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

Diplomarbeit Nr. 3220

Interaktive, verständnisorientierte Optimierung von semantisch-annotierten Visualisierungen

Hannes Pfannkuch

Studiengang:	Softwaretechnik
Prüfer:	Prof. Thomas Ertl
Betreuer:	Dipl. Phys. Michael Raschke
begonnen am:	21. Juli 2011
beendet am:	20. Januar 2012
CR-Klassifikation:	H.1.2, H.5.2, I.3.6

Zusammenfassung Die weltweit erzeugte Datenmenge wächst rapide an. Um diese Daten für Menschen verständlich aufzubereiten, sind Visualisierungen ein bewährtes Mittel. Da Visualisierungen aber in der Regel keine eindeutige Bedeutung haben, kann es leicht zu Missverständnissen kommen. Ein Konzept, das die Wahrscheinlichkeit solcher Missverständnisse senken soll, wird in der vorliegenden Diplomarbeit vorgestellt. Es zeigt, wie Visualisierungen mit semantischen Metainformationen annotiert werden können. Dadurch ist es für Benutzer möglich, die Visualisierungen an ihre persönlichen Bedürfnisse anzupassen, indem sie entweder einzelne grafische Elemente oder das komplette Visualisierungskonzept austauschen. Um die Umsetzbarkeit dieses Konzepts zu zeigen, wurde ein Prototyp entwickelt, in dem die wichtigsten Aspekte des Konzepts implementiert wurden. Zum Abschluss werden das Konzept, der Prototyp und die verwendeten Technologien evaluiert.

Abstract The worldwide created data volume is rapidly increasing. To prepare these data for human users, visualizations are a proven way. Because visualizations in general don't have an explicit meaning, they easily can be misunderstood. This diploma thesis shows a concept to lower the probability of misunderstandings. It shows a technologie to annotate visualizations with semantic meta information. Thereby, it becomes possible for users, to adapt the visualizations to their individual needs by changing single graphical elements or the complete visualization concept. To show the applicability of this concept, a prototype was developed, which implements the central aspects of the concept. At the end, the concept, the prototype and the used technologies are evaluated.

Inhaltsverzeichnis

1. Einführung

Der Titel der vorliegenden Diplomarbeit - Interaktive, verständnisorientierte Optimierung von semantisch-annotierten Visualisierungen - ist nicht unbedingt intuitiv verständlich und wird deshalb hier erläutert.
Zuerst wird auf die zweite Hälfte des Titels - semantisch-annotierte Visualisierungen - eingegangen. Die Semantik ist einerseits die „Bedeutung, Inhalt (eines Wortes, Satzes oder Textes)“ und andererseits ein “Teilgebiet der Linguistik, das sich mit den Bedeutungen sprachlicher Zeichen und Zeichenfolgen befasst“ [httc].
Eine Annotation ist ein „Strukturelement, durch das bestimmte Daten [...] eingebunden werden“[htta]. Annotieren ist der Vorgang, in dem die Daten eingebunden werden sollen. Die zweite Hälfte des Titels bezieht sich also auf Visualisierungen, in die Daten eingebunden werden sollen, die den Visualisierungen eine bestimmte Bedeutung geben.
Die erste Hälfte des Titels - Interaktive, verständnisorientierte Optimierung - wird nun erläutert. Die Visualisierungen sollen interaktiv und verständnisorientiert optimiert werden können. „Interaktiv“ ist die Anpassung deshalb, weil es zu einem Wechselspiel zwischen Benutzer und Programm kommt: Der Benutzer wählt zum Beispiel einige Elemente der Visualisierung aus, die ausgetauscht werden sollen und das Programm zeigt ihm daraufhin eine angepasste Version der Visualisierung. „Verständnisorientiert“ soll verdeutlichen, dass ein Benutzer eine Visualisierung solange anpassen kann (durch Austauschen einzelner Elemente oder des gesamten Visualisierungskonzeptes), bis er sie richtig versteht.

1.1. Motivation

Die zunehmende Nutzung des Internets - unter anderem hervorgerufen durch den Gebrauch mobiler Geräte - führt zu einem ständigen Anstieg des erzeugten Datenvolumens. Während im Jahre 2007 281 Exabyte (281 * 10^{18} Byte) an Daten entstanden sind, waren es 2010 bereits mehr als ein Zettabyte (10^{21} Byte). Das Datenvolumen wird auch weiter rapide ansteigen (siehe Abbildung 1). Dies wird vor allem durch die zunehmende Verbreitung von Smartphones in Schwellen- und Entwicklungsländern verursacht. 90% dieser erzeugten Daten sind unstrukturiert, liegen also nicht in Datenbanken, sondern verteilt auf unzähligen Servern, gespeichert in den unterschiedlichsten Formaten. Mittlerweile wird ein großer Teil nicht mehr durch Menschen erzeugt, sondern automatisch durch technische Geräte und Algorithmen [JG11]. Diese Faktoren (immer größeres Datenvolumen, unstrukturiert, automatisch erzeugt) machen es zunehmend schwierig, die Daten sinnvoll zu analysieren und zu verwenden. Eine grafische Aufbereitung in Form von Visualisierungen stellt eine Möglichkeit dar, diese unvorstellbaren Datenmengen besser analysieren zu können. Bisher fehlt es noch an Methoden, mit denen Daten automatisiert visualisiert werden können. Häufig werden Visualisierungen von Menschen mit technischer Ausbildung (Ingenieure, Informatiker) erstellt, jedoch von Marketingexperten oder Managern verwendet. Oder sie werden von Marketingexperten erstellt und sollen von Kunden verstanden werden. Durch das un-

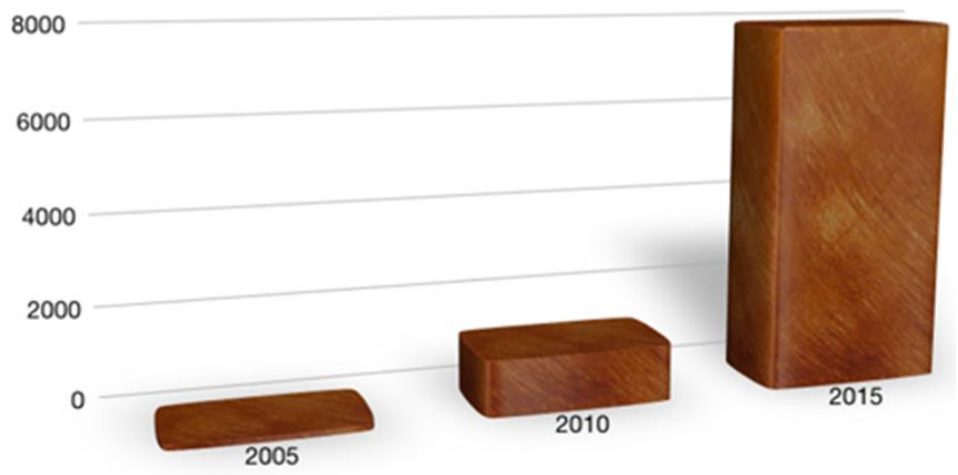


Abbildung 1: Erzeugte Datenmengen in den Jahren 2005, 2010 und 2015 (Prognose) in Exabyte [JG11].

terschiedliche Hintergrundwissen von Ersteller und Nutzer kann es leicht zu Missverständnissen kommen.
Im Zuge dieser Diplomarbeit wird ein Konzept vorgestellt, bei dem der Nutzer Visualisierungen interaktiv an seine Bedürfnisse anpassen kann. Dieses Konzept soll ermöglichen, dass Missverständnisse zwischen Erstellern und Benutzern zukünftig vermieden werden können. Darüber hinaus kann dieses Konzept ein Schritt hin zur automatischen Generierung von Visualisierungen sein.

1.2. Aufbau

In der vorliegenden Diplomarbeit werden in Kapitel 2 die zum Verständnis notwendigen Grundlagen über das Semantic Web und die Verwendung von Ontologien, Visualisierungen, Human-Computer Interaction sowie den Einsatz von Vektorgrafiken gelegt.
In Kapitel 3 werden die Aufgabenstellung dieser Diplomarbeit erklärt und der gewählte Lösungsansatz, mit dem die Aufgabenstellung gelöst werden soll, erläutert.
Anschließend wird in Kapitel 4 ein Konzept zur interaktiven, verständnisorientierten Optimierung von Visualisierungen vorgestellt. Teil dieses Konzeptes ist die Annotation von Visualisierungen mit semantischen Informationen, sowie ein Modell zum Austausch einzelner grafischer Elemente oder des gesamten Visualisierungskonzeptes einer Visualisierung. In Kapitel 5 wird ein Kernbereich des Konzeptes beschrieben, der später als Prototyp umgesetzt werden soll. Dieser Prototyp wird in Kapitel 6 vorgestellt. Zuerst wird die Software-Architektur und anschließend der eigentliche Prototyp beschrieben. In Kapitel 7 wird das entworfene Konzept und dessen Umsetzung durch den Prototypen evaluiert. Im abschließenden Kapitel 8 wird ein Fazit gezogen und ein Ausblick darüber gegeben, welche zukünftigen Entwicklungen durch diese Diplomarbeit ermöglicht werden.

