kleiner als 1000 und nie negativ. Sie kommt als Nummernbereich oder einzeln vor. Sie kann Zusätze wie „A“ enthalten. Für Nummernbereiche sind Trennzeichen erforderlich.

```
-   /   ,   +
```

Text 2: Liste von Trennzeichen

Zwischen Nummern und Trennzeichen kann ein Leerzeichen vorkommen, muss aber nicht. Daraus ergibt sich ein regulärer Ausdruck, mit dem man jede Hausnummer finden kann.

```
[0-9]{1,3}\s*[a-zA-Z]?\s*([-,/+]\s*[0-9]{1,3}\s*[a-zA-Z]?)?
```

Text 3: Regulärer Ausdruck für Hausnummern (PERL-Notation)

Alle Felder, die diesem Ausdruck entsprechen, werden als Hausnummer angesehen. Hierbei kann es zu Fehlern kommen. Besonders in Fällen, in denen Raumbezeichnungen (Bsp.: R-207) ein ähnliches Muster aufweisen, kann es zu solchen Fehleinteilungen kommen. Ein weiteres Beispiel wären Bestellnummern, die

sofern sie kleiner als 1000 sind und als Text gespeichert werden als Hausnummer angesehen werden. Es kommt weiterhin zu Fehlern, wenn Straßennamen eine Zahl enthalten (Bsp.: Straße des 17. Juni).

Somit existieren am Ende des Algorithmus für jedes Feld, das Text ist, vier Wahrheitswerte für das Auffinden der Adresselemente.

Eine hundertprozentige Trefferquote bei den relevanten Feldern wird dabei nicht erreicht. Das lässt sich durch die Vielfalt der Straßen- und Ortsnamen nicht abdecken. Damit man die Struktur der Adresse findet, müssen die Felder innerhalb der Datenbankstruktur weiter analysiert und zusammengefasst werden.

2.5.1.2 Relationsanalyse

Jedes Feld ist einem Attribut zugeordnet und lässt sich so in einer Gruppe zusammenfassen. Dass das Adresselement x in der Gruppe (Attribut) a vorkommt hat die Wahrscheinlichkeit p. Das einfache „*Boolean Retrieval*“ der Feldanalyse wird somit in ein „*Extended Boolean Retrieval*“ überführt. Die Wahrscheinlichkeit ergibt sich aus der Anzahl der Treffer für das entsprechende Element und der Gesamtanzahl der Felder beim Attribut.

$$p_{a,PLZ} = \frac{n_{a,PLZ}}{N_a}$$

$$p_{a,Ort} = \frac{n_{a,Ort}}{N_a}$$

$$p_{a,Straße} = \frac{n_{a,Straße}}{N_a}$$

$$p_{a,HNo} = \frac{n_{a,HNo}}{N_a}$$

Abbildung 5: Wahrscheinlichkeit p der Adresselemente in einem Attribut a. N bezeichnet die Anzahl aller Felder und n die Anzahl der Treffer.

Im nächsten Schritt werden die Attributnamen ausgewertet. Es ist zu erwarten, dass Attribute, die Adresselemente aufnehmen, auch nach diesen bezeichnet werden. Bei der Überprüfung werden sowohl die englischen als auch die deutschen Begriffe

herangezogen. Es müssen auch die englischen Begriffe ausgewertet werden, da Englisch im Bereich der Informationstechnik Fachsprache ist und viele Datenbanksysteme auch außerhalb Deutschlands entwickelt werden.

Tabelle 4: Attributbezeichner für die einzelnen Adresselemente in Kleinschreibung

Postleitzahl	**Ortsname**	**Straßenname**	**Hausnummer**
plz	siedlung	strasse	hausnummer
postleitzahl	gemeinde	straße	nummer
zip(code)	stadt	street	house
postal code	ort	thoroughfare	no
code postal	gemarkung		nr
	city		hausnr
	town		number

Wird für ein Attribut eine Übereinstimmung des Namens mit einem Stichwort aus der Liste gefunden, so wird die Wahrscheinlichkeit des Adresselementes zu dem das Stichwort gehört, um (bis zu) 20% aufgewertet. Postleitzahl und Ortsname bzw. Straßenname und Hausnummer stehen in enger Verbindung zueinander. Das bedeutet, dass die Elemente selten räumlich weit voneinander entfernt sind. Diese enge Verflechtung soll sich auch in der Bewertung der Spalten niederschlagen. Haben beide Elemente der eben genannten Paare eine Wahrscheinlichkeit von über 30%, wird überprüft wie groß der Unterschied der einzelnen Wahrscheinlichkeiten ist. Weichen die Werte um weniger als 50% voneinander ab, wird angenommen, dass beide Elemente in dem Attribut erfasst werden. Beide Elemente erhalten daraufhin die höhere der beiden Wahrscheinlichkeiten zugewiesen.

Im Anschluss wird für jedes Adresselement das beste Attribut in der Relation ermittelt. Grundsätzlich haben Attribute Vorrang, in denen ein Elementepaar (also Straße und Hausnummer oder PLZ und Ort) enthalten ist. Für die Ermittlung des besten Attributes werden dessen Wahrscheinlichkeiten für jedes Adresselement zum Vergleich herangezogen. Das Attribut mit der höheren Wahrscheinlichkeit wird dem jeweiligen Element zugeordnet, so dass am Ende der Relationsanalyse jedem Adresselement ein Attribut mit der entsprechenden Wahrscheinlichkeit zugeordnet ist.

$$p_{R,PLZ} = \max_{i=1,\dots,n} \left(p_{a_i,PLZ} \right)$$
$$p_{R,Ort} = \max_{i=1,\dots,n} \left(p_{a_i,Ort} \right)$$
$$p_{R,Straße} = \max_{i=1,\dots,n} \left(p_{a_i,Straße} \right)$$
$$p_{R,HNo} = \max_{i=1,\dots,n} \left(p_{a_i,HNo} \right)$$

Abbildung 6: Wahrscheinlichkeiten p der Elemente in der Relation R mit den Attributen a_i

2.5.1.3 Abschließende Adresslokalisierung

Zum Abschluss wird jedem Element die wahrscheinlich richtige Relation mit dem dazugehörigen Attribut zugeordnet.

Auch bei der abschließenden Ermittlung der Adresse wird nach dem Grundsatz verfahren, dass räumlich nahe stehende Elemente bevorzugt werden.

Dazu wird bei jeder Relation geprüft, ob die Pärchen eine Wahrscheinlichkeit von über 50% haben. Ist das zum Beispiel bei Straßenname und Hausnummer der Fall, wird angenommen, dass sie zusammengehören. Das Element mit der geringeren Wahrscheinlichkeit erhält dabei eine Aufwertung auf die durchschnittliche Wahrscheinlichkeit beider Elemente.

$$p_{R,PLZ} = max\left(p_{R,PLZ}, \frac{p_{R,PLZ} + p_{R,Ort}}{2} \right)$$
$$p_{R,Ort} = max\left(p_{R,Ort}, \frac{p_{R,PLZ} + p_{R,Ort}}{2} \right)$$

$$p_{R,Straße} = max\left(p_{R,Straße}, \frac{p_{R,Straße} + p_{R,HNo}}{2} \right)$$
$$p_{R,HNo} = max\left(p_{R,HNo}, \frac{p_{R,Straße} + p_{R,HNo}}{2} \right)$$

Abbildung 7: Neubewertung der Wahrscheinlichkeiten p in der Relation R

Somit wird sichergestellt, dass die Nähe aufgewertet, aber nicht überbewertet wird. Die Zuordnung von Elementen und Relationen erfolgt dann nach der maximalen Wahrscheinlichkeit in den Relationen.

Wenn über Wahrscheinlichkeiten Elemente ermittelt werden, sind immer auch Fehler möglich. Die größten Fehler resultieren aus den unvollständigen Suchlisten für Orts- und Straßennamen, sowie der großen Allgemeinheit des Hausnummernmusters. Während bei den Orts- und Straßennamen aufgrund ihrer Spezialisierung auf den geografischen Bereich wenige Fehlbewertungen zu erwarten sind, ist es bei den Hausnummern ein großes Problem. Besonders fehlende Hausnummernangaben in Adressen verringern die Wahrscheinlichkeit, die richtige Relation zu finden, die die Hausnummer enthält. Bei dem Adresselement „Hausnummer“ wird der Nutzer häufig korrigierend eingreifen müssen, wenn er weitere Felder in seinen Datenbanken unterhält, die Zahlen in Textfeldern enthalten.

2.5.1.4 Visualisierung des ER-Modells

Zur Prüfung der automatischen Adresslokalisierung durch den Nutzer muss das gesamte ER-Modell grafisch abgebildet werden. Jede Relation erhält dabei ein eigenes Schaubild, das Relationsname, Attribute und Datentypen der Attribute aufnimmt. Die Relationen werden dabei in einer tabellenartigen Struktur angeordnet. Existieren im ER-Modell Verknüpfungen zwischen den einzelnen Relationen, so werden diese durch eine Verbindung mittels einer Linie abgebildet. Die Linie beginnt/endet an den durch die Verknüpfung angegebenen Attributen.

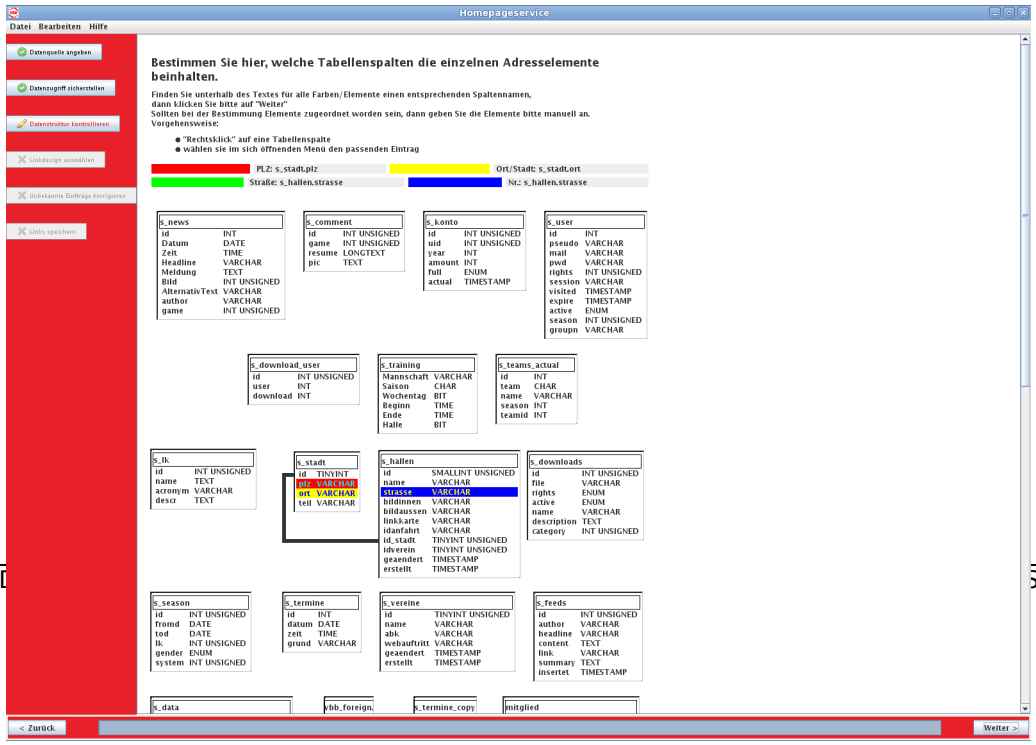


Abbildung 8: Beispiel ER-Modell in der Anwendung

Die identifizierten Adresselemente werden im GUI farblich hervorgehoben, so dass sie schnell im Modell zu finden sind. Jedes Element erhält dabei seine eigene Farbe. Ist ein Attribut mit mehreren Elementen belegt, so wird nur eine Farbe verwendet, damit die Übersichtlichkeit nicht verloren geht. Welche Elemente dem Attribut wirklich zugeordnet sind, ist über das Kontextmenü für das Attribut in Erfahrung zu bringen.

Abbildung 9: Kontextmenü eines Attributs mit den darin vorkommenden Adresselementen

Über das Kontextmenü lassen sich Korrekturen bei der Identifzierung der Adresselemente vornehmen. Im oberen Teil des Fensters werden die einzelnen Elemente mit Relation und Attribut noch einmal gesondert aufgeführt.

2.5.1.5 Datenbeschaffung

Nachdem der Ort der Adressen bekannt ist, müssen die Daten importiert werden. Wichtig ist hierbei, dass die Verknüpfungen zwischen den Relationen der Adresselemente bekannt sind. JAVA® bringt mit dem Interface `java.sql.Connection` eine Schnittstelle mit, mit der man einfach auf die Metadaten der Datenbank zurückgreifen kann. Über diese Metadaten lassen sich dann neben Tabellen auch Primär- und Fremdschlüssel suchen. Alle Informationen werden in Form einer Datenbankabfrage zurückgeliefert und nicht weiter formatiert. Mit dem SQuirreL SQL-Client [SQUIRREL01] lassen sich die Verknüpfungen über die Fachkonzeptklasse `ForeignKeyInfo` verwalten. Diese Verknüpfungen sind essentiell, wenn die Adresselemente auf verschiedene Relationen verteilt sind. Nur über diese Verknüpfungen lassen sich eindeutige Beziehungen herstellen und somit eindeutige Adressen erzeugen. Fehlen die Daten zu den Verknüpfungen, wird ein kartesisches Produkt zwischen den Elementen erzeugt, welches größtenteils nicht vorhandene Adressen enthält.

In der Abfrage der Daten wird für jedes Element ein eigener Datensatz bereitgestellt. Sind Straße und Hausnummer zum Beispiel im gleichen Feld gespeichert, wird sowohl der Straße, als auch der Hausnummer erst einmal der komplette Datensatz

zugeordnet. Die Hausnummer enthält damit auch den Straßennamen, der aber nicht gewünscht ist. Aus diesem Grunde wird in einem zweiten Schritt alles aus dem Datensatz eliminiert, was nicht dem Adresselement entspricht. Dazu werden als Erstes die Elemente für Hausnummer und Postleitzahl endgültig festgelegt. Wie oben in der Feldanalyse beschrieben, können diese beiden Elemente durch ihr Muster eindeutig identifiziert werden. Sollte der Datensatz also Straße und Hausnummer enthalten, wird die Hausnummer herausgetrennt und es bleibt nur noch der Straßenname übrig. Das gleiche passiert, wenn Postleitzahl und Ort in einem Datensatz stehen. Sollten sogar alle vier Adresselemente im Datensatz enthalten sein, wird zuerst die Hausnummer herausgefiltert. Danach wird die Postleitzahl ermittelt. Zwischen der Postleitzahl und einem Trennzeichen (nicht numerisches Zeichen oder Zeichenkettenende) wird der Ortsname ermittelt. Der verbleibende Rest ist dann der Straßenname. Straßennamen können Ziffern (z.B. Straße des 17. Juni) enthalten. In Deutschland stehen die Hausnummern immer hinter der Straßenangabe. Um eine versehentliche Interpretation einer Zahl im Straßennamen als Hausnummer zu vermeiden, wird immer das letzte Hausnummernmuster, das in der Trefferliste steht, für die Hausnummer ausgewählt.