2. Grundlagen

In diesem Kapitel wird die Erarbeitung der für diese Diplomarbeit notwendigen Grundlagen beschrieben. In Abschnitt 2.1 wird erklärt, was Ontologien sind und wofür diese verwendet werden sollen. Abschnitt 2.2 gibt einen Überblick über das Semantic Web. Abschnitt 2.3 beschreibt verschiedene Ontologiesprachen und Abschnitt 2.4 gibt einen Überblick über das Erstellen von Ontologien. Der darauffolgende Abschnitt 2.5 beschäftigt sich mit Visualisierungen und beschreibt die Visualisierungspipeline. Abschnitt 2.6 untersucht die Human-Computer Interaction (HCI). Es wird ein Überblick über die Entwicklung der HCI gegeben. Anschließend werden die von Norman geprägten Begriffe *Gulf of Execution* und *Gulf of Evaluation* erläutert, sowie Normans Erkenntnisse über mentale Modelle beschrieben. Abschnitt 2.7 gibt einen Überblick über Vektorgrafik und das SVG-Format. Im abschließenden Abschnitt 2.8 werden die im Verlauf der Diplomarbeit verwendeten Technologien aufgelistet.

2.1. Ontologie

Der Begriff Ontologie stammt aus dem Griechischen und lässt sich bis zu Aristoteles zurückführen. In der Philosophie ist es eine Bezeichnung für das Teilgebiet der Metaphysik, das sich mit dem Sein beschäftigt. Der Duden bezeichnet es als die „Lehre vom Sein“ [httb]. Dabei werden Beziehungen zwischen verschiedenen „Entitäten“ diskutiert. In der Informatik handelt es sich hierbei um „an explicit specification of a conceptualization“ , wie Gruber [Gru93] es definiert. In der Informatik wird, ebenso wie in der Philosophie, durch die Ontologie Wissen über die reale Welt beschrieben. Eine Ontologie ist immer auf eine bestimmte Domäne, also auf einen bestimmten Fachbereich oder einen bestimmten Anwendungsfall, beschränkt. Es werden Beziehungen zwischen verschiedenen Elementen dargestellt. Eine Ontologie ist, im Gegensatz zu einer Taxonomie, nicht nur ein Baum, der hierarchische Beziehungen beschreibt, sondern ein Netz von Verbindungen. Wenn im Folgenden von Ontologien die Rede ist, so handelt es sich immer um Ontologien im Sinne der Informatik.
Ontologien sollten sowohl von Menschen als auch von Maschinen lesbar sein. Um maschinenlesbar zu sein, müssen sie formal und explizit definiert sein. Leider ist die natürliche Sprache, mit der Menschen kommunizieren, selten formal und sehr häufig implizit. Bekannte formale Sprachen, die sowohl von Menschen als auch von Maschinen verstanden werden, sind Programmiersprachen. Allerdings muss in der Ontologie zusätzlich die Bedeutung der beschriebenen Ressourcen enthalten sein. Durch die enthaltene Bedeutung in der Ontologie können logische Schlüsse gezogen werden und Widersprüche erkannt werden. Eine Ontologie soll nicht nur ein für ein bestimmtes Projekt entworfenes Datenmodell sein, sondern allgemeingültig.
Eine der bekanntesten Ontologien ist WordNet[1]. Es handelt sich um eine lexikalische Datenbank, die versucht Beziehungen zwischen Wörtern und deren Bedeutung herzustellen. Basis von WordNet ist eine Matrix, die jedem Begriff verschiedene Bedeutungen

[1] `http://wordnet.princeton.edu/`

zuordnet. Dadurch können zum Beispiel Synonyme gefunden werden [Stu09].
Ein weiteres bekanntes Beispiel für eine Ontologie ist das Unified Medical Language System (UMLS)[2]. UMLS wurde durch die United Stateds National Library of Medicine entwickelt. Das Ziel von UMLS ist es, verschiedene Online-Datenbanken und medizinische Bücher einander anzugleichen und in einer einheitlichen Datenbank zur Verfügung zu stellen. Das dadurch entstehende Modell soll bei der Analyse medizinischer Fachtexte helfen. UMLS besteht unter anderem aus dem UMLS-Metathesaurus und dem UMLS-Semantic Network. Der UMLS-Metathesaurus dient der Zuordnung von Begriffen aus verschiedenen Terminologien zu einheitlichen Konzepten. Das UMLS-Semantic Network stellt ein Netz dar, das Relationen zwischen verschiedenen medizinischen Konzepten abbildet. Die Begriffe des Metathesaurus sind den Konzepten des Netzes zugeordnet [Stu09].

2.2. Semantic Web

Das heutige World Wide Web (Web) besteht aus einer riesigen Menge von elektronischen Dokumenten. Diese Dokumente sind durch sogenannte Hyperlinks miteinander verbunden. Das Auffinden von Dokumenten im Web ist für menschliche Anwender durch diese Hyperlinks oder durch Suchmaschinen möglich. Da über die verschiedenen Dokumente keinerlei semantische Informationen vorliegen, können heutige Suchmaschinen nur stichwortbasiert suchen und die Dokumente sind nicht maschinenlesbar. Das Web hat eine sehr heterogene Struktur. Es werden verschiedene Kodierungstechniken und Dateiformate verwendet, es ist in verschiedenen natürlichen Sprachen verfasst und die Struktur und der Aufbau der einzelnen Homepages sind sehr unterschiedlich. Durch diese Heterogenität ist es oft auch für Menschen schwierig, die gesuchten Informationen zu finden. Durch die fehlende Eindeutigkeit natürlicher Sprachen ist die Maschinenlesbarkeit zusätzlich erschwert. So wird die Suche nach dem Begriff „Bank“ Ergebnisse für Sitzgelegenheiten und für Geldinstitute liefern [PH08a]. Schwierig ist die aktuelle, textbasierte Suche auch dann, wenn der Suchende nur eine ungefähre Vorstellung vom Ziel seiner Suche hat und sie möglicherweise gar nicht in Worte fassen kann [Mar06]. Ein weiteres Problem bei der Verarbeitung des Webs durch automatische Prozesse ist, dass Homepages dahingehend optimiert sind, dass sie durch menschliche Benutzer interpretiert und verstanden werden können. Es werden oft Bilder und Links zu weiterführenden Informationen eingesetzt, die von Menschen interpretiert und verstanden werden können, aber für einen „Software Agenten“ nicht zu verstehen sind. Ein „Software Agent“ ist ein Programm, das zu einem (teilweise) autonomen Verhalten in der Lage ist [Hor07].

Ein Ansatz zur Lösung dieser Probleme wäre es, mit Hilfe von künstlicher Intelligenz und maschineller Sprachverarbeitung Computer in die Lage zu versetzen, natürlichsprachige Texte zu verstehen. Dadurch wären Computerprogramme in der Lage, aus dem Kontext heraus eine Unterscheidung zwischen der oben erwähnten Sitzgelegen-

[2] http://www.nlm.nih.gov/research/umls/

heit und dem Geldinstitut zu machen. Da das Verständnis natürlicher Sprache jedoch ein sehr komplexes Problem ist, müsste noch sehr viel Forschung betrieben werden, um diesen Ansatz umzusetzen. Um grafische Zusammenhänge zu erkennen, müsste auch die maschinelle Bildverarbeitung noch deutlich verbessert werden.

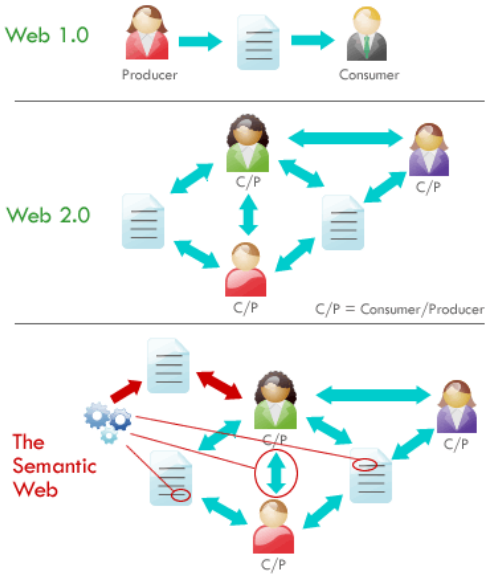


Abbildung 2: Im Web 1.0 gab es eine klare Trennung zwischen Produzent und Konsument (oben), seit dem Web 2.0 kann jeder Produzent und Konsument sein (Mitte), im Semantic Web können Programme Informationen aufbereiten (unten) [wcw].

Abbildung 2 gibt einen Überblick über die Entwicklung des Internets. Zu Beginn war das Internet eine Einbahnstraße. Die Internetseiten wurden von wenigen Anbietern erstellt und von vielen Konsumenten abgerufen. Durch die Weiterentwicklung verschiedener Technologien kann mittlerweile jeder Konsument genauso auch zum Anbieter werden. Diese sogenannten Web 2.0 Technologien haben sozialen Netzwerken, Blogs und Wikis zum Durchbruch verholfen. Das Semantic Web soll nun zusätzlich Möglichkeiten schaffen, dass Computerprogramme dem Konsumenten die gewünschten Informationen von verschiedenen Quellen zusammengefasst präsentieren. Ein Beispiel dafür ist die semantische Suchmaschine Wolfram Alpha[3]. Diese Suchmaschine ist prinzipiell in der Lage, Fragen in natürlicher Sprache zu beantworten. So erhält man auf die Frage „How old is Angela Merkel“ das Alter der deutschen Bundeskanzlerin auf den Tag genau. Die Sprachverarbeitung funktioniert allerdings nur in wenigen Fällen. So wird zum Beispiel die Frage nach der Körpergröße von Angela Merkel nicht verstanden. Wolfram Alpha erkennt aber, dass der Suchende sich für Angela Merkel interessiert und liefert, im Gegensatz zu Google oder Bing, keine Liste von Links, sondern ein kurzes Dossier.

[3] `http://www.wolframalpha.com/`

Die Verarbeitung der Anfragen dauert etwas länger als bei herkömmlichen Suchmaschinen, allerdings spart sich der Suchende die Auswahl, auf welchen Link er klicken muss und das Laden der verlinkten Seite.

Das Semantic Web ist eine Vision von Tim Berners-Lee, dem Erfinder des Webs. Diese Vision verfolgt, im Gegensatz zum oben beschriebenen Konzept, nicht den Ansatz, dass Computer die bestehenden Homepages besser verstehen können. Die Idee hinter dem Semantic Web ist, dass die Daten mit semantischen Informationen angereichert werden. Das folgende Zitat aus dem Jahre 2001 zeigt, dass das Semantic Web kein Ersatz für das Web sein soll, sondern eine Erweiterung.

> „The Semantic Web is not a separate Web, but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.“ [TBL01]

In dieser Vision ist das Web nicht nur eine Menge von verlinkten Dokumenten, sondern eine Informationsdatenbank, in der die Daten so gespeichert sind, dass sie durch Maschinen „verstanden“ oder zumindest sinnvoll verarbeitet werden können. Dazu ist es nötig, die Maschinenlesbarkeit in einer Weise zu verbessern, dass sie menschlichen Benutzern das Arbeiten mit dem Web erleichtert. Die Idee ist, die benötigten Informationen in einer Art und Weise zur Verfügung zu stellen, dass sie durch Maschinen verarbeitet werden können. Um diese Vision zu verwirklichen, sind als Erstes einheitliche und offene Standards notwendig. Das *World Wide Web Consortium* [w3c] (W3C) hat es sich zur Aufgabe gemacht, diese Standards zu erarbeiten, zu definieren und zu veröffentlichen. Die Standards RDF, RDFS und OWL (siehe Abschnitt 2.3) wurden bereits ausformuliert.
Eine wichtige Komponente zur Entwicklung des Semantic Web sind die oben eingeführten Ontologien. Durch die in den Ontologien enthaltenen Bedeutungen können logische Schlüsse gezogen werden. Dadurch können Informationen, die nicht explizit in Dokumenten enthalten sind, erkannt werden. So könnte zum Beispiel durch die beiden Informationsschnipsel, dass in Deutschland der Euro die offizielle Währung ist, und dass der Euro nur in Ländern innerhalb von Europa offizielle Währung ist, geschlossen werden, dass Deutschland in Europa liegt. Dieser Schluss ist möglich, ohne dass die explizite Information „Deutschland liegt in Europa“ vorliegt.
Abbildung 3 zeigt die unterschiedlichen Ansätze von Web und Semantic Web. Um im Web die Qualität von Suchergebnissen zu verbessern, müssen Verbesserungen auf Seiten der anfragenden Server durchgeführt werden, zum Beispiel bessere maschinelle Sprachverarbeitung oder Verteilung der Anfragen auf mehrere Server. Beim Semantic Web erfolgen die Verbesserungen auf Seite der vorhandenen Dokumente. Da die Dokumente mit semantischen Metainformationen angereichert sind, ist keine maschinelle Sprachverarbeitung natürlicher Sprache nötig. Dadurch sinkt der Aufwand einer Suchanfrage.

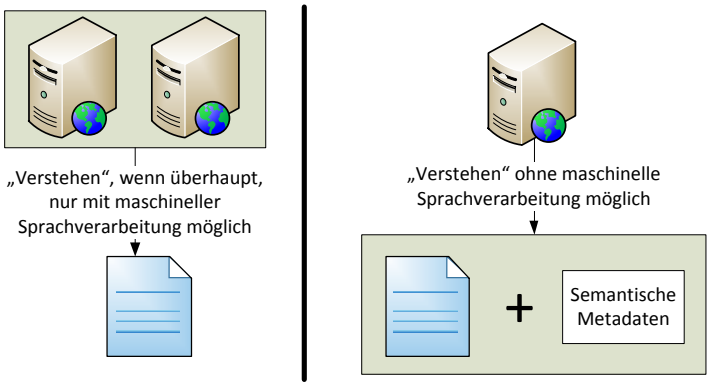


Abbildung 3: Im Web (links) liegt der Fokus für Verbesserungen auf Seite der anfragenden Server, im Semantic Web (rechts) auf Seite der Dokumente.

2.3. Ontologiesprachen

Es existieren viele verschiedene Sprachen, mit denen Ontologien beschrieben und entworfen werden können. Es folgt eine Übersicht über vorhandene Ontologiesprachen.

2.3.1. Simple HTML Ontology Extension

Die Sprache Simple HTML Ontology Extension (SHOE) wurde an der University of Maryland 1996 entwickelt. SHOE ist eine Erweiterung von HTML und wurde entwickelt, um Web-Dokumente leichter mit maschinenlesbaren semantischen Informationen anreichern zu können. Diese Annotationen sollten verwendet werden, um Suchmechanismen im Web zu verbessern. Da die von SHOE verwendeten Tags nicht in der HTML-Spezifikation vorkommen, hat das Einbetten von Informationen keinen Einfluss auf das Aussehen der Web-Dokumente. SHOE spielt heutzutage keine Rolle mehr, die Entwicklung wurde eingestellt. Die neueste Version der Spezifikation[4] ist aus dem Jahr 2000 [GPCFL04].

2.3.2. Ontology Inference Layer/Language

Ontology Inference Layer oder auch Ontology Inference Language (OIL) wurde im Rahmen des Schwerpunkts „Technologien der Informationsgesellschaft“ des Forschungsrahmenprogramms der Europäischen Union entwickelt und wurde im Jahr 2000 veröffentlicht. OIL wurde, ebenso wie SHOE, konzipiert um Web-Dokumente mit semantischen Informationen anzureichern. OIL ging bereits ein Jahr nach der Veröffentlichung in DAML + OIL auf und wurde nicht mehr eigenständig weiterentwickelt.

[4] `http://www.cs.umd.edu/projects/plus/SHOE/spec.html`

2.3.3. DAML + OIL

DAML + OIL wurde in einem gemeinsamen Projekt in den USA und der Europäischen Union entwickelt. DAML steht für DARPA Agent Markup Language, wobei DARPA wiederum für Defense Advanced Research Projects Agency (eine Behörde des Verteidigungsministeriums der USA) steht. Ziel war es, eine Sprache zu entwickeln, die es ermöglicht, Web-Ressourcen mit semantischen Informationen anzureichern. DAML + OIL wurde in mehreren Schritten entwickelt und ist die Zusammenführung eines DARPA-Projekts (DAML) und OIL. Die erste Version wurde im Oktober 2000 unter dem Namen DAML + ONT veröffentlicht. Im Dezember 2000 wurde bereits die zweite Version herausgegeben. Im Zuge dessen erfolgte die Umbenennung in DAML + OIL. Die endgültige Version wurde im März 2001 veröffentlicht und behob einige in den vorherigen Versionen aufgetretenen Fehler. Die Syntax von DAML + OIL ist XML-basiert. Da DAML + OIL in OWL aufgegangen ist, und heute keine Rolle mehr spielt, wird an dieser Stelle nicht weiter darauf eingegangen.

2.3.4. Resource Description Framework

Das Resource Description Framework (RDF) ist eine durch das W3C standardisierte Sprache zur Beschreibung von Objekten und deren Beziehungen. Die beschriebenen Objekte werden als Ressourcen bezeichnet. Jede dieser Ressourcen wird durch einen eindeutigen Bezeichner (Uniform Resource Identifier, kurz URI) beschrieben. Wichtig ist, dass die Ressourcen eindeutig identifiziert werden können, damit Missverständnisse verhindert werden. Beispielsweise könnte es bei einer Ontologie, die wissenschaftliche Publikationen jeweils Autor und Verlag zuweist, zu Verwechslungen kommen. Die Ressource „Springer-Verlag“ könnte sowohl für den Medienkonzern und Herausgeber einer großen Boulervardzeitung als auch für den wissenschaftlichen Verlag stehen.
Ursprünglich wurde RDF zur Beschreibung von Metadaten im Web entwickelt. Es war für die Beschreibung von Informationen über Homepages, wie zum Beispiel Autor oder Copyright gedacht. Später wurde die Vision des Semantic Web erweitert. Als Ressource kann heute jede mit einer URI bezeichnete Entität gesehen werden, auch wenn diese außerhalb des Webs liegt. Nach dieser Erweiterung des Semantic Webs auf Bereiche außerhalb des Webs wurde die Spezifikation von RDF noch einmal überarbeitet und 2004 neu veröffentlicht [PH08a].
Ein RDF-Dokument beschreibt einen gerichteten Graphen, der in Pfeilrichtung gelesen wird. Der Graph wird durch eine Menge von Tripeln beschrieben. Ein Tripel besteht aus zwei Knoten und einer die beiden verbindenden Kante und entspricht einer binären Relation zwischen den beiden Knoten. Das Tripel ist dabei wie ein Satz aufgebaut und besteht aus Subjekt, Prädikat und Objekt. Wie in Abbildung 4 zu sehen, werden Knoten und Kanten jeweils mit einer URI beschriftet. Auch wenn die URIs nicht online aufgerufen werden können (wie zum Beispiel in Abbildung 4), so werden sie in der Regel trotzdem in der Struktur einer online abrufbaren Adresse angegeben. Der Grund dafür ist, dass RDF einige aus XML stammende Mechanismen verwendet, mit denen URIs möglichst effizient gespeichert werden können (siehe Listing 1).