Alle ermittelten Adressen werden dann in eine der ISO 19973 [ISO19773] entsprechenden Struktur überführt, wobei nur die Variablen für PLZ, Ort, Straße und Hausnummer mit Werten belegt werden.

2.5.2 Adresstransformation

Für die Überführung der Adressen in einen ordentlichen Link müssen die Quelladressen verifiziert werden. Dazu steht eine Schnittstelle beim Fahrinfoserver des Verkehrsverbundes Berlin-Brandenburg (VBB) zur Verfügung. An den Server übermittelte Adressen werden mit dem internen Datenbestand verglichen. Der Server gibt dann eine Liste mit Adressen zurück, die seiner Meinung nach am ehesten übereinstimmen. Im günstigsten Fall ist das eine einzige Adresse, im schlechtesten Fall sind es über 100.

Die Rückgabe der Adressen erfolgt in einem XML-Format.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:tns="vbbonline"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="vbbonline">
  <!-- validation object for an address -->
  <xsd:element name="validation">
    <xsd:complexType>
      <xsd:sequence>

        <!-- address which was the input for the validation -->
        <xsd:element   minOccurs="0"
                            name="srcAddress"
                            type="xsd:string"/>

        <!-- possible meaning of the source address -->
        <xsd:element maxOccurs="unbounded" name="evalAddress">
          <xsd:complexType>
            <xsd:simpleContent>

              <xsd:extension base="xsd:string">
                <!--   value for matching source address with
                            the evaluated address -->
                <xsd:attribute name="scoring" type="xsd:integer"
                            use="optional"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Text 4: XML-Schema der Verifizierungsantwort

Die Anfrage für die Verifizierung erfolgt über einen einfachen `HTTP-GET` Request. Die URL enthält dabei die Adresse als Parameter.

```
<URL>       ::= http://<SERVER>/<PATH>/query.exe/dol?<PARAMETER>
<PARAMETER>::= L=vs_hps&ST=2&S=<ADRESSE>&getstop=yes
<ADRESSE>   ::= <PLZ> <Ortsname>, <Straße> <Hausnummer>
```

Aufgrund der Trennung der Adresselemente beim Import lassen diese sich einfach für die URL formatieren. Die Rückgabeadressen haben das gleiche Format wie in der URL. Aufgrund des Formates und des Wissens über den Aufbau der Adressen lassen sich die Rückgabedaten wieder in die einzelnen Bestandteile zerlegen, um sie mit der Ausgangsadresse zu vergleichen.

Der Vergleich ist notwendig, damit nicht der erste Eintrag für den Link verwendet wird. Dafür werden alle Elemente einzeln betrachtet. Jedes Element bekommt dabei einen Wert (`score`) kleiner oder gleich 100 zugewiesen. Der Maximalwert von 100 bedeutet, dass das Element der Quelle zu 100% mit dem Element vom Server übereinstimmt. Ein Wert von 0 (Null) bedeutet, dass keine Übereinstimmungen gefunden worden sind.

Bei der Postleitzahl findet ein einfacher Damerau-Levenshtein Vergleich ($dld(x,y)$) statt. Die Umwandlung in eine Zahl und Bildung der Differenz ist nicht sinnvoll, da ein einziger Ziffernunterschied an der ersten oder zweiten Stelle schon riesige Differenzen erzeugen würde, obwohl sie nach dem Postleitzahlensystem dicht nebeneinander liegen. Die Differenz wird dann über die Länge der Postleitzahl der Quelle normalisiert.

$$score_{PLZ} = 100 - 100 \cdot \left(\frac{dld(PLZ_{original}, PLZ_{Server})}{length(PLZ_{original})} \right)$$

Abbildung 10: Formel für Wert des Postleitzahlenvergleichs

Der komplizierteste Vergleich ist beim Ortsnamen zu finden. Hier tritt das Problem wieder auf, dass die Originaladresse vielleicht nur den Namen eines Ortsteiles enthält. Weiterhin werden sämtliche zurückgelieferte Adressen beim Ortsnamen mit dem Namen des dazugehörigen Namen des Ortsteils versehen, falls es einen solchen in dem Ort gibt. Besonders in Berlin wirkt sich das nachteilig aus. Sämtliche Zeichenkettenvergleiche geben große Abweichungen jenseits von 50% zurück. Das kommt daher, dass die Bezirksnamen länger sind als das Wort „Berlin“ und damit eine große Menge an Einfüge-Operationen vorgenommen werden muss, damit man zum Beispiel auf „Berlin-Charlottenburg“ kommt. Die Anzahl der Operationen ist, bedingt durch die Länge der Bezirksnamen, größer als 50% des Gesamtnamens. Ein Vergleich, ob im verifizierten Ortsnamen der Quellort enthalten ist, ist als Ersatz für die Distanzberechnung nicht möglich, da bei einer Adresse für den Ort „*Burg*“ im Spreewald auch eine Adresse in „*Oranienburg*“ als 100% Übereinstimmung gewertet werden würde. Trotzdem wird verglichen, ob die verifizierte Adresse den Quellort beinhaltet. Ist der Quellort ein Teil des verifizierten Ortsnamen, so wird das Ergebnis der Übereinstimmung um (bis zu) 30% nach oben korrigiert. Bevor diese Korrektur

erfolgt, werden beide Werte einer phonetischen Analyse unterzogen. Ziel dieser Analyse ist es, Fehler zu finden, die dadurch entstanden sind, dass eine Adresse akustisch an einen Bearbeiter übermittelt wurde und dieser sich der Schreibung aufgrund der Wortmelodie nicht sicher ist. So werden Fehler wie „*Schwed*“, „*Schwehdt*“ oder „*Schwet*“ als richtig erkannt, wenn das gesuchte Wort „*Schwedt*“ ist.

Die phonetische Distanz ($phd(x,y)$) geht in die Ermittlung des Grades der Übereinstimmung zu 35% und die Damerau-Levenshtein-Distanz zu 65% ein. Das gleiche Verfahren wird auch bei den Straßennamen angewendet.

$$maxLO = max\left(length(Ort_{original}), length(Ort_{Server})\right)$$

$$maxLS = max\left(length(Straße_{original}), length(Straße_{Server})\right)$$

$$score_{Ort} = 100 - 65 \cdot \left(\frac{dld\left(Ort_{original}, Ort_{Server}\right)}{maxLO}\right) - 35 \cdot \left(\frac{phd\left(Ort_{original}, Ort_{Server}\right)}{maxLO}\right)$$

$$score_{Straße} = 100 - 65 \cdot \left(\frac{dld\left(Straße_{original}, Straße_{Server}\right)}{maxLS}\right) - 35 \cdot \left(\frac{phd\left(Straße_{original}, Straße_{Server}\right)}{maxLS}\right)$$

Abbildung 11: Formeln für Werte von Vergleich von Straßen- und Ortsnamen

Der Vergleich der Hausnummern beschränkt sich auf die Zahl der Hausnummer. Der Vergleich kann deshalb auf die Zahlen reduziert werden, da die Schnittstelle vom Server immer nur Zahlen zur Verfügung stellt. Der Server kennt weder Nummernbereiche, noch Nummernzusätze. Deshalb wird nur eine Zahl zurückgeliefert, auch wenn es sich um einen Hausnummernbereich handelt. Für den Vergleich werden aus der originalen Hausnummer die Zahl oder Zahlen, wenn es sich um einen Nummernbereich handelt, extrahiert. Bei einem Nummernbereich wird nur die erste und letzte Nummer registriert. Bei dem Vergleich muss beachtet werden, dass es in Deutschland zwei verschiedene Systeme gibt, Hausnummern zu vergeben. Beim sogenannten *Berliner System* werden die Zahlen auf der einen Straßenseite aufsteigend durchnummeriert und am Ende der Straße auf der anderen Seite vom Ende zum Anfang weiter hochgezählt. Das andere System zählt auf beiden Straßenseiten in eine Richtung linear hoch. Dabei werden auf der einen Straßenseite die ungeraden und auf der anderen die geraden Zahlen verwendet.

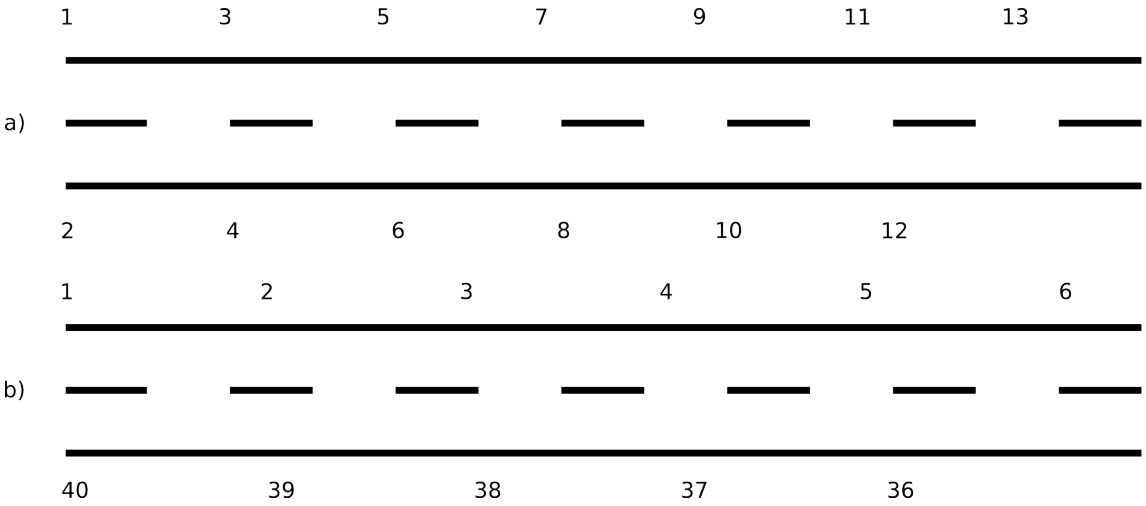


Abbildung 12: Straßennummerierung nach a) normalen System und b) Berliner System

Während beim zweiten System eine Differenz von x Nummern immer den gleichen Abstand von $\frac{x}{2}$ Grundstücken bedeutet, sind beim Berliner System die Abstände nicht wirklich aussagekräftig. Beim Berliner System können sich 100 und 1 direkt gegenüberliegen, aber auch 100 Grundstücke voneinander entfernt sein. Beim Berliner System ist eine genaue Aussage über die Lage der Grundstücke mit den entsprechenden Hausnummern somit nicht möglich. Beim anderen System kann es sein, dass durch den Unterschied von gerader und ungerader Hausnummer die falsche Straßenseite gewählt wird. Dies spielt beim Routing aber keine Rolle. Für den Vergleich der Hausnummern wird der kleinste Betrag der Differenz gesucht. Ist bei der originalen Hausnummer der Bereich von drei bis neun angegeben und die Antwort vom Server enthält die Nummer sieben, dann ist die geringere Differenz mit $|7-9|=2$. Der Betrag wird mit 4 multipliziert, um die Berechnung des Vergleichswertes vorzunehmen.

$$score_{HNo} = 100 - 4 \cdot |HNo_{original} - HNo_{Server}|$$

Abbildung 13: Berechnung der Genauigkeit der Hausnummer

Die vier Vergleichswerte für die einzelnen Adresselemente werden für die abschließende Bewertung noch einmal gewichtet. Die Postleitzahl geht mit 25% in die Bewertung ein. Straße und Ortsname gehen beide jeweils mit 30% in die Bewertung ein. Die Hausnummer hat nur einen Anteil von 15%.

Der Gesamtwert errechnet sich wie folgt:

$$score = \frac{score_{PLZ} + 1{,}2 \cdot \left(score_{Ort} + score_{Straße}\right) + 0{,}6 \cdot score_{HNo}}{4}$$

Abbildung 14: Berechnung des Gesamtwertes

Sind nicht alle Adresselemente in der originalen Adresse enthalten, so wird der Nenner für jedes nicht vorhandene Element um 0,7 (0,5 bei der Hausnummer) reduziert. Der Nenner wird nicht um das Gewicht im Zähler reduziert, da die Genauigkeit mit jedem fehlendem Element abnimmt. Stimmen alle Elemente zu 100% überein, aber die Postleitzahl fehlt, würde es für Adressen in „*Ruhlsdorf*“ mehrere 100% Treffer geben, weil im Land Brandenburg zwei „Ruhlsdorf“ existieren. Durch die Korrektur ist die Adresse nicht mehr zu 100% korrekt, sondern nur noch zu 90%.

Jede Adresse aus der Server-Antwort wird mit der originalen Adresse verglichen. Die Adresse mit dem höchsten Wert wird dabei für den Link verwendet. Sollte der höchste Wert kleiner als 80 sein, so wird dem Nutzer die Gesamtliste der Adressen vom Server zur Auswahl vorgeschlagen. Werte unter 80 sind nicht genau genug, da wenigstens ein Element nicht erkannt wurde. Damit der Nutzer auch die Unterschiede der einzelnen Adressen vom Server erkennt, kann er sich jede dieser Adressen über einen Browser bei GoogleMaps© anzeigen lassen.

==> Screenshot Linkkorrektur <==

Google® bietet für GoogleMaps© auch ein Framework an, mit dem sich die Genauigkeit der Adressen kategorisieren lassen. Die Kategorien werden hier verwendet, um den Zahlenwerten der Bewertung eine Bedeutung zuzuordnen.

Kategorie (Level)
UNKOWN_LOCATION
POST_CODE_LEVEL
TOWN_LEVEL
STREET_LEVEL
ADDRESS_LEVEL

Tabelle 5: Einstufung der Genauigkeit der Adressen

Stimmt die Postleitzahl exakt überein, dann hat der Link das „POST_CODE_LEVEL“ erreicht. Kommen Postleitzahl und Ortsname auf über 180 Punkte, so hat der Link „TOWN_LEVEL“. Wenn die für den Link genutzte Adresse das TOWN_LEVEL erreicht und die Punktzahl für den Straßennamen größer als 80 ist, hat der Link „STREET_LEVEL“. Stimmt dann noch die Hausnummer zu mehr als 80% überein, hat der Link „ADDRESS_LEVEL“. Sollte keine dieser Kategorien erreicht werden, wird der Link als „UNKNOWN_LOCATION“ eingeordnet.

Der zu erzeugende Link soll zur Fahrinfo des Verkehrsverbundes Berlin/Brandenburg führen und dort die Adresse in das entsprechende Feld schon eintragen. Dazu muss der URL als Parameter mitgeteilt werden, ob die Adresse als Start oder Ziel in die Fahrinfo eingetragen werden soll. Soll die Adresse als Startpunkt dienen, wird allen Parametern, die mit der Adresse in Verbindung stehen ein „S“ vorangestellt, ansonsten ein „Z“. Die URL hat mit einer Adresse als Ziel folgendes Schema:

```
<URL>       ::= http://<SERVER>/<PATH>/query.exe/dn?
<PARAMETER>
<SERVER>    ::= www.vbb-fahrinfo.de
<PATH>      ::= hafas
<PARAMETER>::= ZT=2&Z=<ADRESSE>&getstop=yes
```

Text 5: Unvollständige Backus-Naur-Form für die Definition der Ziel-URL

<Adresse> ist dabei die Adresse, die dann im Internetportal mit der Fahrplanauskunft eingetragen werden soll.

2.5.3 Link-Speicherung

Die ermittelten Links müssen letztendlich noch in ein Format überführt werden, damit der Nutzer sie auch weiter verwenden kann. Die Anbieter von Adressdatenbanken wollen die Daten nicht direkt in ihre Datenbanken integriert bekommen. Daher muss bei der Beschreibung auch immer die Originaladresse als Schlüssel mit angegeben werden. Die Wünsche des Formates sind recht unterschiedlich. Zwei der vier Anbieter, welche ein ER-Modell zur Verfügung stellen, bevorzugen ein XML-Format. Einer wünscht die Daten so aufbereitet zu bekommen, dass sie sich einfach in eine SQL-Datenbank importieren lassen. Ein Anbieter hat sogar den Wunsch geäußert die Daten für einen Import in Excel®-Tabellendokumente nutzen zu wollen. Allen Formaten ist gemein, dass man sie in einer einfachen Textdatei speichern kann. Die Formate unterscheiden sich in der Fülle an Zusatzinformationen, die für jeden Datensatz gespeichert werden müssen.

2.5.3.1 XML-Export

Beim XML-Export werden im Kopf des Dokumentes Metadaten erzeugt. Die Metadaten enthalten den Zeitpunkt der Erzeugung, den Raumbezug auf Berlin/Brandenburg und die Lizenz unter der das Dokument steht. Alle Angaben werden nach dem Dublin Core Standard [DUBLINCORE01] vorgenommen. Im Anschluss erfolgt die Ergebnisliste. Am Anfang jeder erfolgreichen Adresstransformation steht die Quelladresse. Der Name des XML-Elementes ist „`postalAddress`“. Dieses Element nimmt alle Attribute und Objekte aus der [ISO19773] auf, die im ER-Modell aus Kapitel 2 zu sehen sind. Im Anschluss folgt das „`data`“ Element, welches die XML-Elemente „`status`“, „`accuracy`“ und „`url`“ enthält. Die ersten beiden genannten Elemente enthalten Informationen über den Erfolg der Verifizierung. Das XML-Element „`url`“ nimmt den Hyperlink auf.

==> XML-Schema des Exports <==

2.5.3.2 CSV-Export

Eine CSV-Datei enthält die Daten in einer Art Tabelle. Jede Zeile des Textdokumentes entspricht einer Zeile und jede Spalte wird durch ein frei definierbares Trennzeichen von einer anderen getrennt. Im Normalfall ist das Trennzeichen ein Komma (`,`). Da in

den Datensätzen aber Kommata vorkommen können, werden die Daten beim Export durch ein Semikolon getrennt. Semikolons sind in den Datenformaten nicht erlaubt und können deshalb als Trennzeichen fungieren.

Die erste Zeile nimmt die Spaltennamen auf. Sie dienen später zur Identifizierung der Daten in den Spalten.

Spaltenname	Inhalt
zip	Postleitzahl
city	Ortsname
street	Straßenname
house	Hausnummer
link	Link-Daten
status	Status der Serveranfrage
accuracy	Genauigkeit der Serveradresse

Tabelle 6: Tabellenspalten beim CSV-Export

Damit der Import der Daten in Tabellenkalkulationsprogramme ohne Fehler abläuft, müssen die Postleitzahl und die Hausnummer noch maskiert werden. Die Programme interpretieren diese Spalten sonst als Zahl und stellen sie dann auch als solche dar. Damit diese Adresselemente auch wie gewünscht dargestellt werden, werden Hausnummer und Postleitzahl in Hochkommata eingeschlossen, was den importierenden Programmen anzeigt, dass es sich um Textelemente handelt.

2.5.3.3 SQL-Export

Der SQL-Export verwendet die gleichen Attribute (Spalten) wie der CSV-Export. Hinzu kommt das Attribut „`id`“, welches den Primärschlüssel der Relation aufnimmt. Dieser Primärschlüssel ist notwendig, damit die Integrität der Datenbank gewahrt bleibt und jedem Datensatz eine eindeutige Referenz zugeordnet ist. Die Relation mit den Daten bekommt den Namen „*vbb_link*“. Sollte eine solche Relation schon vorhanden sein, steht im SQL-Skript eine Anweisung, diese vorher zu löschen. Im Anschluss an die Löschanweisung folgt die Anweisung zur Erzeugung der Tabelle. Allen Attributen, außer dem Primärschlüssel, wird der Datentyp „Text“ zugeordnet. Weiterhin dürfen die Attribute „`id`“ und „`link`“ nicht leer sein (`NOT NULL`). Der Primärschlüssel ist eine Zahl und kann nur positive Werte aufnehmen. Da nicht alle Datenbanksysteme ein

automatisches Hochzählen des Primärschlüssels erlauben, wird der Primärschlüssel bei der Skripterzeugung mit generiert. Mit jedem Datensatz, der dem Skript hinzugefügt wird, wird der Primärschlüssel um eins erhöht.

2.6 Datenschutz

In letzter Zeit sind vermehrt Datenschutzverletzungen bekannt geworden. Das Recht auf informationelle Selbstbestimmung ist bisher nicht im Grundgesetz verankert, wird aber als Ausprägung des allgemeinen Persönlichkeitsrechts angesehen. Es wurde mit dem Volkszählungsurteil aus dem Jahre 1983 in die deutsche Rechtssprechung eingeführt und bedeutet in der Kurzform:„Jeder hat das Recht zu wissen, wer was über ihn weiß“ [DATENSCHUTZ01]. Die derzeitige Diskussion dieses Recht in das Grundgesetz mit aufzunehmen, verdeutlicht die Bedeutung dieses Rechtes. Die Software achtet die Belange des Datenschutzes und der informationellen Selbstbestimmung.

Die Datenquellen sind auf RDBMS beziehungsweise auf Quellen, die über ODBC angebunden sind, beschränkt. Datenbanken verfügen in den meisten Fällen über ein System zur Rechtevergabe für einzelne Nutzer. Diese Rechte verhindern, dass ein Nutzer Daten (Relationen) sehen kann, für die er nicht berechtigt ist, sie einzusehen. Durch die Zugangsdaten für die Datenquellen wird der Nutzer authentifiziert.

Bei der Lokalisierung der Adressen werden sämtliche Felder der Datenbank, welche Text enthalten, auf Adressen überprüft. Dabei landen viele Daten im System, die nicht benötigt werden. Aus diesem Grunde werden die Daten nach der Bewertung des einzelnen Feldes sofort wieder aus dem Speicher genommen.

Beim Import der Adressen werden allein die Adressen eingelesen. Es werden weder Schlüssel erfasst noch andere Daten zusätzlich mit abgefragt.

Dadurch, dass sich bei der Verifizierung der originalen Adressen nur diese im Speicher befinden, können keine weiteren Daten an den mit der Verifizierung beauftragten Server übermittelt werden. Die zu verifizierenden Adressen, werden nach der Abfrage sofort aus dem Speicher des Servers gelöscht. Die Anfragen finden weitestgehend anonym statt. Komplett anonym ist nicht möglich, da sich der Nutzer über die zum Computer gehörende Internetadresse zurückverfolgen lässt.

Bei der anschließenden Speicherung der Daten findet keine Referenzierung zu der Quelle statt. Es werden nur die Werte der Originaladresse und die Daten aus der Adresstransformation gespeichert. Eine Zuordnung der Daten zur Quelle muss durch den Nutzer selbstständig vorgenommen werden.

Die Anwendung kann im Prinzip auch noch einen zusätzlichen Nutzen für den Anwender der Software erzeugen. Durch die Angabe der Genauigkeit lässt sich zum Beispiel ermitteln, ob die in der Datenbank befindliche Adresse überhaupt existent ist. Das wird in dem Fall interessant, in dem die Datenbank Adressen von Systemnutzern enthält, die vom Datenbankverwalter nicht selbst eingetragen werden. Es lassen sich, über die Verbindung zur Datenbank, Nutzer identifizieren, die anstatt der realen Anschrift eine erfundene Anschrift eintragen.

Um die Datenintegrität der genutzten Datenbanken zu gewährleisten, werden nur Abfragen durchgeführt. Die Software enthält keine Anweisungen, die Felder aktualisiert oder neue Datensätze hinzufügt.

2.7 Usability

Die Nutzeroberfläche ist in drei Bereiche gegliedert. Links findet sich eine Seitenleiste, in der die Verarbeitungsschritte aufgeführt sind. Unten befindet sich eine Statusleiste und Buttons zur Navigation. Im zentralen Bereich befindet sich die eigentliche Arbeitsfläche. Damit kommt die Software den Anforderungen der klaren Gliederung in Bereiche [KRUG06, S. 36f]nach.

Ziel der Software ist es, dass auch Personen, die nicht sehr viel Erfahrung mit Computern haben, auf eine einfache Art und Weise ihre Adressen in Hyperlinks umwandeln können. Der Nutzer wird hierfür Schritt für Schritt angeleitet, wie er sein Ziel erreichen kann. Dazu ist der Ablauf in sechs Schritte unterteilt. In welchem Schritt man sich gerade befindet, ist in der Seiteleiste an dem Stiftsymbol () zu erkennen. Erfolgreich abgeschlossene Schritte werden mit einem Häkchen () gekennzeichnet.

Was im jeweiligen Schritt zu machen ist, steht im oberen Teil des Fensters. Dort werden unter anderem die Felder erklärt sowie die Notwendigkeit der Erhebung von Daten erläutert. Die Texte versuchen so weit wie möglich auf Fachbegriffe zu verzichten und allgemein bekannte Formulierungen zu verwenden.

Viele Elemente wurden mit Tool-Tipps versehen, um den Nutzer darauf hinzuweisen, was er mit den Elementen machen kann.

Um den Nutzer bei Prozessen, die viel Zeit in Anspruch nehmen, nicht zu beunruhigen, werden dort Fortschrittsbalken im unteren Teil des Fensters eingeblendet.

Kapitel 3 - Schlussfolgerungen

Das Projekt hat gezeigt, wie vielschichtig die Software-Entwicklung ist.

Trotz der im Vergleich zu anderen Software-Lösungen geringen Dimensionen des Programms, war die Erstellung des Pflichtenheftes ein sehr umfangreicher Teil der Arbeit. Die Analyse der Anforderungen, was denn bei einem Projekt benötigt wird, ist nicht unerheblich und bedarf einiger Übung. Während sich die Use-Cases noch relativ einfach finden lassen, muss die Aufstellung einer Funktionsliste wohl überlegt sein. Daraus ergeben sich Abhängigkeiten zu den nicht funktionellen Anforderungen, welche selten gleich offensichtlich sind. Besonders die Analyse der systemtechnischen Anforderungen und die Anforderungen an die Dokumentation waren in diesem Projekt nicht einfach zu entwickeln. Welche Anforderungen wichtig sind und welche notfalls in einer ersten Version noch ausgelassen werden könnten, ist eine Frage der Sichtweise. Deshalb sollten in größeren Projekten immer Rollen verteilt werden, die die unterschiedlichen Aspekte aus Sicht von Entwickler, Nutzer, Auftraggeber und -nehmer betrachten. Sind alle Anforderungen aufgenommen und bewertet, muss eine Lösung gefunden werden, die optimal alle Anforderungen berücksichtigt. Dabei hat sich gezeigt, dass einzelne Lösungen in der Bewertung nicht weit voneinander entfernt sind. Die Auswahl der Lösung fällt dann auch einmal zu Gunsten der Vorlieben der dominantesten Rolle, auch wenn die Zahlen dagegen sprechen. Dass die Software-Entwicklung ein dynamischer Prozess ist, hat sich daran gezeigt, dass mit jedem Prototypen immer wieder fehlende Funktionen erkannt wurden, die erst einmal als wichtig eingestuft wurden. Stellenweise stellte sich heraus, dass es sich um Wünsche handelt, die die ganze Architektur des Systems auf den Kopf stellen würden. Als Beispiel sei hier die Anzeige von Datensätzen in der Strukturansicht genannt. Sie sollte dem Nutzer helfen, Spalten mit Adresselementen zu identifizieren. In der Ansicht mit den Datenbankstrukturen hätten somit Daten importiert und sinnvoll ausgewählt werden müssen, damit sie den Inhalt der Relation auch wirklich repräsentieren. Da bei dieser Version aber auf Datensparsamkeit geachtet wurde, bleibt es bei der einfachen Ansicht der Strukturdaten. Andere Wünsche ließen sich im Rahmen der Arbeit integrieren und bilden somit die dynamischen Prozesse im Requirements-Management ab.