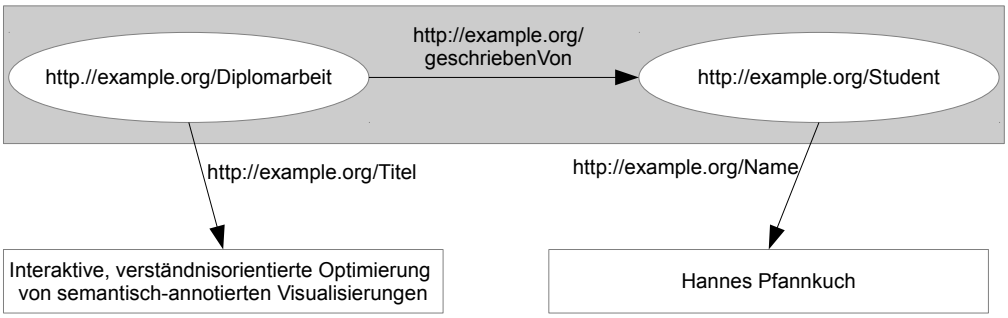


Abbildung 4: Zu sehen ist ein RDF-Graph, der aussagt, dass die Diplomarbeit (links oben) mit dem Titel 'Interaktive...' (links unten) von einem Studenten (rechts oben) mit dem Namen 'Hannes Pfannkuch' (rechts unten) geschrieben wurde.

Die ellipsenförmigen Knoten sind Ressourcen, die Subjekt und Prädikat in verschiedenen Relationen sein können. Wenn Datenwerte, zum Beispiel Text oder Zahlen, dargestellt werden sollen, so werden sie als rechteckige Knoten dargestellt. Diese rechteckigen Knoten heißen Literale und können nur als Objekt und nicht als Subjekt in Relationen verwendet werden. Literale können mit einem Datentyp versehen werden. Literale ohne Datentypen werden immer als Zeichenkette interpretiert.
Die Bedeutung eines RDF-Graphen ist für Menschen leicht zu verstehen. So sagt der Graph in Abbildung 4 aus, dass die Diplomarbeit mit dem Titel „Interaktive, verständnisorientierte Optimierung von semantisch-annotierten Visualisierungen“ von einem Studenten mit dem Namen „Hannes Pfannkuch“ geschrieben wird. RDF-Graphen können in verschiedenen Sprachen serialisiert werden. Am häufigsten wird die XML-Serialisierung verwendet, da es in vielen Programmiersprachen Bibliotheken gibt, mit denen XML verarbeitet werden kann.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:ex ="http://example.org/">
4    <rdf:Description rdf:about="http://example.org/Diplomarbeit">
5      <ex:geschriebenVon>
6        <rdf:Description rdf:about="http://example.org/Student">
7        </rdf:Description>
8      </ex:geschriebenVon>
9    </rdf:Description>
10 <rdf:RDF>
```

Listing 1: XML-Serialisierung des oberen, grau hinterlegten Teils des Graphen in Abbildung 4.

Listing 1 zeigt die XML-Serialisierung des oberen, grau hinterlegten Teils des Graphen aus Abbildung 4. In Zeile 1 ist eine optionale XML-Deklaration zu sehen, in der XML-Version und Zeichenkodierung angegeben werden. Anschließend beginnt das Doku-

ment mit einem Knoten `rdf:RDF`, der in der Regel als Wurzel eines RDF-Dokuments verwendet wird. Hier werden die beiden Namensräume `rdf:` und `ex:` und ihre jeweilige Abkürzung definiert. Alle Elemente mit einer besonderen Bedeutung in RDF werden also an dem Präfix `rdf:` zu erkennen sein, alle Elemente, die auf fiktive Begriffe unter `http://example.org` verweisen am Präfix `ex:`. Ab Zeile 4 wird das einzige Tripel beschrieben. Subjekt und Objekt werden durch Elemente vom Typ `rdf:Description` beschrieben, wobei `rdf:About` jeweils den Bezeichner angibt. Das Prädikat des Tripels wird als das Element `ex:geschriebenVon` angegeben.
Ein RDF-Graph, solange er klein genug ist, kann zwar von Menschen schnell verstanden werden, von Maschinen jedoch nicht. Die Bedeutung des Prädikats „geschriebenVon“ ist für einen Menschen, der der deutschen Sprache mächtig ist, klar, auch wenn es nicht in einer für Computer verständlichen Sprache formalisiert wurde. Aus diesem Grund wurde eine Sprache benötigt, mit der die nötigen Metainformationen definiert werden können. Diese Sprache wird im folgenden Abschnitt beschrieben.

2.3.5. Resource Description Framework Schema

Mit RDF ist es möglich, Beziehungen zwischen verschiedenen Entitäten zu beschreiben. Die verwendete Menge an Entitäten und Beziehungen bezeichnet man als Vokabular. Beim Erstellen des Vokabulars ist es für denjenigen Menschen, der das Vokabular erstellt, eindeutig, was es zu bedeuten hat. Bei unserem Beispiel in Abbildung 4 soll es klar sein, dass ein Student ein Mensch ist, der an einer Universität eingeschrieben ist und dass eine Universität eine Institution ist, die der Forschung und Lehre dient. Für andere Menschen könnte ein Student aber genauso gut ein Mensch sein, der an einer Fachhochschule eingeschrieben ist. Schon für Menschen ist die Bedeutung eines RDF-Graphen also nicht unbedingt eindeutig. Für einen Computer ist es unmöglich, eine Bedeutung zu erkennen, da alle Bezeichner einfach nur Zeichenketten sind. Die Bedeutung der Zeichenketten muss also in irgendeiner Form formal und explizit festgehalten werden. Ohne diese Bedeutung kann kein Computerprogramm logische Schlüsse ziehen.

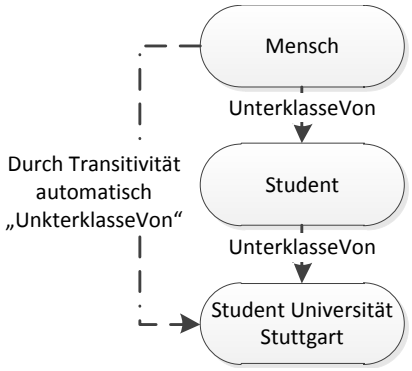


Abbildung 5: Da die Unterklassendefinition in RDFS transitiv ist, ist die Klasse ’Student Universität Stuttgart’ automatisch Unterklasse von ’Mensch’.

Mit dem Resource Description Framework Schema (RDFS) können die zum Verständnis benötigten Informationen beschrieben werden. Dadurch ist es möglich mit RDFS Ontologien zu erstellen. Es können sowohl Informationen über die Ressourcen als auch über die Beziehungen zwischen den Ressourcen gespeichert werden. RDFS unterstützt die Definition von Klassen und Unterklassen. So könnte man die Klasse 'Mensch' definieren und anschließend die Unterklasse 'Student'. Damit ist klar, dass jeder Student auch ein Mensch ist. Die Definition von Klassen und Unterklassen ist transitiv. Wenn man also die Klasse 'Student Universität Stuttgart' als Unterklasse von 'Student' definiert, dann ist jeder dieser Studenten an der Universität Stuttgart automatisch auch ein Mensch (siehe Abbildung 5). Die XML-Serialisierung des Graphen aus Abbildung 5 ist in Listing 2 zu sehen. Es beginnt in Zeile 1 mit dem `rdf:RDF` - Element, in dem die drei Namensräume `rdf:`, `rdfs` und `ex:` definiert werden. In den folgenden Elementen vom Typ `rdfs:Class` wird zuerst die Klasse 'Human' und anschließend die Unterklassen 'Student' und 'StudentAtUniversityOfStuttgart' definiert. Für jede Klasse wird zusätzlich ein `rdfs:label` für die Sprache "de", also Deutsch, definiert, das in Abbildung 5 verwendet wird.