Die Ermittlung der Adressen in Datenbeständen ist keine einfache Angelegenheit. Es hat sich gezeigt, dass selbst in strukturierten Daten, wie einer Datenbank, die Suche nach Texten an der Ungenauigkeit der Muster scheitern kann. Die Beschreibung von Strukturen ist schon heute möglich, aber nicht überall durch Standards festgelegt. Besonders im Bereich von Adressen haben sich noch große Lücken gezeigt. Die Entwicklung des ISO-Standard 19773 weist dabei in die richtige Richtung, kann aber nur ein Zwischenpunkt der Entwicklung sein. Die Normierung des Adressformates wäre wichtig, da das nicht nur informationstechnische Vorteile hätte. Bei der immer weiter zusammenwachsenden Welt, könnten sich Menschen in ihnen fremden Ländern besser orientieren, ohne gleich technische Hilfsmittel in die Hand nehmen zu müssen. Es hat sich bei der Arbeit auch gezeigt, dass die unterschiedlichen Standards zur Formatierung von Text hinderlich sind. Während das einfache lateinische Alphabet keine Probleme verursacht, sind Umlaute immer wieder ein Problem. Da nur in seltenen Fällen der verwendete Zeichensatz ermittelt werden kann, wäre ein Umstieg aller auf das Format UTF-8 wünschenswert.

Durch die automatische Erzeugung der Links mit verifizierten Adressen lässt sich eine Menge Arbeitszeit einsparen. Das langwierige Eingeben, Auswählen und Bestätigen für jede Adresse entfällt und man kann in der Zeit, wo der Computer die Links generiert, anderen Aufgaben nachkommen. Die Verifizierung der Adressen hat den Vorteil, dass sich Nutzer des Links nicht mit einer Vielzahl von Korrekturabfragen quälen müssen, bevor sie die gewünschte Auskunft bekommen. Durch den freien Zugang zu den Quelltexten lässt sich auch eine Just-In-Time Erzeugung der Links mit den vorhandenen Klassen realisieren.

Ideen für eine Erweiterung der Software sind genügend vorhanden. Man könnte die Listen mit den passenden Adressen mit Geokoordinaten versehen. So ließe sich die Adresse dann eindeutig auf der Erde bestimmen. Könnte man diese Koordinaten in die Hyperlinks integrieren, käme es zu keinen Fehlauskünften durch Dopplungen mehr. Wie oben schon angesprochen, könnte der Software-Nutzer durch weitere Informationen zum Datenbankinhalt bei der Adresslokalisierung mehr unterstützt werden. Eine einfache Update-Funktion könnte integriert werden, um die Daten der Links zu aktualisieren. Dort könnten dann auch bisher nicht erkannte Adressen vielleicht zugeordnet werden.

Literaturverzeichnis

[BUCHMANN05] Buchmann, Andreas; Smolarek, Ralf, MySQL 5 interaktiv, dPunkt 2005

[DATENSCHUTZ01] , Informationelle Selbstbestimmung - Was bedeutet das?, http://www.datenschutz.de/recht/grundlagen/, 2008-09-01

[DIN5008] Schreib- und Gestaltungsregeln für die Textverarbeitung , Deutsches Institut für Normung , 2005

[DPOST01] , Richtig adressieren, http://www.deutschepost.de/dpag?skin=hi&check=yes&lang=de_DE&xmlFile=1004048, 2008-08-21

[DUBLINCORE01] , Dublin Core Metadata Element Set, http://dublincore.org/documents/dces/, 2008-06-30

[GROSSMANN04] Grossmann, David A.; Frieder Ophir, Information Retrieval (Algorithm ans Heuristics), Springer 2004

[ISO19115] Geographic information - Metadata EN ISO 19115:2005, Europäisches Komitee für Normung , 2003

[ISO19773] Committee Draft ISO/IEC CD 19773 , ANSI Secretariat USA , 2007

[KRUG06] Krug, Steve, Don't make me think!, mitp 2006

[LANG01] Lang, Hans Werner, Knuth-Morris-Pratt-Algorithmus, http://www.iti.fh-flensburg.de/lang/algorithmen/pattern/kmp.htm, 2008-08-12 (8:30)

[LANG02] Lang, Hans Werner, Boyer-Moore-Algorithmus, http://www.iti.fh-flensburg.de/lang/algorithmen/pattern/bm.htm, 2008-08-12 (8:30)

[MEREI07] Mérei, Emil, ,

[MID03] , Tabellenband - Mobilität in Deutschland, 2003

[NAVARRO05] Navarro, Gonzalo, Consens Mariano , String Processing and Information Retrieval, Springer 2005

[NOGUERAS05] Nogueras-Iso, Javier; Zarazaga-Soria, F. Javier; Muro-Medrano, Pedro R., Geographic Information Metadata for Spatial Data Infrastructures, Springer 2005

[SKULSCHUS04] Skulschus, Marco; Wiederstein, Marcus, XML Schema, Galileo Press 2004

[SQUIRREL01] , SQuirreL SQL, http://www.squirrelsql.org, 2008-05-10

[SULZBERGER08] Sulzberger, Carsten, Levenshtein-Algorithmus, http://www.levenshtein.de, 2008-06-10 (12:00)

[WILKES01] Wilkes, Birgit, Technische Informatik, 2006

[WILZ05] Wilz, Martin, Aspekte der Kodierung phonetischer Ähnlichkeiten in deutschen Eigennamen, 2005

Tabellenverzeichnis

Abbildungsverzeichnis

Anlagenverzeichnis

Anlage 1 - Pflichtenheft

Pflichtenheft

Automatisierung des Homepageservice für große Adressbestände

Verkehrsverbund Berlin-Brandenburg GmbH

Hardenbergplatz 2

10623 Berlin

Version 1.0 – 12.05.2008

Einführung in das Projekt

Veranlassung

In den Zeiten des Klimawandels muss auch das umweltfreundliche Reisen in das Bewusstsein der Bevölkerung gebracht werden. Der Öffentliche Personennahverkehr (ÖPNV) sollte dabei verstärkt genutzt werden. In der Region Berlin-Brandenburg nutzen schon 3,43 Millionen Fahrgäste den ÖPNV. Um die Verkehrsinformationen weiter zu verbreiten, will der Verkehrsverbund Berlin-Brandenburg (VBB) seinen Homepageservice ausbauen. Dieser Service ermöglicht Betreibern von Internetpräsenzen einen Link auf die Homepage zu stellen, der dann die Verbindungen des ÖPNV zu einer Adresse ermittelt.

Der Erstellungsprozess der Links vom Hompageservice geht dabei über mehrere Schritte, die jeweils vom Nutzer manuell zu prüfen und bestätigen sind. Das kann für wenige Adressen noch funktionieren, führt bei Datenbeständen jenseits von 200 Adressen zu großer Unübersichtlichkeit.

Durch eine Automatisierung des Homepageservice lässt sich ein Hinweis zum öffentlichen Nahverkehr für eine Adresse an vielen neuen Stellen etablieren, wo der Nutzer einer Internetseite derzeit Informationen zur Zielerreichung mit den öffentlichen Personennahverkehrsmitteln vermisst.

Zielsetzung

Ziel des Projektes ist es, Adressdaten aus einem heterogenen Datenbestand zu sammeln und in entsprechende Hyperlinks zur Fahrinfo umzuwandeln.

Dazu müssen die Adressdaten über eine Schnittstelle zugänglich gemacht werden. Die Adressen werden dann einem Automaten zugeführt, der diese dann ohne Nutzertätigkeit dem Erstellungsprozess zuführt und dann auch ausführt. Die erstellten Hyperlinks sollen dann in einem vorher definierten Datencontainer abgespeichert werden.

Projekt- und Nutzerumfeld

An dem Projekt sind mehrere Parteien beteiligt: die *Verkehrsverbund Berlin-Brandenburg GmbH* (Auftraggeber), der die Automatisierung des Homepage-Service möchte, die *HaCon Ingenieurgesellschaft mbH* welche den derzeitigen Homepage-Service bereitstellt und die Anbieter von Adressdatenbanken (Benutzer), die ein Interesse an der Nutzung des automatisierten Homepage-Service haben.

Für das Projekt stehen 6 Monate Zeit zur Verfügung. Die Software soll als Freeware, besser als Open-Source-Projekt, entwickelt werden.

Beschreibung der Ausgangssituation (Ist-Zustand)

Vorhandenes System

Der derzeitige Homepageservice wird über eine Internetseite abgewickelt. Er ist Teil des Routing-Dienstes VBB-Fahrinfo (http://www.vbb-fahrinfo.de). Er wird von der HaCon Ingenieurgesellschaft administriert und gewartet. Das Nutzerinterface stellt eine Internetseite dar, die durch den gesamten Erstellungsprozess des Hyperlinks führt. Der Prozess ist in 4 Schritte unterteilt. Im ersten Schritt wird die Form des Links festgelegt. Im zweiten Schritt werden die optionalen Voreinstellungen erfasst und auf Konsistenz geprüft. Wurde zum Beispiel keine Vorauswahl des Zieltyps getroffen und es existieren Einträge sowohl bei Haltestellen, Adresse und Sonderziele, muss der gewünschte Typ nochmals nachgetragen werden. Im dritten Schritt müssen die Nutzungsbedingungen akzeptiert werden. Im letzten Schritt wird dann der Quelltext zur Verfügung gestellt. Diesen muss sich der Nutzer dann aus der Internetseite herauskopieren und selbsttätig abspeichern.

Mengengerüst

Die Datenbestände an Adressen für die zugrundeliegende Fahrinfo werden von NAVTEQ geliefert. Die Fahrinfo-Software stellt Anforderungen hinsichtlich der Form an optimale Adressangaben. In der Version 5.24 muss folgendes Format eingehalten werden:

[PLZ] [Ort][-Ortsteil]?, [Straße] [Hausnummer]

Tabelle 7: Formatdefinition

PLZ	Postleitzahl als numerische Angabe mit 5 Stellen und führenden Nullen
Ort	Zeichenkette mit Ortsangabe
Ortsteil	*Optional*: Ortsteil bei größeren Städten und Gemeinden
Straße	Straßenname
Hausnummer	Nummer als alphanumerische Zeichenkette, ohne Möglichkeit

	der Bereichsangabe

Die Formulardaten werden ungesichert zum Server übertragen. Die Anfragen werden vom Server für die Dauer der Nutzung des HompageService auf dem Server zwischengespeichert. Nach Ablauf der Sitzung werden die Daten gelöscht.

Aufgabenstellung (Soll-Zustand)

Die Software soll den Prozess der Link-Erstellung automatisieren und bei Bedarf (z.B. Fehlermeldungen) Rückfragen an den Nutzer stellen. Das komplizierte Eintippen von Adressen soll durch die automatische Suche von Adressen in heterogenen Datenbeständen ersetzt werden. Die Filterung der Adressen soll automatisch erfolgen, sofern Adressstrukturen ermittelbar sind. Ansonsten soll der Nutzer den Ablageort der Adressen manuell vorgeben. Die Software soll mit allen Datenbanken, die den SQL Standard SQL-92 unterstützen und Datentypen unterscheiden, arbeiten können. Weiterhin soll die Software Datenbestände aus kommaseparierten Textdateien und von Tabellenkalkulationsdokumenten (.odt, .xls) ermitteln können.

Datenmodell

Die Eingangsdaten (Adressbasis) sind in tabellarischer Form vorzuhalten. Welches Format die Tabelle hat, soll nicht eingegrenzt werden. Der Typ der Tabelle soll wie oben beschrieben eine Datenbank, eine kommaseparierte Textdatei, ein Tabellenkalkulationsdokument nach ISO 26300 oder XLS-Dokument des Formates Word 98® sein. Die Ausgabe muss in die gleichen Datentypen möglich sein.

Nutzungs-Szenarien (Use Cases)

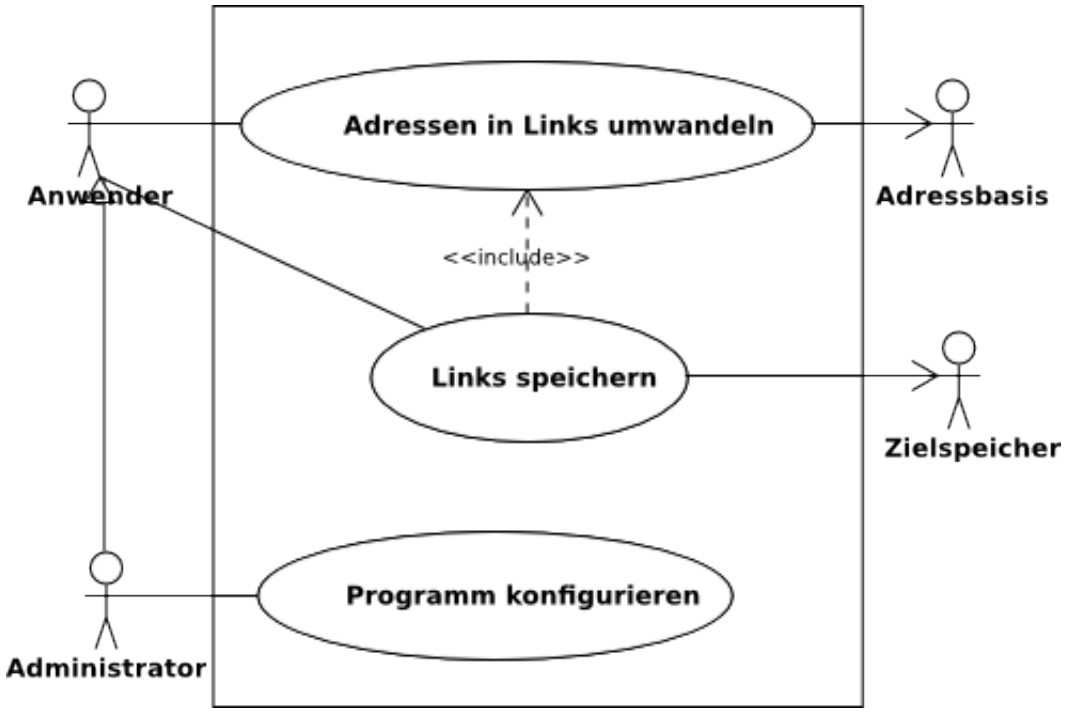


Abbildung 15: Anwendungsfalldiagramm

***Use-Case:* Programm konfigurieren**

Ziel: Einstellungen vornehmen, so dass der allgemeine Anwender ohne Einschränkungen arbeiten kann

Kategorie: sekundär

Vorbedingung: -

Nachbedingung Erfolg: Adressen sind auslesbar und erzeugte Links lassen sich speichern

Nachbedingung Fehlschlag: Die Anwendungsfälle „Adressen in Links umwandeln“ und „Links speichern“ sind nicht fehlerfrei ausführbar

Akteure: Anwender, die besondere Kenntnisse über die angeschlossenen Systeme haben (in der Regel Administratoren)

Beschreibung:

1 Zugangsdaten für die Adressdaten einstellen

2 Speicherform der Links festlegen

3 Eingestellte Daten in einem Profil speichern

Alternativen:

1a Ort der Adressdatei festlegen

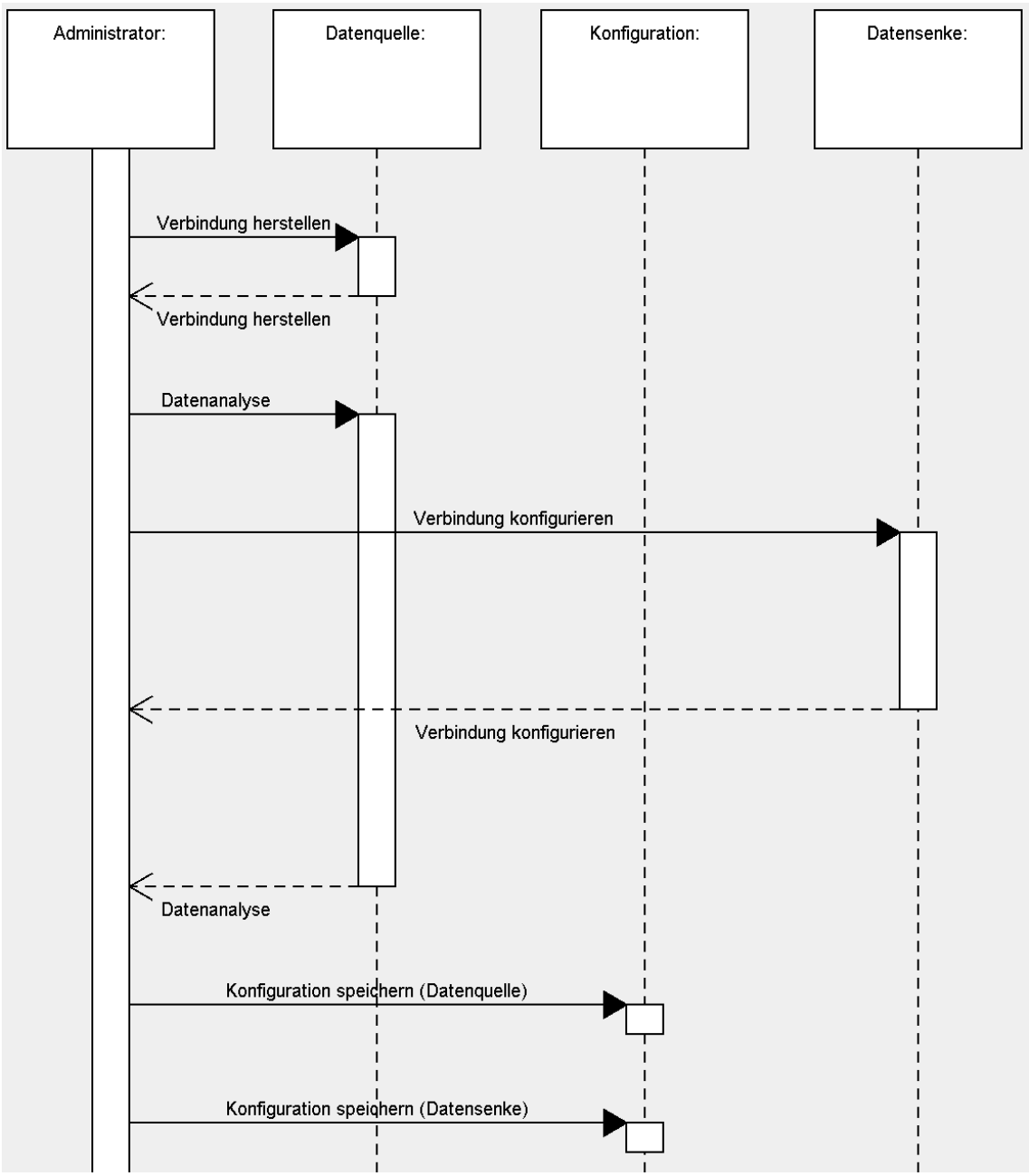


Abbildung 16: Abfolge bei der Programmkonfiguration

***Use-Case:* Adressen in Links umwandeln**

Ziel: Adressen aus einer Adressliste in Links zum Routenplaner des VBB umzuwandeln

Kategorie: primär

Vorbedingung: Zugang zu Adressdaten muss gewährleistet sein

Nachbedingung Erfolg: Jeder Adresse wurde ein Link zugeordnet

Nachbedingung Fehlschlag: Links wurden nicht erzeugt mit entsprechender Meldung

Akteure: Anwender und die Adressdatenbasis

Beschreibung:

1 Verbindung zur Datenbasis herstellen

2 Adressdaten ermitteln

3 Adressen in Links umwandeln

Erweiterung:

3a Links korrigieren

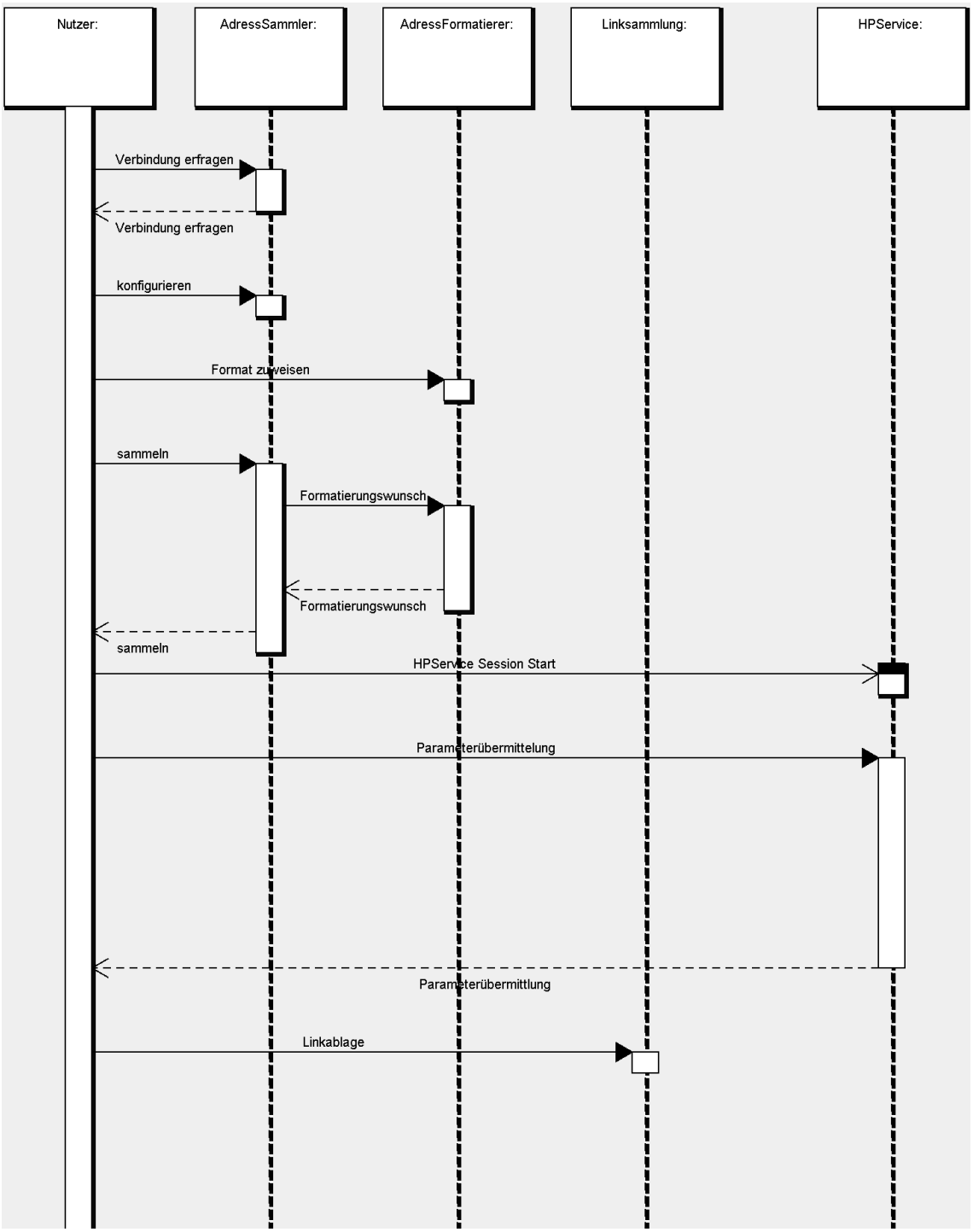


Abbildung 17: Abfolge bei der Link-Umwandlung

Use Case: **Links speichern**

Ziel: erzeugte Links dauerhaft auf einem Datenträger zu speichern

Kategorie: primär

Vorbedingung: Liste mit Links

Nachbedingung Erfolg: Links sind wieder abrufbar in einem anderen System abgelegt

Nachbedingung Fehlschlag: Fehlermeldung mit der Fehlerursache

Akteure: Anwender, Speichersystem

Beschreibung:

1 Links speichern

2 Speicherung überprüfen

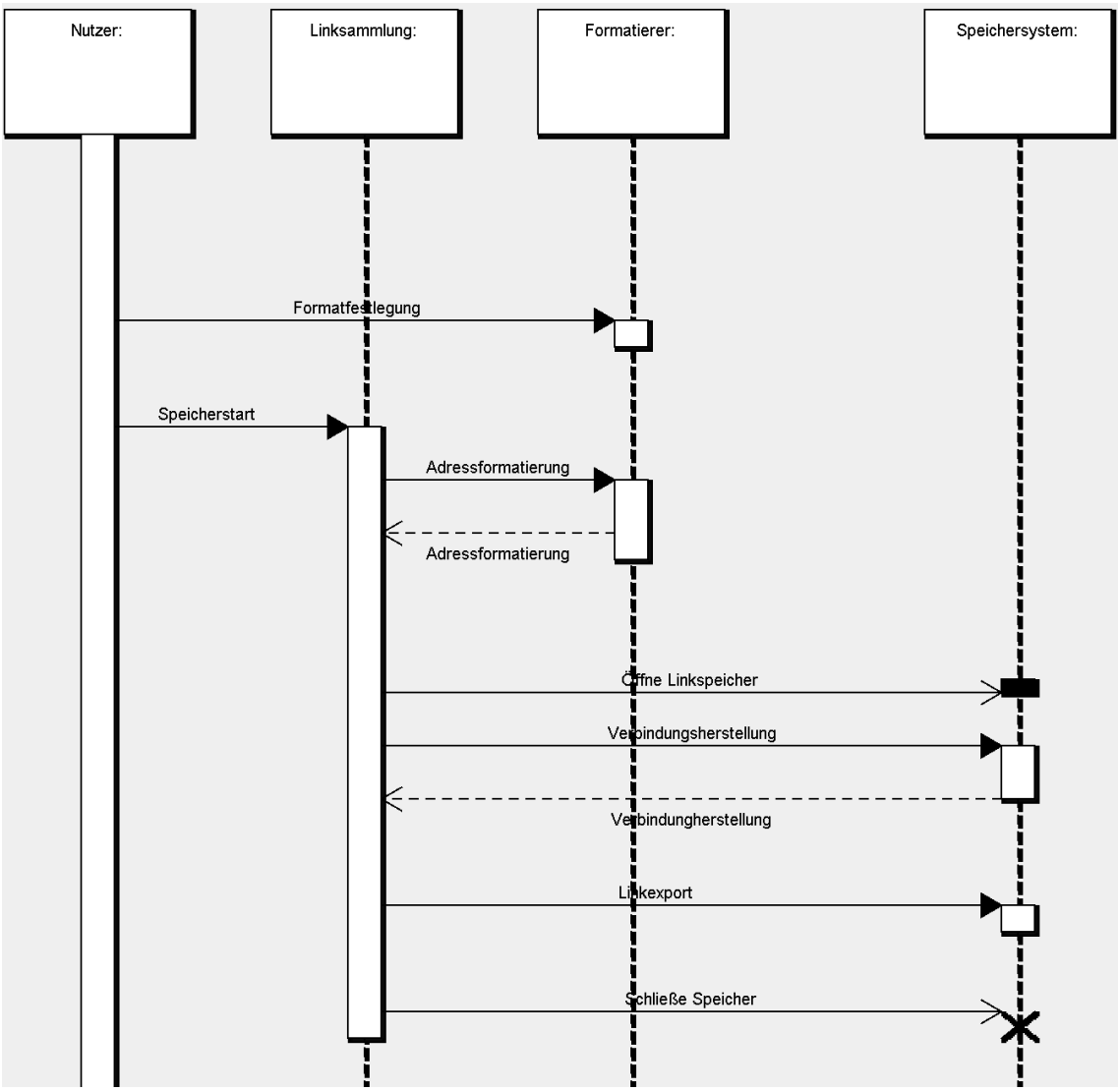


Abbildung 18: Abfolge bei der Link-Speicherung

Schnittstellen

Die Software muss eine Schnittstelle bereitstellen, über die mit dem RDBMS kommuniziert werden kann. Die Kommunikation muss bidirektional möglich sein, um auch das Ablegen von Links zu ermöglichen.

Anforderungen

Funktionale Anforderungen

/ F1 / Konfiguration speicherbar

Beschreibung: Eine Struktur einer Datensammlung sollte sich speichern lassen, damit bei späteren Durchläufen der Konfigurationsaufwand entfällt.

Gewichtung: Soll

Nachweis: Vorhandensein von Konfigurationsdateien und das erfolgreiche Einlesen zur Wiederverwendung

/ F2 / Finden von Adressen in heterogenen Datenbeständen

Beschreibung: In Datensammlungen, die nicht nur aus Adressen bestehen, sollen die Adressen automatisch gefunden werden. Notfalls sind die ungefähren Ablageorte anzugeben, um die Suche zu beschleunigen oder einzuschränken.

Gewichtung: Muss

Nachweis: Finden von Adressdaten, die in einer Einzeltabelle vorliegen und Finden von Adressen, die in einer normalisierten Datenbank vorgehalten werden

/ F3 / Automatisierung des Generierungsprozesses

Beschreibung: Die 4 Schritte zur Generierung der Links sollen weitestgehend ohne Interaktion des Nutzers ablaufen. Dabei bedeutet weitestgehend, dass Nachfragen aufgrund nicht gefundener Adressen möglich sein müssen.