```
1  <rdf:RDF
2    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4    xmlns:ex="http://www.example.de/">
5    <rdfs:Class rdf:about="&ex;Human">
6      <rdfs:label xml:lang="de">Mensch</rdfs:label>
7    </rdfs:Class>
8    <rdfs:Class rdf:about="&ex;Student">
9      <rdfs:label xml:lang="de">Student</rdfs:label>
10     <rdfs:subClassOf rdfs:resource="&ex:Human"/>
11   </rdfs:Class>
12   <rdfs:Class rdf:about="&ex;StudentAtUniversityOfStuttgart">
13     <rdfs:label xml:lang="de">Studen Universiät Stuttgart</rdfs:label>
14     <rdfs:subClassOf rdfs:resource="&ex:Human"/>
15   </rdfs:Class>
16 </rdf:RDF>
```

Listing 2: XML-Serialisierung von Abbildung 5

Eine Beziehungen zwischen Ressourcen wird in RDFS als *Property* bezeichnet. Für Properties lassen sich Definitions- und Wertebereich festlegen. Mit `domain` lässt sich der Definitionsbereich des Propertys, also die Klasse aller möglichen Subjekte festlegen, mit `range` der Wertebereich, also die Klasse aller möglichen Objekte. In Abbildung 6 wird somit festgelegt, dass das Property 'geschriebenVon' nur Diplomarbeiten als Subjekt und nur Studenten als Objekt haben kann.

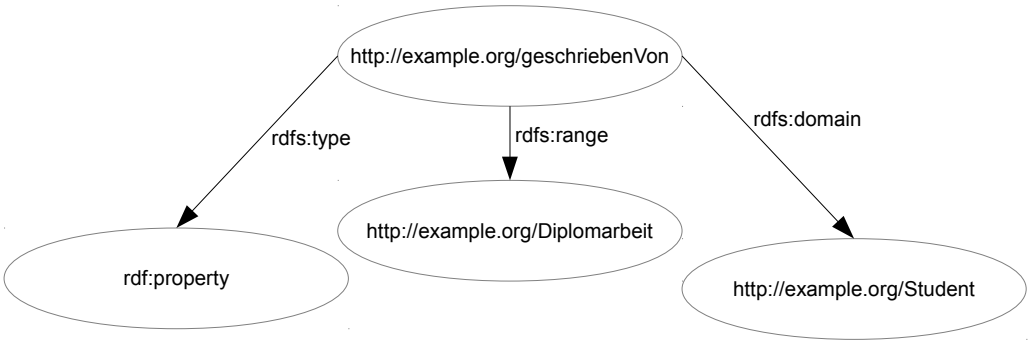


Abbildung 6: Zu sehen ist ein RDFS-Graph, der aussagt, dass es sich bei 'geschriebenVon' um ein Property handelt, dessen Subjekte vom Typ 'Diplomarbeit' und dessen Objekte vom Typ 'Student' sein müssen.

2.3.6. Web Ontology Language

Die Modellierungsfähigkeit von RDFS unterliegt gewissen Einschränkungen. So ist es beispielsweise nicht möglich zu modellieren, dass etwas *nicht* gilt, oder dass bestimmte Klassen keine gemeinsamen Elemente enthalten dürfen. Aus diesem Grund veröffentlichte das W3C im Jahre 2004 die Spezifikation der Web Ontology Language (OWL)[Con09]. OWL basiert auf der Prädikatenlogik erster Ordnung. Das Ziel von OWL war es, eine Sprache zu schaffen, die möglichst ausdrucksstark, andererseits aber in akzeptabler Zeit entscheidbar ist. OWL ist eine Erweiterung von RDF(S). Es gibt zusätzliche, vordefinierte Relationen mit einer festgelegten Bedeutung. Es existieren zwei verschiedenen Syntaxen. Die eine wird als abstrakte OWL-Syntax bezeichnet und ist für Menschen relativ einfach zu lesen. Die andere basiert auf RDF und wird als OWL-RDF-Syntax bezeichnet. Dokumente in letzterer Syntax sind immer gültige RDF-Dokumente. Um ein gutes Gleichgewicht zwischen Ausdrucksstärke und Entscheidbarkeit zu schaffen gibt es drei verschiedene Versionen von OWL, OWL Full, OWL DL und OWL Lite.

OWL Full OWL Full erlaubt sämtliche OWL Sprachelemente. Ebenso sind alle Sprachelemente aus RDF(S) erlaubt. Es gibt im Prinzip nur eine Einschränkung: es muss sich um gültige RDF-Syntax handeln. Da RDF(S) zu viele Freiheiten bei der Modellierung lässt, ist OWL Full nicht entscheidbar. Es sind aber auch prädikatenlogische Ausdrücke höherer Ordnung möglich. Durch die Unentscheidbarkeit ist die logische Schlussfolgerung durch Computerprogramme nicht möglich. Aufgrund dessen wurden die beiden anderen Versionen von OWL entwickelt. Abbildung 7 gibt einen Überblick über die verschiedenen Versionen.

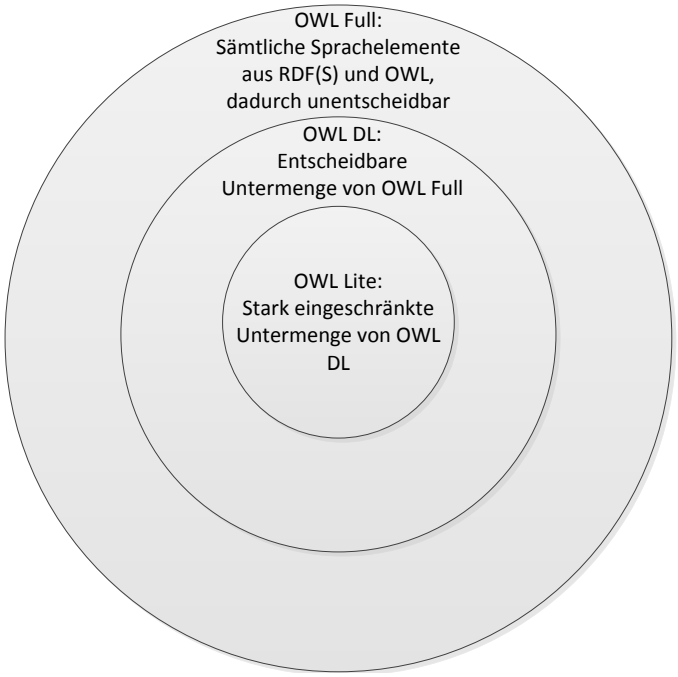


Abbildung 7: Die Sprachversion OWL Full enthält alle Sprachelemente von RDF(S) und OWL und ist unentscheidbar. OWL DL ist eine entscheidbare Untermenge von OWL Full. OWL Lite ist eine stark eingeschränkte Untermenge von OWL DL.

OWL DL OWL DL ist eine Untermenge von OWL Full. DL steht für „Description Logic“. In OWL DL sind einige Sprachkonstrukte aus OWL Full nicht erlaubt. OWL DL wurde so konstruiert, dass es sich dabei um eine entscheidbare Untermenge von OWL Full handelt. Es ist also möglich durch einen immer terminierenden Algorithmus festzustellen, ob eine Aussage aus einer Ontologie gefolgert werden kann. Es gibt verschiedene Inferenz-Tools für OWL DL.

OWL Lite OWL Lite ist eine Untermenge von OWL DL. Es enthält die wichtigsten Sprachelemente. Durch weitgehende Einschränkungen im Vergleich zu OWL DL spielt OWL Lite in der Praxis jedoch nur eine geringe Rolle [PH08b].

2.4. Ontologieerstellung

Für die Erstellung von Ontologien haben sich bisher noch keine standardisierten Vorgehensmodelle herausbilden können. Klar zu sein scheinen aber gewisse Grundsätze. So gibt es nie *den* einen richtigen Weg eine Ontologie zu erstellen. Es gibt immer Alternativen. Das Ergebnis des Erstellungsprozesses hängt stark davon ab, für welchen Zweck die Ontologie entworfen werden sollte. Es spielt auch eine große Rolle, wer die

Ontologie entworfen hat. Ein anderer Mensch, der in der Regel ein anderes Hintergrundwissen hat, hätte eine Ontologie möglicherweise komplett anders, aber deshalb nicht unbedingt schlechter entworfen. Das Erstellen einer Ontologie ist ein iterativer Prozess. Der erste Entwurf ist nur selten auch endgültig. In den meisten Fällen müssen nachträgliche Änderungen vorgenommen werden, oft auch, weil sich die Anforderungen während des Erstellungsprozesses ändern.
Unter Berücksichtigung dieser Punkte sollte klar sein, dass das im Folgenden aufgezeigte Schema nur ein Leitfaden ist und kein unantastbarer Algorithmus, der Schritt für Schritt abgearbeitet werden muss. Abbildung 8 ist ein Workflowdiagramm, das das Schema beschreibt. Die einzelnen Schritte werden im Folgenden erläutert.

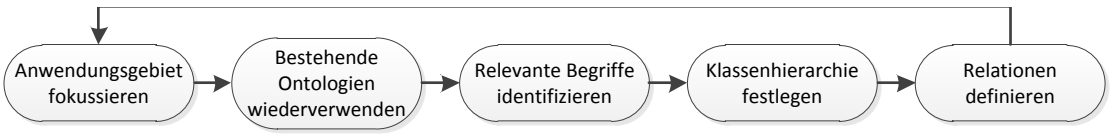


Abbildung 8: Iterativer Workflow zur Erstellung von Ontologien, detaillierte Beschreibung siehe folgender Text.

Anwendungsgebiet fokussieren
Als Erstes sollte das Anwendungsgebiet und der Umfang abgesteckt werden. Es sollte auch festgelegt werden, in welchen Bereichen wie detailliert modelliert werden soll. Dadurch wird die Ontologie zwar auf den konkreten Anwendungsfall zugeschnitten und ist nicht ohne Weiteres für andere Einsatzgebiete wiederzuverwenden. Trotzdem sollte es zur Begrenzung des Aufwandes durchgeführt werden, da man sonst Gefahr läuft, sich zu verzetteln. Ein oft verwendetes Beispiel ist die unter anderem von Stuckenschmidt [Stu09] beschriebene Erstellung einer Wein-Ontologie. Diese Ontologie kann für eine Restaurant-Software, die passende Weine zu verschiedenen Gerichten empfiehlt, verwendet werden. Dazu müssen auch die entsprechenden Speisen in der Ontologie vorkommen. Falls die Ontologie aber dafür verwendet werden soll, einen Weinbauern bei der Bewirtschaftung seiner Weinberge zu unterstützen, dann sind die passenden Speisen uninteressant. In diesem Fall sollten die passenden Anbaumethoden für die jeweiligen Weinsorten Teil der Ontologie werden. Zur Erläuterung der einzelnen Schritte wird hier die Erstellung der Wein-Ontologie für eine Restaurant-Software beschrieben.
Um Umfang und Bereich der Ontologie zu bestimmen, wird in der Literatur die Beantwortung sogenannter „Competency Questions“ empfohlen. Diese Competency Questions sollten so gestellt werden, dass sie sich (hoffentlich) mit der zu erstellenden Ontologie beantworten lassen. Mit einer möglichst vollständigen Liste sollte sich später überprüfen lassen, ob die Ontologie die gewünschten Informationen bereitstellt. In unserem Beispiel könnten die Fragen wie folgt aussehen:

- Ist Trollinger ein Rot- oder ein Weißwein?
- Passt Rotwein zu Fisch?
- Welches Essen passt zu Rosé?
- Gibt es einen bestimmten Wein nur als Flaschenwein, oder auch im offenen Ausschank?

Dies ist nur ein kleiner Ausschnitt der benötigten Fragen. Um die Ontologie später in vollem Umfang zu testen, sind deutlich mehr Fragen nötig.

Bestehende Ontologien wiederverwenden
Nachdem klar ist, welchen Bereich die Ontologie abdecken soll, ist es sinnvoll nach bereits existierenden Ontologien zu suchen, die eventuell wiederverwendet werden können. Da jedoch in der Regel keine komplette Ontologie wiederverwendet werden kann, gilt es abzuwägen, ob es sich lohnt, eine vorhandene Ontologie zu erweitern. Oft ist die Anpassung aufwendiger als eine komplette Neuentwicklung.

Relevante Begriffe identifizieren
Nach der Sammlung der Competency Questions werden nun die konkreten Begriffe gesucht, die in der Ontologie verwendet werden sollen. Als Erstes wird eine Liste der zu verwendenden Substantive erstellt. Für die Wein-Ontologie könnte ein Teil dieser Liste wie folgt aussehen: Wein, Rotwein, Weißwein, Lage, Jahrgang, Fleisch, Fisch, Dessert...
Die Sammlung dieser Begriffe kann dabei durch Interviews mit Experten auf dem entsprechenden Gebiet, durch die Analyse von Texten und Büchern des entsprechenden Gebiets oder durch die Analyse einschlägiger Datenbanken erfolgen.

Klassenhierarchie definieren
Die gesammelten Begriffe werden nun zu einer Klassenhierarchie zusammengefasst. Dabei muss untersucht werden, in welchem Verhältnis die Klassen zueinanderstehen. So ist zum Beispiel die Klasse Weißwein eine Unterklasse der Klasse Wein und eine disjunkte Schwesterklasse der Klasse Rotwein. Empfehlenswert ist hierbei ein Top-Down Vorgehen. Man beginnt mit der allgemeinsten Klasse und spezialisiert diese dann immer weiter. Bei den Weinen ist vermutlich die Klasse Wein die allgemeinste Klasse, die alle andern weinspezifischen Klassen enthält.

Relationen definieren
Anschließend müssen Eigenschaften definiert werden. So könnte man zum Beispiel die Eigenschaften passtZu und hergestelltVon definieren. Die Eigenschaften können bestimmte Einschränkungen erhalten. Eine dieser Einschränkungen ist die Kardinalität. So könnte man hergestelltVon die Kardinalität 1 zuweisen, da ein Wein immer genau von einem Weingut hergestellt wurde. Eine weitere mögliche Einschränkung ist die

Definition von Domain (Definitionsbereich) und Range (Wertebereich). So könnte man Domain von hergestelltVon auf die Klasse Wein und Range auf die Klassen Weingut und Genossenschaft beschränken.

Damit ist der erste Durchgang der Erstellung der Ontologie abgeschlossen. Da sich die Anforderungen, auch bedingt durch den Erkenntnisgewinn während der Erstellung, ändern können, müssen in der Regel mehrere Durchgänge gemacht werden. Erst dann kommt man zu einer einsatzbereiten Ontologie.
Wie die erstellte Ontologie verwendet werden kann, wird nun an einem Beispiel erklärt: Aus den drei Relationen 'Merlot istEin Rotwein', 'Rotwein passtZu dunklem Fleisch' und 'Rostbraten istEin dunkles Fleisch' kann gefolgert werden, dass Merlot zu Rostbraten passt.

2.5. Visualisierung

Unter Visualisierung versteht man die Transformation von abstrakten Daten, zum Beispiel Texte oder Messdaten, in eine grafische Repräsentation, die durch Menschen wahrgenommen und verstanden werden kann [RF94]. Visualisierungen dienen der Exploration und dem Verstehen komplexer Sachverhalte. Oft werden Visualisierungen auch zur Kommunikation mit anderen Menschen verwendet. Da die Menschheit immer mehr Daten erzeugt und speichert (wie bereits zu Beginn beschrieben, war es allein im Jahr 2010 mehr als ein Zettabyte), sind Visualisierungen ein wichtiges Werkzeug um mit diesen riesigen Datenmengen umgehen zu können. Visualisierungen werden in vielen Bereichen von Forschung, Lehre, Wirtschaft und Unterhaltungsindustrie eingesetzt.

Die sogenannte Visualisierungspipeline (Abbildung 9) repräsentiert einen Workflow, der die Überführung von Daten in grafische Darstellungen in vier Schritten beschreibt [RBH90]. In der folgenden Beschreibung sind die englischen Begriffen aus Abbildung 9 in Klammern angegeben. Der erste Schritt ist die Gewinnung der Daten (data acquisition), zum Beispiel aus Simulationsdaten, Datenbanken oder Messwerten. Die hieraus gewonnenen Rohdaten (raw data) werden anschließend durch Filtern aufbereitet. Dabei werden aus den Rohdaten die benötigten Daten ausgewählt und in das benötigte Datenformat überführt. Im Prozess des Filterns (filtering) werden die Daten durch verschiedene Verfahren, wie zum Beispiel Resampling, Interpolation oder Klassifizierung bearbeitet. Die in diesem Abschnitt der Pipeline entstandenen Daten werden als Visualisierungsdaten (visualization data) bezeichnet. Diese Visualisierungsdaten werden im nächsten Schritt, dem Mapping, auf renderbare Repräsentationen (renderable representations) abgebildet. Die Daten werden nun durch grafische Primitive (Punkte, Linien, Flächen) sowie verschiedene Attribute (Farbe, Transparenz, Textur) repräsentiert. Im abschließenden Schritt, dem Rendern (rendering), werden die Primitive zu Bildern oder Videos zusammengefasst und durch Berechnen von verschiedenen Effekten (Schatten, Ausleuchtung, Schattierungen) möglichst realistisch dargestellt.

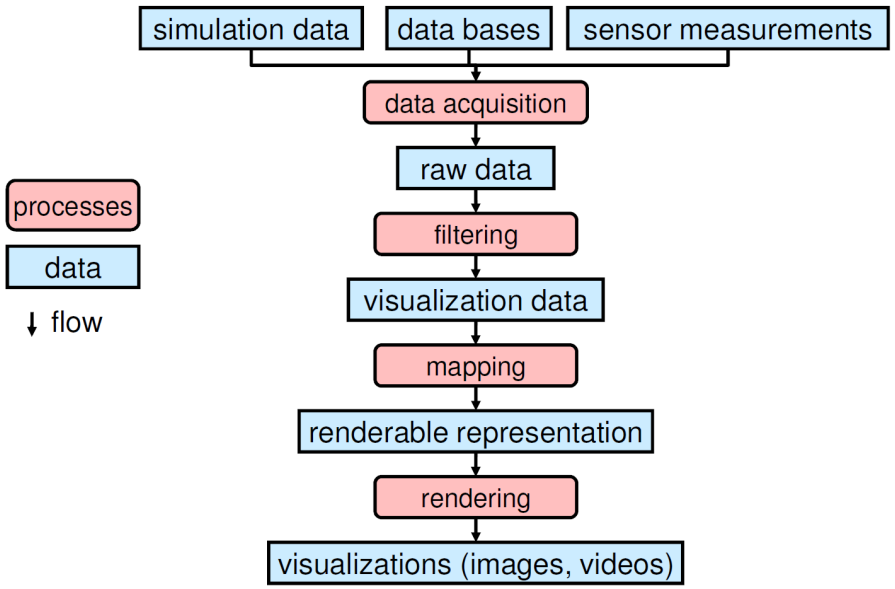


Abbildung 9: Zu sehen ist die Visualisierungspipeline. Als erstes werden die Rohdaten aus verschiedenen Quellen beschafft und durch Filtern zu den Visualisierungsdaten transformiert. Die Visualisierungsdaten werden auf renderbare Objekte abgebildet. Abschließend werden diese Objekte zu Bildern oder Videos zusammengefasst [Ert11].

Das Thema dieser Diplomarbeit ist im Bereich des Mappings angesiedelt. Die zu visualisierenden Daten werden ebenso wie die verfügbaren grafischen Elemente mit semantischen Informationen annotiert. So können die Daten auf grafische Elemente abgebildet werden.

2.6. Human-Computer Interaction

Unter Human-Computer Interaction (HCI, deutsch: Mensch-Computer-Interaktion) versteht man einen Teilbereich der Informatik, der sich mit der Interaktion zwischen Menschen und Computern befasst. Wobei unter Computern in diesem Fall nicht nur Personal Computer verstanden werden, sondern auch andere interaktive Geräte, wie beispielsweise Fahrkartenautomaten oder Smartphones. Wichtige Forschungsbereiche sind dabei die Gestaltung von Benutzeroberflächen, die Interaktion von Menschen mit bestimmten Eingabegeräten und Techniken sowie kognitive und psychologische Modelle im menschlichen Gehirn. Ziel der HCI ist es, die Interaktion zwischen Mensch und Computer zu vereinfachen und dafür zu sorgen, dass der Mensch im Mittelpunkt steht und nicht der Computer. Abschnitt 2.6.1 gibt einen Überblick über die Entwicklung der HCI, Abschnitt 2.6.2 beschreibt den Gulf of Execution und den Gulf of Evaluation. Abschnitt 2.6.3 beschreibt mentale Modelle und die daraus resultierenden Probleme im Bereich der Visualisierung.

2.6.1. Entwicklung der HCI

In den Anfangszeiten der Computerentwicklung waren Computer riesige Maschinen, die ganze Räume benötigten. Sie wurden in der Regel nur durch sehr wenige Menschen benutzt. Da die Benutzer meist Forscher oder Experten waren, spielte die HCI in dieser Zeit noch keine große Rolle. Es war kein Problem, wenn die Bedienung aufwendig und schwer zu erlernen war. Durch den Siegeszug der digitalen Technik und dem Einzug der Computer in private Haushalte wurde die HCI immer wichtiger. Und spätestens seit dem Boom mobiler Geräte wie Smartphones und Tablet-Computer sind elektronische Geräte zu allgegenwärtigen Begleitern geworden. Da die Geräte nun durch jeden verwendet werden können, ist es wichtig, dass die Bedienung intuitiv und leicht zu erlernen ist. Da digitale Geräte zunehmen den Alltag vieler Menschen bestimmen, müssen Bedienoberflächen auch ohne Erklärung verstanden werden können. So ist es beispielsweise nicht möglich, erst das Handbuch eines Fahrkartenautomaten zu lesen, bevor man sich eine Fahrkarte kauft.

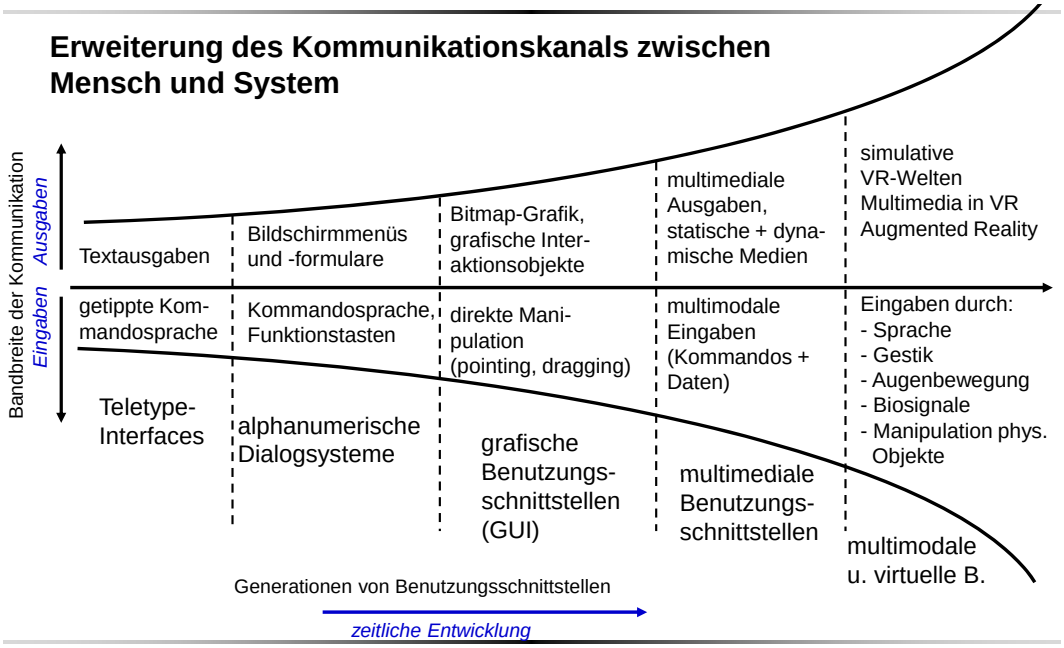


Abbildung 10: Zu sehen ist die Entwicklung der Ein- und Ausgabetechniken von Teletype-Interfaces über alphanumerische Dialogsysteme, grafische Benutzungsschnittstellen, multimediale Benutzungsschnittstellen bis hin zu multimodalen und virtuellen Benutzungsschnittstellen [Sch09].

Abbildung 10 zeigt, wie sich in den letzten 50 Jahren sowohl Eingabe- als auch Ausgabetechniken stark verändert haben: von der reinen Textausgabe und der Steuerung durch getippte Kommandos, über grafische Interaktionsobjekte und Konzeptn der direkten Manipulation, bis hin zu Virtual Reality und Steuerung durch Sprache, Gesten und in naher Zukunft möglicherweise auch durch Gedankenströme.

2.6.2. Gulf of Execution und Gulf of Evaluation

Norman [Nor86] prägte die beiden Begriffe *Gulf of Execution* und *Gulf of Evaluation*. Der Gulf of Execution beschreibt die Lücke zwischen dem Ziel des Benutzers und den am Gerät tatsächlich auszuführenden Handlungen zur Erreichung dieses Ziels. Ein einfaches Beispiel ist der Vorgang des Geldabhebens an einem Geldautomaten. Das Ziel ist eine bestimmte Summe Bargeld von einem Konto abzuheben. Die tatsächlich auszuführenden Handlungen sind jedoch aufwendiger:

1. Einführen der Bankkarte in den Automaten
2. Auswahl der Funktion 'Geld abheben'
3. Eingabe der PIN
4. Auswahl des Betrags
5. Entnahme des Geldes

Um ein einfaches, klar definiertes Ziel zu erreichen, sind also eine ganze Reihe von Handlungen und Interaktionen mit dem System notwendig. Gegenstand der HCI ist es auch, die Lücke zwischen Ziel und auszuführenden Handlungen möglichst zu verkleinern.

Der Gulf of Evaluation beschreibt die Lücke zwischen der Ausgabe eines Geräts und dem Verständnis des aktuellen Systemzustands. Es geht also um die Frage, ob das System leicht erfassbare und interpretierbare Informationen über seinen Zustand liefert. Um den Gulf of Evaluation zu überbrücken, muss der Benutzer folgende Schritte durchführen:

1. Perzeption (Wahrnehmung) des Systemzustands.
2. Interpretation des wahrgenommenen Systemzustands.
3. Auswerten, ob das Interpretierte mit den ursprünglichen Zielen übereinstimmt.

Dabei sollten die Anstrengungen des Benutzers zur Durchführung dieser Schritte durch gutes Design des Systems möglichst gering gehalten werden.
In dieser Diplomarbeit wird ein Versuch beschrieben, den Gulf of Evaluation im Bereich der Visualisierung zu verkleinern, indem der Benutzer Visualisierungen an seine persönlichen Bedürfnisse anpassen kann.

2.6.3. Mentale Modelle und daraus resultierende Probleme im Bereich der Visualisierung

Die Sicht eines Menschen auf seine Umgebung, sich selber, seine eigenen Fähigkeiten und die an ihn gestellten Aufgaben unterscheidet sich stark von Mensch zu Mensch.

Um sich die Interaktion mit der Umwelt, mit Mitmenschen oder mit verschiedenen Technologien zu erleichtern, bauen sich Menschen im Unterbewusstsein mentale Modelle. Norman hat bei seinen Experimenten folgende Beobachtungen über mentale Modelle gemacht:

Mentale Modelle...

- ...sind nicht vollständig.
- ...hängen von den Fähigkeiten des Menschen ab.
- ...sind instabil und verändern sich im Lauf der Zeit, vor allem wenn bestimmte Tätigkeiten länger nicht ausgeführt werden.
- ...haben keine festen Grenzen. Verschiedene Operationen und Geräte werden vermischt.
- ...sind nicht wissenschaftlich. Menschen legen abergläubisches Verhalten an den Tag, obwohl sie sich dessen bewusst sind.
- ...sind „geizig“. Menschen nehmen höheren physischen Aufwand als nötig in Kauf, um den mentalen Aufwand gering zu halten.

Diese Beobachtungen stellen klar, dass die verschiedenen mentalen Modelle sich voneinander unterscheiden. Dadurch lassen sich Verständnisprobleme beim Benutzen von Software erklären. Der Nutzer hat ein anderes mentales Modell als der Ersteller und versteht die Bedeutung, die transportiert werden soll, nicht. Nur in einer idealen Welt, in der alle Menschen für eine bestimmte Sache immer das selbe mentale Modell hätten, würden sich die beiden Modelle und die grafische Repräsentation der Bedeutung gleichen [Nor83].
Im Bereich Visualisierung versucht der Ersteller eine Information grafisch an den Nutzer zu übermitteln. Wie in Abbildung 11 zu erkennen, hat der Ersteller die Idee eines

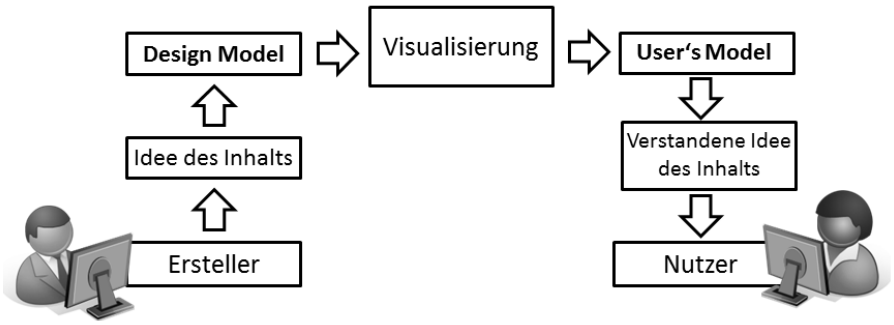


Abbildung 11: Unterschiede in den mentalen Modellen von Ersteller und Nutzer sind häufig der Grund für Verständnisprobleme bei der Betrachtung von Visualisierungen [MR11].

Inhaltes, den er vermitteln will. Dazu wählt der Ersteller ein *Design Model* und versucht dieses durch die Visualisierung darzustellen. Der Nutzer erzeugt beim Betrachten der Visualisierung sein eigenes *User Model*. Dieses Modell basiert auf seiner Interpretation der Visualisierung, wird aber auch beeinflusst durch das Hintergrundwissen und die persönlichen Vorlieben des Nutzers. Durch dieses Modell erhält der Nutzer ein Verständnis des Inhalts. Im Idealfall würden die mentalen Modelle von Ersteller und Nutzer übereinstimmen. Wie Normans Beobachtungen jedoch gezeigt haben, ist dies in der Regel nicht der Fall. Das mentale Modell des Nutzers hängt nicht nur davon ab, wer dieser Nutzer ist. Ebenso entscheiden ist der Zeitpunkt der Betrachtung und der zu diesem Zeitpunkt aktuelle Wissenszustand sowie die Erfahrung des Nutzers. Um trotzdem das mentale Modell des Nutzers möglichst gut zu treffen, sollte die Visualisierung an die Aufgabenstellung, die Anforderungen und die Fähigkeiten des Nutzers angepasst werden können. Je schlechter die beiden mentalen Modelle übereinstimmen, desto eher kommt es zu Verständnisproblemen. Im schlimmsten Fall kann der Nutzer die Visualisierung nicht verwenden.
In der vorliegenden Diplomarbeit wird ein Interaktionskonzept beschrieben, das es dem Nutzer ermöglicht, die Visualisierung interaktiv an seine Bedürfnisse anzupassen, um dadurch Verständnisprobleme zu vermeiden [MR11].

2.7. Vektorgrafik

In diesem Abschnitt werden Vektorgrafiken eingeführt. In 2.7.1 werden Vektorgrafiken allgemein und in 2.7.2 das Vektorgrafikformat Scalable Vector Graphics beschrieben.

2.7.1. Vektorgrafiken allgemein

Hinter Vektorgrafiken steht die Idee, nicht für jeden einzelnen Punkt der Grafik einen Farb- und Helligkeitswert zu spezifizieren, sondern die Grafik durch grafische Primitive zu beschreiben. So wird zum Beispiel für eine Linie nur Start- und Endpunkt sowie Linienstil, -breite und -farbe abgespeichert. Im Gegensatz dazu stehen Rastergrafiken, bei denen jeder einzelne Punkt des Bildes beschrieben ist. Wie in Abbildung 12 gut zu erkennen ist, haben Vektorgrafiken gegenüber Rastergrafiken den großen Vorteil der beliebigen und verlustfreien Skalierbarkeit. Hinzu kommt, dass vor allem bei großen Grafiken, die Vektorgrafiken häufig deutlich weniger Speicherplatz benötigen. Moderne Vektorgrafikprogramme bieten auch Funktionen, um Transparenzeffekte und Farbverläufe abzuspeichern. Dadurch lassen sich deutlich bessere und realistischere Abbildungen der realen Welt erstellen. Vektorgrafiken haben ihre Stärken bei Grafiken, die einfach durch grafische Primitive dargestellt werden können. Fotos, die als Rastergrafik aufgenommen werden, können nicht verlustfrei in Vektorgrafiken umgewandelt werden. Vektorgrafiken sind bei der Erstellung von Diagrammen und Logos sehr beliebt. Die meisten Vektorgrafikformate unterstützen auch, dass Rastergrafiken eingebunden werden können. Die eingebundenen Rastergrafiken werden wie ein Rechteck behandelt.

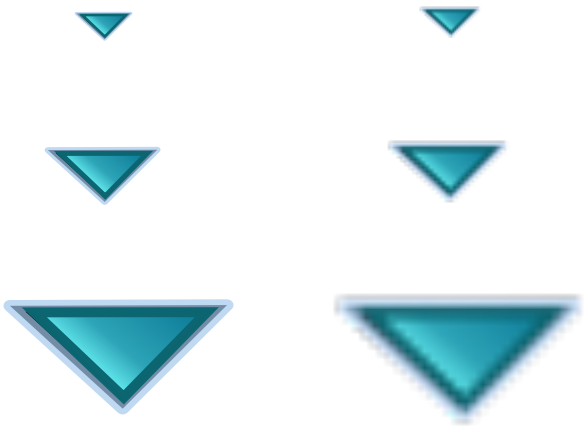


Abbildung 12: Links eine Beliebig skalierbare Vektorgrafik, rechts eine Rastergrafik, die durch das Zoomen stark verpixelt wirkt.

Der Rahmen der eingebundenen Rastergrafik wird mit der Vektorgrafik skaliert, die Rastergrafik selber kann aber nicht verlustfrei skaliert werden.
Um Vektorgrafiken auf Bildschirmen darzustellen oder auszudrucken zu können, müssen sie in eine Rastergrafik umgewandelt werden. Da dies zu einem Zeitpunkt passiert, zu dem sowohl Skalierungsfaktor als auch die Auflösung des Zielgeräts bekannt sind, kann die Vektorgrafik immer optimal dargestellt werden. Drucker haben in der Regel einen eigenen Rastergrafikprozessor, der die Vektorgrafik umrechnet. Dadurch können Vektorgrafiken in jeder beliebigen Größe ausgedruckt werden [httd].
Da bei Vektorgrafiken die einzelnen grafischen Primitive, aus denen die Grafik zusammengesetzt ist, gespeichert werden, sind Vektorgrafikformate eine gute Wahl, wenn einzelne Elemente ersetzt werden sollen.

2.7.2. Scalable Vector Graphics

Das W3C empfiehlt Scalable Vector Graphics (SVG) als Format zur Beschreibung von Vektorgrafiken. Die Spezifikation [5] wurde erstmalig im Jahr 2001 veröffentlicht. SVG basiert auf XML. Dadurch sind XML-Dokumente in einer Baumstruktur aus verschiedenen Elementen aufgebaut. Das Dokument beginnt mit der XML-Deklaration. Der SVG-Teil beginnt mit dem Tag `<svg>`. In diesem Start-Tag werden unter anderem Höhe und Breite der Grafik angegeben. Das Dokument wird mit dem Tag `</svg>` beendet. Listing 3 zeigt den grundlegenden Aufbau eines SVG-Dokuments.

[5] `http://www.w3.org/TR/2003/REC-SVG11-20030114/`

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->

<svg
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
   width="210mm"
   height="297mm"
   id="svg2"
   version="1.1"
   inkscape:version="0.48.2 r9819">
<!--Inhalt des Dokuments>
</svg>
```