Gewichtung: Muss

Nachweis: Erstellung eines Links von einer Adresse, die dem System bekannt ist, ohne dass der Nutzer eine Eingabe tätigen muss

/ F4 / Speicherung der generierten Links in verschiedenen Zielformaten

Beschreibung: Die erzeugten Links sollen in den Formaten der Ausgangsdaten persistent gespeichert werden können.

Gewichtung: Muss

Nachweis: Speicherung eines Links und danach Abruf dieses Links

Leistungs- (Performance-) Anforderungen

/ P1 / Maximale Dauer zur Linkgenerierung darf 5 Sekunden pro Adresse nicht überschreiten

Beschreibung: Die Automatisierung des Generierungsprozesses geht über mehrere Schritte. Diese Schritte sollen in der Summe nicht mehr als 5 Sekunden pro Adresse in Anspruch nehmen.

Gewichtung: Soll

Nachweis: Durchführung von 100 Prozessen und Messung der benötigten Zeit

/ P2 / Mindestens 50 verschiedene Generierungsprozesse müssen parallel möglich sein

Beschreibung: Die Software soll abhängig von der zur Verfügung stehenden Bandbreite so viele Automatisierungsprozesse wie möglich parallel durchführen können.

Gewichtung: Muss

Nachweis: Zählen der Anfragen, die schon gestartet, aber noch nicht beendet wurden

/ P3 / Ermittlung des Adressortes darf nicht länger als 1 Minute dauern

Beschreibung: Die Ermittlung der Struktur soll eine Minute nicht überschreiten, damit der Nutzer nicht auf die Idee kommt, dass die Software fehlerhaft sei.

Gewichtung: Soll

Nachweis: Vorlage einer Datenbank mit 100 Tabellen, in der nur ein Adressfeld existiert

Qualitäts- (Zuverlässigkeits-) Anforderungen

/ Q1 / Paketlaufzeiten von bis zu 7 Sekunden dürfen nicht zu einem Verbindungsabbruch führen

Beschreibung: Aufgrund von viel Netzwerkverkehr kann es zu Verzögerungen bei der Zustellung der Daten kommen. Es sind mindestens 7 Sekunden Verzögerungszeit einzuplanen, bevor die Verbindung abgebrochen werden darf

Gewichtung: Soll

Nachweis: Künstliche Verzögerung durch entsprechende Hard- oder Software

/ Q2 / Keine Falschzuordnungen von generierten Links und Ausgangsadressen

Beschreibung: Die erzeugten Links müssen nach der Speicherung noch eineindeutig den jeweiligen zu Grunde liegenden Adressen zugeordnet werden können. Dabei dürfen keine Fehler auftreten.

Gewichtung: Muss

Nachweis: Test an 100 Beispieldatensätzen

/ Q3 / 97% der Adressen mit bekannter Postleitzahl dürfen zu keiner Nachfrage führen

Beschreibung: Der Automatisierungsprozess soll den Nutzer entlasten. Dazu soll der Automatisierungsprozess bei Rückfrage durch den Server in 97% der Fälle selbstständig die richtige Wahl in der Auswahl treffen, ohne beim Anwender nachzufragen.

Gewichtung: Muss

Nachweis: Test an 100 Beispieldatensätzen

Systemtechnische Anforderungen

/ S1 / System soll betriebssystemunabhängig sein

Beschreibung: Für Datenbanken kommen die unterschiedlichsten Betriebssysteme in Frage, deshalb muss auch die Software betriebssystemunabhängig sein

Gewichtung: Muss

Nachweis: Lauftest auf einem Linux und Windows-System

/ S2 / System soll ab 40kBit/s arbeiten können

Beschreibung: Die Arbeit soll nicht an der Bandbreite der Netzwerkverbindung scheitern. Daher ist die minimale Geschwindigkeit für die Funktionsweise auf die Leistung des analogen Telefonnetzes festgesetzt.

Gewichtung: Kann

Nachweis: Durch künstliche Verzögerung mittels Hard- und/oder Software

/ S3 / Die Software muss mit eingeschränkten Rechten funktionieren

Beschreibung: Damit die Software auch von Anwendern benutzt werden kann, die zwar den Zugang zur Datenbank kennen, aber keine Installationsrechte haben, muss die Software auch mit eingeschränkten Rechten funktionieren

Gewichtung: Muss

Nachweis: Test von einem Nutzeraccount mit eingeschränkten Rechten

Anforderungen an die Dokumentation

/ D1 / Eine kontextunabhängige Print-Dokumentation muss für die Inbetriebnahme vorliegen

Beschreibung: Damit die Software auch in Betrieb genommen werden kann muss eine Dokumentation vorliegen, welche den Ablauf der Inbetriebnahme beschreibt und die benötigten Daten aufzeigt.

Gewichtung: Muss

Nachweis: Vorlage

/ D2 / Eine kontextunabhängige Print-Dokumentation soll für den normalen Anwender im Betrieb vorliegen

Beschreibung: Damit der Anwender sich zu einzelnen Optionen oder Ähnlichem Notizen machen kann, ist eine gedruckte Version der Dokumentation notwendig.

Gewichtung: Soll

Nachweis: Vorlage

/ D3 / Kontextsensitive Dokumentation in der Software

Beschreibung: Zur Unterstützung des Anwenders ist es wünschenswert, diesem bei seinen Arbeitsschritten durch passende einblendbare Hilfen zu unterstützen.

Gewichtung: Muss

Nachweis: Aufrufbare Hilfedialoge zu den einzelnen Prozessen

Zukunftsaspekte

Für die Zukunft kann man sich vorstellen den Generierungsprozess Just-In-Time in den Aufbau einer Website zu integrieren. Das heißt bei Abruf der Webseite, wird der Generierungsprozess für eine Adresse gestartet und der Link dann direkt in die entsprechende Website mit eingebettet.

Eine weitere Erweiterung wäre eine Update Funktion, bei der alle bisherigen Adressen auf Aktualisierungen überprüft werden, ohne dass die alten Daten überschrieben werden müssen.

Morphologischer Kasten

	Lösung 1	Lösung 2	Lösung 3	Lösung 4
/ B1 / Datenbank-verbindung	ODBC (JDBC)	Textimport	TCP	
/ B2 / Datenanalyse	Datenstrukturanalyse	Stichwortsuche	Datenstruktur-vorgabe	
/ B3 / Adresssammlung	unsortiert	Sortierter Text	Strukturierte Daten	
/ B4 / Adress-formatierung	DIN 5008	Französische Norm	HACON-Format	Eigenes Format
/ B5 / Linkerstellung	XML-Datenaustausch	HTML-Parser	SOAP (WebServices)	Proprietäres Protokoll
/ B6 / Linkspeicherung	Datenbankexport	Objektserialisierung	Textexport	

/ B1 / Datenbankverbindung

Lösung 1: Die Daten werden über eine SQL-Schnittstelle ermittelt

Lösung 2: Daten werden als Klartext importiert

Lösung 3: Die Datenquelle wird über ein anderes TCP-Protokoll angebunden

/ B2 / Datenanalyse

Lösung 1: Durch Analyse der Struktur der Anordnung der Daten und der Struktur der Inhalte werden die relevanten Informationen ermittelt

Lösung 2: Die Datenbestände werden über Stichworte durchsucht und darüber die Datensätze ermittelt

Lösung 3: Es werden Daten gesucht, die einer vorgegebenen Datenstruktur entsprechen

/ B3 / Adresssammlung

Lösung 1: Die Adressen werden als Text, so wie sie gefunden werden, einfach hintereinander angeordnet

Lösung 2: Nach einem Sortierkriterium werden die Adressen als Text geordnet

Lösung 3: Die gefundenen Adressen werden in eine Datenstruktur eingepasst um später einzelne Elemente extrahieren zu können

/ B4 / Adressformatierung

Lösung 1: Die Adressen werden nach Norm DIN 5008 formatiert, wobei Zeilenumbrüche gelöscht werden

Lösung 2: Die Adressen werden nach französischer Norm formatiert, wobei Zeilenumbrüche gelöscht werden

Lösung 3: Die Adressen werden nach Idealvorgaben für den HompageService formatiert

Lösung 4: Eigenes Format, welches die Bestandteile Ort und Straße/HNr. enthält

/ B5 / Link-Erstellung

Lösung 1: Die Kommunikation mit dem Server läuft über eine proprietäre XML-Schnittstelle

Lösung 2: Die Kommunikation mit dem Server erfolgt mittels HTTP-Requests und einem HTML-Parser

Lösung 3: Die Kommunikation mit dem Server erfolgt über eine standardisierte SOAP-Schnittstelle

Lösung 4: Für die Kommunikation wird ein eigenes Übertragungsprotokoll entwickelt

/ B6 / Link-Speicherung

Lösung 1: Die Daten werden als SQL-Statements zum Ziel exportiert

Lösung 2: Die Objekte, die aus der Linkerstellung resultieren, werden einfach serialisiert

Lösung 3: Die Daten werden als Klartext in einer Datei abgespeichert

Verträglichkeitsmatrix

Funktionen: Datenbankverbindung und Datenanalyse

	ODBC (JDBC)	Textimport	TCP	
Datenstruktur-analyse	ja	n/a	ja, bei Vorgabe der Strukturmerkmale	
Stichwortsuche	ja, aber nur in einzelnen Tabellen	ja	ja	
Datenvorgabe	ja	ja	ja	

Funktionen: Datenanalyse und Adresssammlung

	Datenstruktur-analyse	Stichwortsuche	Datenvorgabe	
unsortiert	n/a	ja	ja	
Sortierter Text	ja	ja	ja	
Strukturierte Daten	ja	ja, wenn sich aus den Stichwörtern Strukturen ableiten lassen	ja	

Funktionen: Adresssammlung und Adressformatierung

	unsortiert	Sortierter Text	Strukturierte Daten	
DIN 5008	n/a	ja	ja	
Französische Norm	n/a	ja	ja	
HACON Format	n/a	ja	ja	
Eigenes Format	n/a	ja	ja	

Funktionen: Adressformatierung und Link-Erstellung

	DIN 5008	Französische Norm	HACON Format	Eigenes Format
XML-Datenaustausch	ja	ja	ja	ja
HML-Parser	ja	ja	ja	ja
SOAP (WebServices)	ja	ja	ja	ja
Proprietäres Protokoll	ja	ja	ja	ja

Funktionen: Link-Erstellung und Link-Speicherung

	XML-Datenaustausch	HTML-Parser	SOAP (WebServices)	Proprietäres Protokoll
Datenbankexport	ja	ja, wenn der Parser den Dokumentenbaum erzeugen kann	ja	ja
Objekt-serialisierung	n/a	n/a	ja	ja
Textexport	ja	ja	ja	ja

Lösungsmöglichkeiten

Variante 1 (ODBC-Anbindung):

Die Adressdaten werden über einen ODBC-Treiber angebunden. (/ B1 / - Lösung 1). Über eine Datenstrukturanalyse (/ B2 / - Lösung 1) werden die Adressen lokalisiert. Die Adressen werden in eine Datenstruktur (/ B3 / - Lösung 3) überführt, woraus dann die Textrepräsentation im Hacon-Format (/ B4 / - Lösung 3) erstellt wird. Die Erstellung der Links erfolgt mittels HTTP-Anfragen am derzeitigen Internetinterface und wird durch Parsen der Antwortseiten automatisiert (/ B5 / - Lösung 2). Die erzeugten Links werden mittels SQL-Skript exportiert. (/ B6 / - Lösung 1).

Module: / B1 / (1) - / B2 / (1) - / B3 / (3) - / B4 / (3) - / B5 / (2) - / B6 / (1)

Variante 2 (Textimport):

Die Daten müssen in Textform einlesbar sein (/ B1 / - Lösung 2). Die Adressen werden über eine Stichwortsuche (oder reguläre Ausdrücke) gefiltert (/ B2 / - Lösung 2) und in eine sortierte Liste (/ B3 / - Lösung 2) überführt. Die Adressen werden in das Adressschema der Deutschen Post (/ B4 / - Lösung 1) konvertiert. Die Adressen werden über eine XML-Schnittstelle (/ B5 / - Lösung 1) an den Server von Hacon übermittelt. Die erzeugten Links werden wieder als Text (/ B6 / - Lösung 3)in einer Datei abgespeichert.

Module: / B1 / (2) - / B2 / (2) - / B3 / (2) - / B4 / (1) - / B5 / (1) - / B6 / (3)

Variante 3 (TCP-Schnittstelle):

Die Daten werden über eine Netzwerkschnittstelle in einem vorgegeben Format entgegengenommen (/ B1 / - Lösung 3 + / B2 / - Lösung 3) und aufgrund des Formates in eine Datenstruktur (/ B3 / - Lösung 3) integriert. Die Adressen werden nach Französischer Norm (/ B4 / - Lösung 2) formatiert und über einen WebService (/ B5 / - Lösung 3) in die entsprechenden Links umgewandelt. Die so erstellten Objekte werden über Objektserialisierung (/ B6 / - Lösung 2) persistent abgespeichert.

Module: / B1 / (3) - / B2 / (3) - / B3 / (3) - / B4 / (2) - / B5 / (3) - / B6 / (2)

Gewichtung der Anforderungen und Ziele

Anforderung	funktionale Ziele	Gewichtung in %	ODBC Anbindung		Textimport		TCP-Schnittstelle	
/ F1 /	Konfigurationsspeicherung	8	7	56	9	72	6	48
/ F2 /	Adresssammlung	9	8	72	6	54	9	81
/ F2 /	Adressformatierung	9	10	90	7	63	9	81
/ F2 /	Link-Erstellung	4	10	40	5	20	10	40
/ F3 /	Adresssuche	14	8	112	2	28	10	140
/ F4 /	Link-Speicherung	7	5	35	10	70	4	28
	Performanceziele							
/ P1 /	min. Dauer Generierung	3	6	18	7	21	5	15
/ P2 /	Multithreadfähigkeit	6	8	48	6	36	9	54
/ P3 /	min. Dauer Adresssuche	2	5	3	8	16	5	10
	Qualitätsziele							
/ Q1 /	min. Verbindungsabbrüche	5	5	25	7	35	6	30
/ Q2 /	keine Falschzuordnungen	6	9	54	6	36	8	48
/ Q3 /	min. Nachfragen	9	8	72	3	27	9	81
	Systemziele							
/ S1 /	Betriebsysytemunabhängigkeit	8	9	72	10	80	8	64
/ S2 /	Durchsatzfähigkeit	3	8	24	9	27	8	24
/ S3 /	Nutzeranforderungen	7	8	56	6	42	6	42
		100	*114*	***777***	*101*	***627***	*112*	***786***
	Entwicklung (Konzept)	*50*	7	350	3	150	4	200
	Herstellung (Umsetzung)	*10*	8	80	4	40	5	50
	Betrieb	*40*	6	240	10	400	5	200
		100	*21*	***670***	*17*	***590***	*14*	***450***
	Ergebnis TNW			77,7		62,7		78,6
	Ergebnis Aufwand			67		59		45

Tabelle 8: Gewichtung der Anforderungen

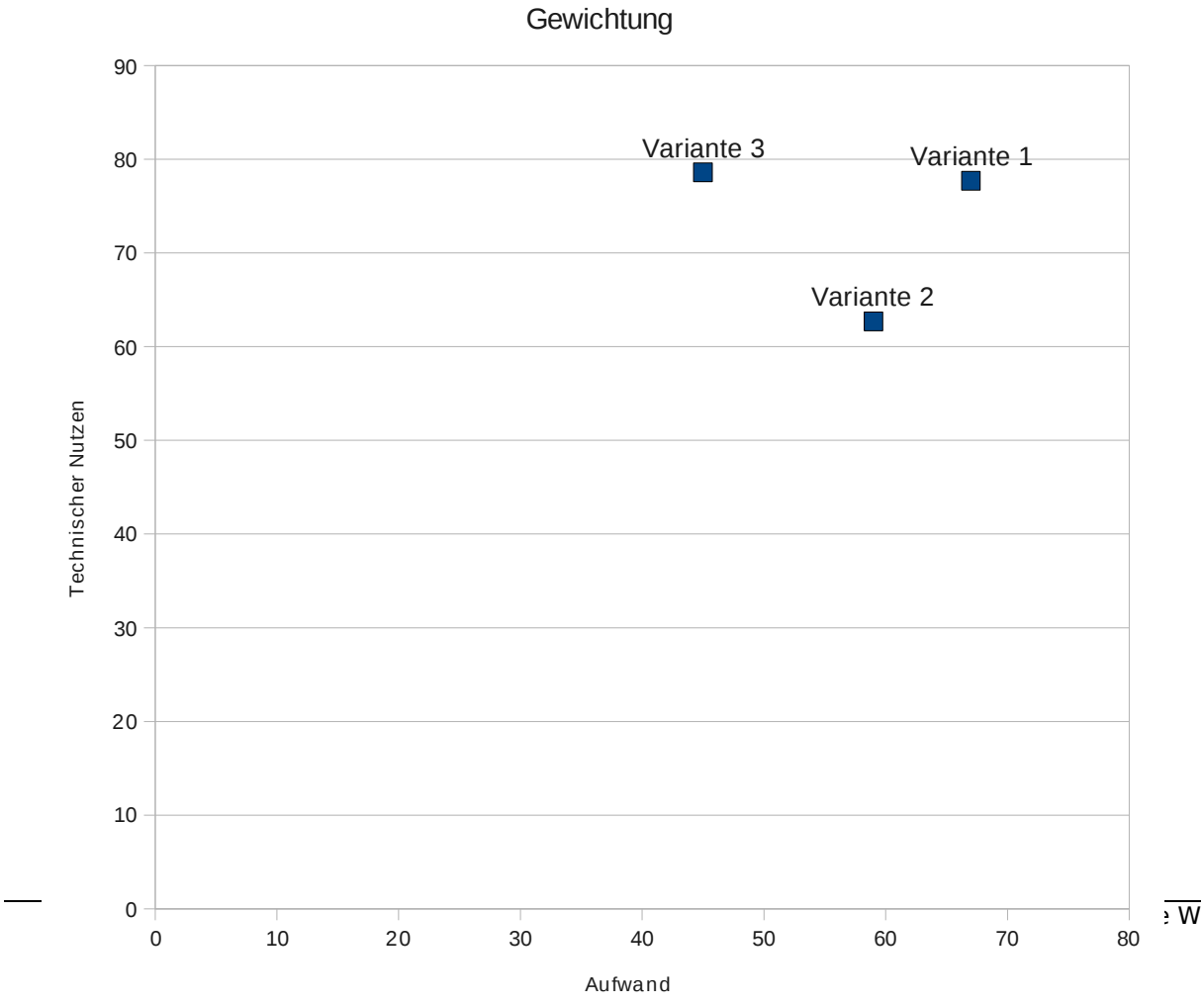


Diagramm 1: Vergleich der Varianten nach Aufwand und technischem Nutzen

Aufgrund der starken Gewichtung von automatisierter Adresssuche und Plattformunabhängigkeit treten bei den Lösungen 1 und 3 beim technischen Nutzwert die Vorteile in den Vordergrund. Die besseren Ergebnisse für die Variante 1 rühren daher, dass die Formate für den Datenimport schon vorgegeben sind. Bei einer freien Wahl des Formates bliebe der Technische Nutzwert konstant, der Aufwand für Entwicklung und Implementierung würde aber sehr stark steigen. Die Datenanalyse von Text ist noch nicht weit fortgeschritten und daher fällt der technische Nutzwert für das Projekt eher gering aus.

Somit fällt die Entscheidung zugunsten der Variante 1.

Technische Produktumgebung

Die Anforderungen zur Betriebssystemunabhängigkeit ergeben, dass die Software in JAVA entwickelt und betrieben wird. Die Nutzerplattformen müssen also eine JAVA Laufzeitumgebung mindestens in der Version 5.0 bereitstellen und für ihre Datenbanken einen JDBC oder ODBC Treiber vorweisen können.

Teilprodukte

Das Projekt lässt sich in 2 Teilprodukte teilen.

Datenbankanalyse

Die Datenbankanalyse soll dem Nutzer viel Konfigurationsarbeit abnehmen. Der Nutzer soll sich nicht um Details seiner Daten kümmern, sondern die Daten einfach an die Prozessautomatisierung weiterleiten.

Die Datenbankanalyse soll aufgrund von vorher definierten Kriterien Adressen in einer Datenbank finden. Siehe **/ F3 /**

Prozessautomatisierung

Bei diesem Teilprodukt geht es darum, die Generierung des Links mit der Adresse ohne Interaktion des Nutzers zu ermöglichen und das für eine Vielzahl von Adressen zu parallelisieren. Siehe **/ F2 /**

Anlage 2 - Bedienungsanleitung

Bedienungsanleitung

VBB-HomepageService -Generator

Installation

Speichern Sie das Archive `HomepageService.zip` in einem von Ihnen frei gewählten Ordner.

Entpacken Sie das Archiv in ein Verzeichnis ihrer Wahl. Wenn Sie Windows XP® oder Windows Vista® als Betriebssystem verwenden, öffnen Sie mit einem `Rechtsklick` auf das Archiv das Kontextmenü und wählen Sie `Extrahieren...` , um es in das Verzeichnis ihrer Wahl zu entpacken.

Inbetriebnahme/Konfiguration

Sie benötigen zur Ausführung der Anwendung eine JAVA® Runtime Environment (JRE) der Version 6.0 oder höher. Sie können sich die neueste Java Version unter http://java.sun.com/javase/downloads/index.jsp herunterladen und installieren, falls sie kein JAVA 6.0 oder höher besitzen.

Sie benötigen für die Funktionsfähigkeit der Software einen JDBC-Treiber, um ihre Daten importieren zu können. Eine Liste an verfügbaren Treibern finden Sie auf den Internetseiten von Sun® unter: http://developers.sun.com/product/jdbc/drivers. Die Seite ist in englischer Sprache verfasst.

Kopieren Sie den heruntergeladenen Treiber in das Verzeichnis `lib` .

Die Software unterstützt von Hause aus eine Menge an Treibern/Datenbanken; dazu gehören:

1. Apache Derby Client, Apache Derby Embedded
2. Axion
3. Firebird JayBird
4. FrontBase
5. H2, H2 Embedded, H2 In-Memory, HSQLDB In-Memory, HSQLDB Server, HSQLDB Standalone, HSQLDB Web Server
6. HXTT Access Client, HXTT Access Embedded, HXTT CSV Client, HXTT CSV Embedded, HXTT DBF Client, HXTT DBF Embedded, HXTT Excel Client, HXTT Excel Embedded, HXTT Paradox Client, HXTT Paradox Embedded, HXTT Text Client, HXTT Text Embedded
7. IBM DB2 App Driver, IBM DB2 Net Driver
8. Informix
9. InstantDB
10. InterClient
11. Intersystems Cache
12. JDBC ODBC Bridge
13. jTDS, jTDS Microsoft SQL, jTDS Sybase
14. Mckoi
15. Microsoft MSSQL Server JDBC Driver
16. Mimer SQL
17. MMMySQL Driver
18. MySQL Driver
19. Oracle OCI Driver, Oracle Thin Driver
20. Pointbase Embedded, Pointbase Server

21. PostgreSQL
22. SAPDB
23. Sunopsis XML
24. Sybase Adaptive Server Anywhere, Sybase Adaptive Server Enterprise
25. ThinkSQL

Treiber hinzufügen

Sollte ihr Treiber nicht in der Liste aufgeführt sein, müssen Sie den Treiber der Treiberliste hinzufügen. Öffnen Sie dazu zum Bearbeiten die Datei `default_drivers.xml` im Programmverzeichnis. Häufig muss die Datei über das Kontextmenü mit einem `Rechtsklick` zum Bearbeiten geöffnet werden.

Fügen Sie hinter `<list>` folgendes Konstrukt ein:

```
<de.vbbonline.db.DriverDesc
Class="net.sourceforge.squirrel_sql.fw.sql.SQLDriver">
      <driverClassName></driverClassName>
      <identifier
Class="net.sourceforge.squirrel_sql.fw.id.UidIdentifier"></identifier>
      <jarFileName/>
      <jarFileNames Indexed="true"/>
      <name></name>
      <url></url>
      <websiteUrl></websiteUrl>
</de.vbbonline.db.DriverDesc>
```

Ergänzen Sie dieses Konstrukt mit den Werten ihres Treibers. Schreiben Sie dazu hinter:

- `<driverClassName>` den (Klassen-)Namen des Treibers, wie er auf der Internetseite des Herstellers angegeben ist
- `<identifier Class="net.sourceforge.squirrel_sql.fw.id.UidIdentifier">` eine Zahl kleiner als -50, die sie noch nicht verwendet haben

- <name> einen selbst ausgedachten Namen, über den Sie nachher den Treiber in der Anwendung identifizieren
- <url> ein Schema, wie der Pfad zu Datenbank angegeben wird (optional)
- <websiteUrl> die Internetadresse des Treiberherstellers (optional)

Speichern Sie die Datei und verlassen Sie das Programm, mit dem Sie die Datei bearbeitet haben. Wenn der Treiber im Verzeichnis `lib` gespeichert ist, können Sie ihn nun in der Anwendung verwenden.

Programm starten

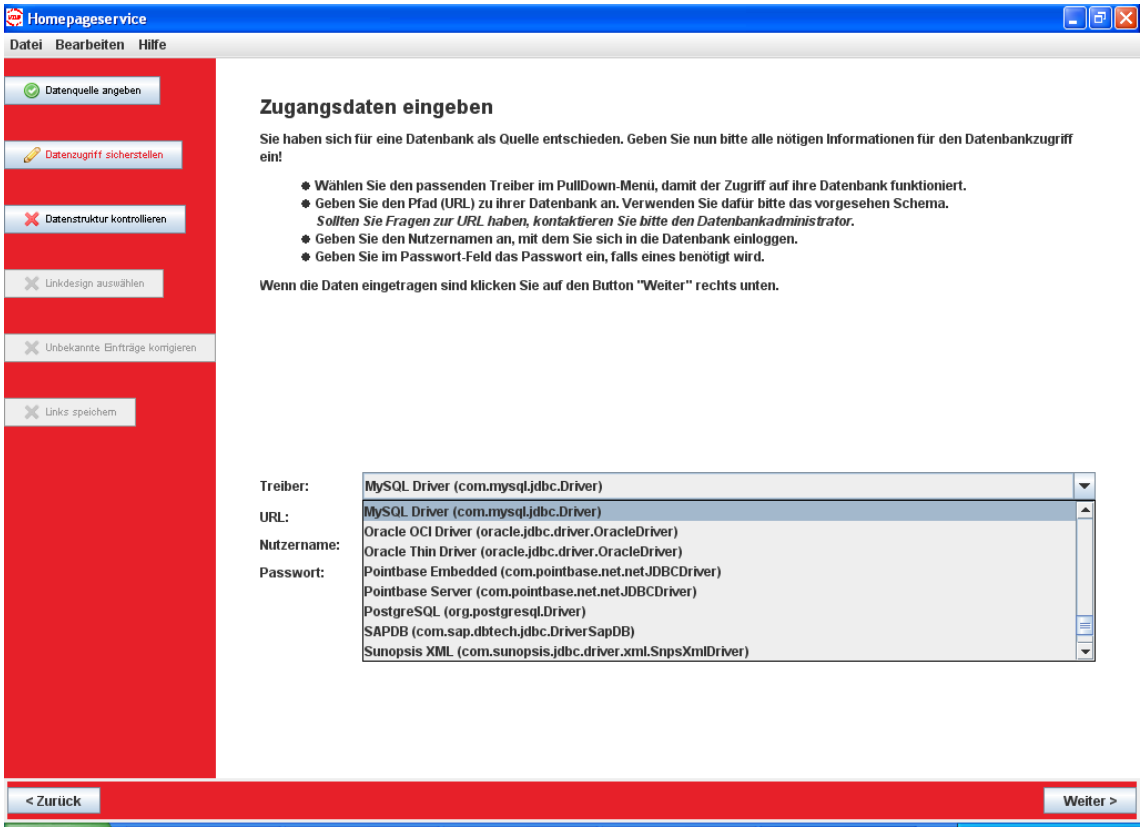
Starten Sie das Programm unter Windows® mit einem Doppelklick auf die Datei `HomepageService.bat` . Es kann sein, dass die Endung `.bat` nicht angezeigt wird. Wenn Sie das Programm unter Linux benutzen wollen, öffnen Sie eine Konsole und wechseln Sie in das Programmverzeichnis. Starten Sie das Programm mit dem Befehl: `java -jar HomepageService.jar`.

Adressen importieren

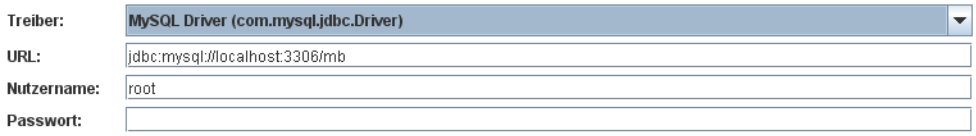
Quelle der Daten auswählen

Wählen Sie im ersten Schritt eine Datenquelle aus. In der *Version 1.0* lassen sich nur Adressen aus relationalen Datenbank Managementsystemen importieren. Wählen Sie die Quelle mit einem `Mausklick` auf den Button `Datenbank` aus.

Zugangsdaten für die Datenquelle bereitstellen



Für den Datenbankzugang wird der Name des Datenbanktreibers benötigt. Wählen Sie den passenden Treiber im gleichnamigen Pull-Down-Menü aus. Sollte ihr Treiber nicht in der Liste sein, verfahren Sie bitte wie unter „*Treiber hinzufügen*“ beschrieben. Dazu muss das Programm geschlossen sein.



Wenn Sie den passenden Treiber ausgewählt haben, geben Sie den Ort der Adressdatenbank im Textfeld URL an. Das Feld sollte mit einer Schemabeschreibung vorbelegt sein, die Ihnen zeigt, wie die Angabe auszusehen hat. Sollten Sie nicht mit der URL ihrer Datenbank vertraut sein, fragen Sie bitte Ihren Datenbankadministrator.

Tragen Sie in den Feldern Nutzername und Passwort ihre Zugangsdaten ein. Das Passwort wird aus Sicherheitsgründen nicht gespeichert!

Mit dem Button Weiter > rechts unten im Fenster kommen Sie zum nächsten Schritt.

Datenstruktur bestätigen

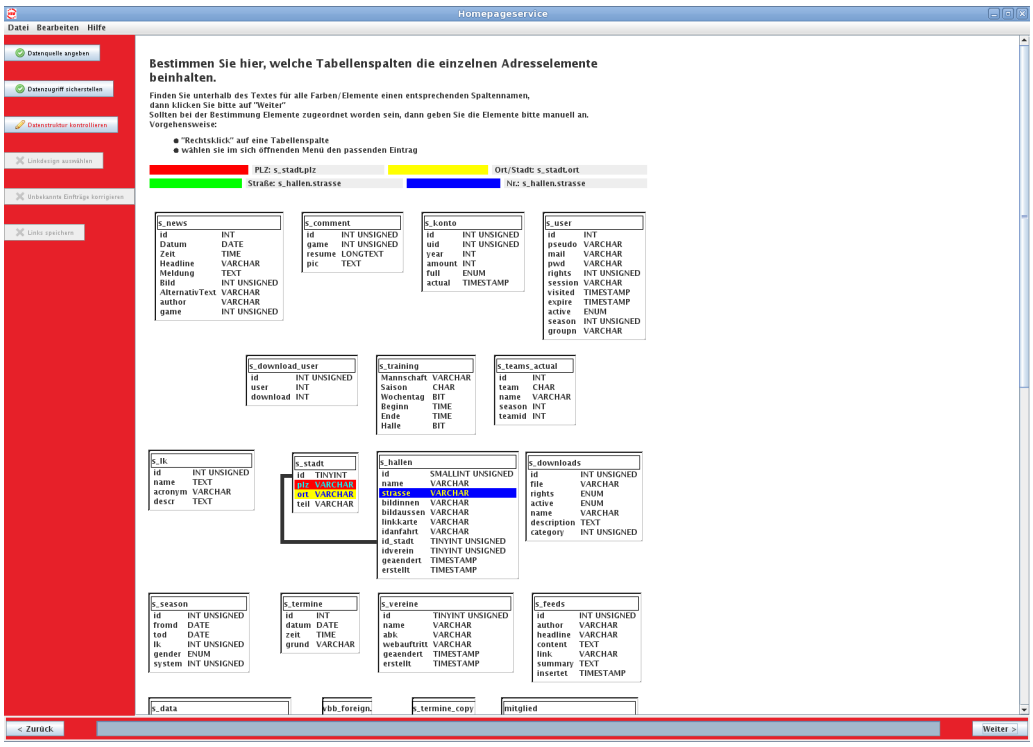
Nach der Analyse der Struktur der Datenbank wird diese noch einmal grafisch aufbereitet dargestellt. Jeder Kasten entspricht dabei einer Tabelle in der Datenbank. In den Kästen stehen oben die Tabellennamen. Darunter folgen auf der linken Seite die Spaltennamen und rechts die Datentypen zu den Spalten. Linien zwischen zwei Tabellen bedeuten, dass diese miteinander verknüpft sind. Sollte eine Spalte mit einer Farbe hinterlegt sein, wurde diese als ein Element einer Adresse identifiziert.

Bedeutung der Farben:

- Rot: Postleitzahl
- Gelb: Ort
- Grün: Straße
- Blau: Hausnummer

Mit einem Rechtsklick auf diese Spalte finden Sie heraus, welche Adresselemente in dieser Spalte wirklich gespeichert sind. Sie lassen sich an dem Haken vor dem Element erkennen.


Postleitzahl
Ort/Stadt
Straße
Hausnummer



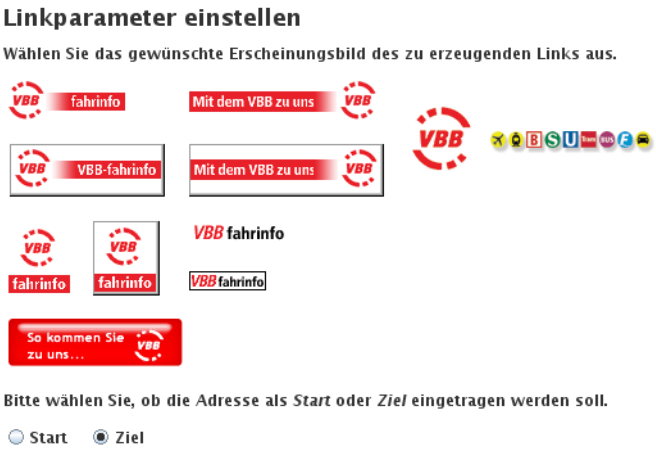
Prüfen Sie im oberen Bereich der Fensters, ob für die Adresselemente Postleitzahl, Ort, Straße und Hausnummer eine Tabelle mit Spalte angegeben ist. Wenn Sie der Meinung sind, dass nicht alle Elemente richtig zugeordnet sind, müssen Sie die Zuordnung manuell vornehmen. Suchen Sie sich dafür im Modell die Tabelle, die das zu verändernde Element enthält. Suchen sie sich die gewünschte Spalte und öffnen Sie das Kontextmenü mit einem `Rechtsklick` . Klicken Sie auf das Element, welches Sie für die Spalte aktivieren wollen. Die Tabelle sollte dann oben bei der Elementliste angezeigt werden.

Wenn die Zuordnungen vollständig sind, gehen Sie weiter zur Auswahl des Link-Design, in dem Sie den Button `Weiter >` betätigen

Links erstellen

Linkdesign auswählen

Wählen Sie das Aussehen des späteren Links aus. Dafür stehen Ihnen 10 Grafiken zur Verfügung.

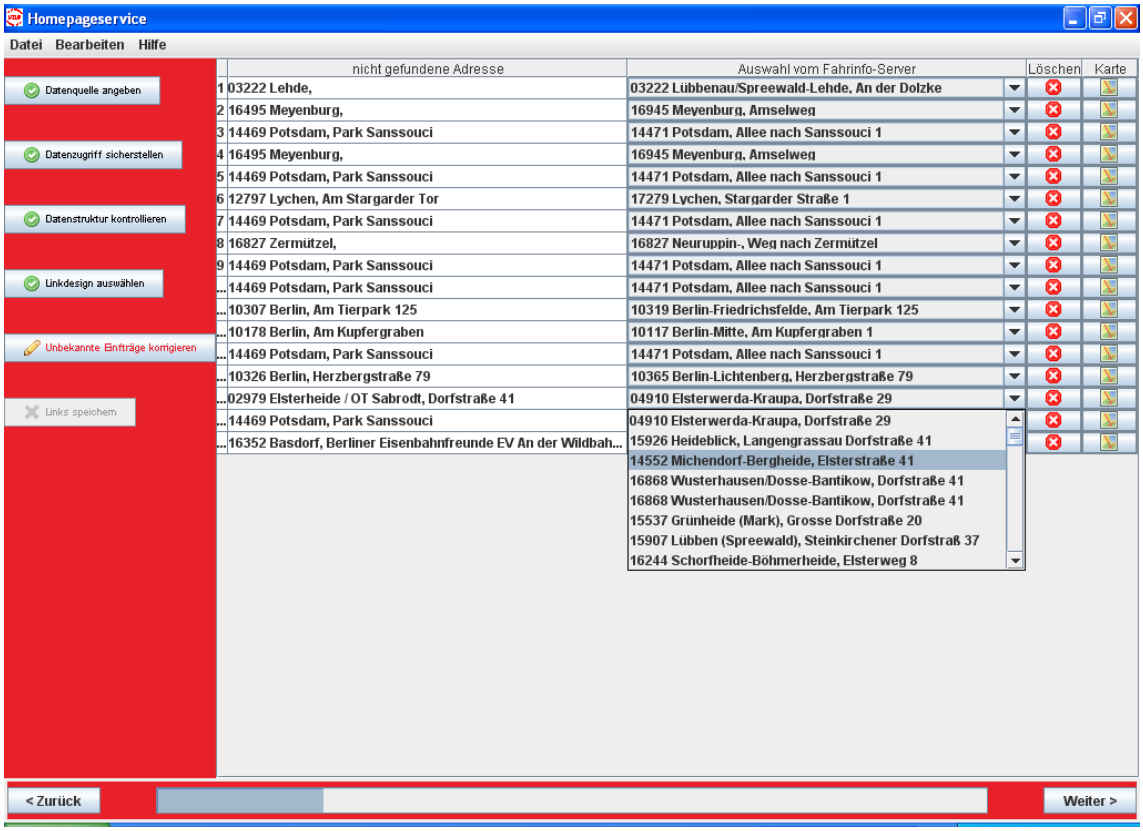


Die von Ihnen ausgewählte Grafik wird grau hinterlegt.

Wählen Sie an dieser Stelle zusätzlich, wo im Routenplaner die Adresse eingetragen werden soll. Sie können die Adresse als Startpunkt oder als Ziel der Routenplanung definieren. Wählen Sie dazu unterhalb der Grafiken den entsprechenden Eintrag aus.

Mit `Weiter >` wird die Erzeugung der Links gestartet.

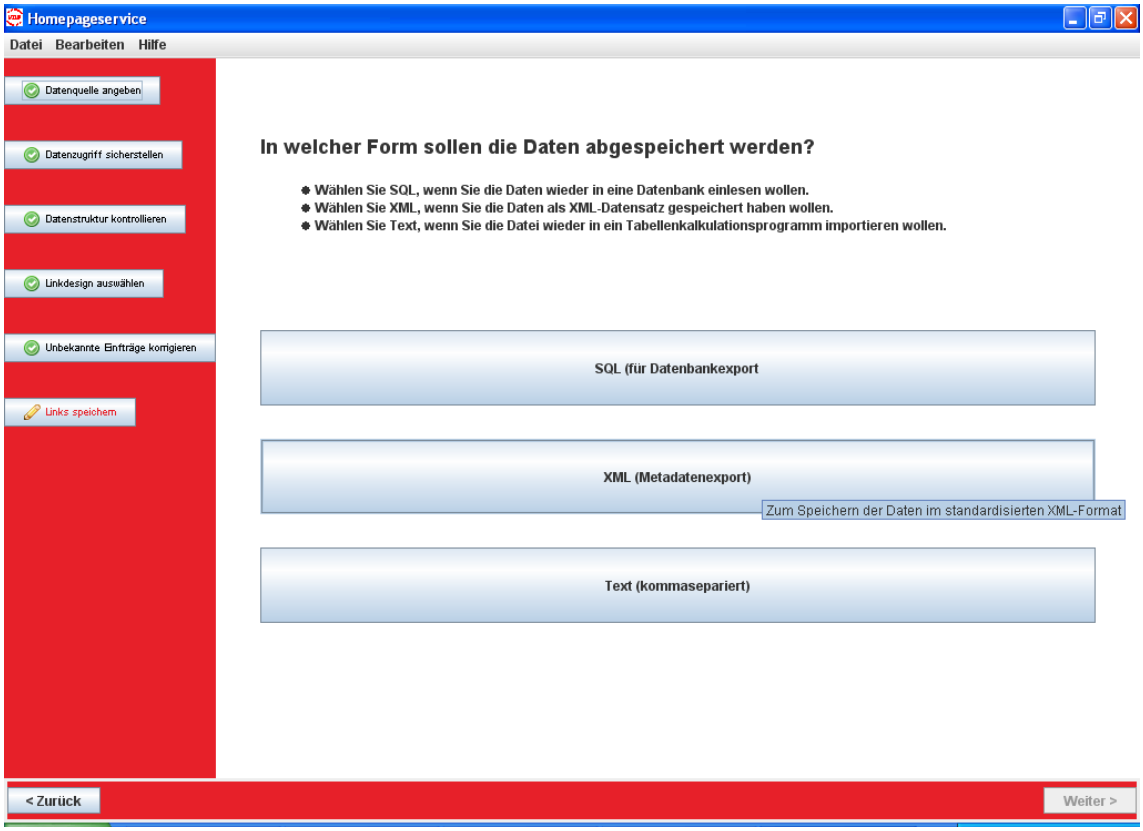
Links korrigieren



Es kann sein, dass die Adressen nicht eindeutig erkannt werden. Wählen Sie im zur Adresse gehörenden Pull-Down-Menü die Adresse aus, die ihrer Meinung am besten zur gesuchten Adresse passt. Sollten Sie keinen passenden Eintrag finden, löschen Sie die Adresse aus dem Datenbestand in dem sie auf das Button mit dem weißen Kreuz auf rotem Grund ( ) drücken. Falls sie nicht wissen, welcher Ort sich hinter einer Adresse in der Auswahl verbirgt, klicken Sie auf das Kartensymbol ( ). Daraufhin öffnet sich ein Browser, der die derzeit ausgewählte Adresse bei GoogleMaps® anzeigt.

Wenn Sie auf `Weiter >` klicken, kommen sie zum letzten Schritt der Link-Generierung.

Links speichern



Wählen Sie im letzten Schritt ein Format und eine Datei in dem sie die Links speichern wollen. Sie haben die Wahl zwischen einem XML-Format, einem SQL-Skript oder einer einfachen Textdatei. Geben Sie im Dateidialog den Namen an, unter dem sie die Datei speichern wollen.

Profile

Datenschutz