Listing 3: Beispiel für eine mit Inkscape erzeugte, leere SVG-Datei

SVG unterstützt drei verschiedene Arten von grafischen Objekten:

Vektorgrafiken Diese Grafiken sind aus grafischen Primitiven zusammengesetzt.

Bilder Es können Bilder in den Formaten PNG und JPEG eingebunden werden.

Text Es kann Text in einer Schriftart, die dem Renderer zur Verfügung stehen muss, eingebunden werden.

Im Folgenden werden die in SVG verfügbaren Elemente beschrieben:

Rechteck Das Element `<rect />` definiert ein Rechteck. Es müssen die Koordinaten, sowie Höhe und Breite angegeben werden. Optional kann ein Radius für abgerundete Ecken angegeben werden.

Kreis Das Element `<circle />` definiert einen Kreis. Es müssen die Koordinaten für den Mittelpunkt und der Radius angegeben werden.

Ellipse Das Element `<ellipse />` definiert eine Ellipse. Es müssen die Koordinaten für den Mittelpunkt und die beiden Halbachsenradien angegeben werden.

Linie Das Element `<line />` definiert eine Linie. Es müssen die Koordinaten für Start- und Endpunkt angegeben werden.

Polygonzug Das Element `<polyline />` definiert einen Polygonzug. Der Polygonzug wird durch die Koordinaten beliebig vieler Punkte angegeben, die durch Linien verbunden werden.

Polygon Das Element `<polygon />` definiert ein Polygon. Wie beim Polygonzug wird das Polygon durch die Koordinaten beliebig vieler Punkte definiert, mit dem Unterschied, dass beim Polygon Start- und Endpunkt durch eine Linie verbunden werden.

Pfad Das Element `<path />` ist das mächtigste Element in SVG. Alles was durch die bisher vorgestellten Elemente dargestellt werden kann, kann auch durch das Pfad-Element dargestellt werden. Die anderen Elemente machen SVG-Dateien besser für Menschen lesbar und sparen oft Speicherplatz. Ein Pfad wird durch die Kommandos `moveto` und `lineto` definiert, wobei `moveto` einen Sprung an die angegebenen Koordinaten definiert und `lineto` eine Linie vom letzten definierten Punkt zu den angegebenen Koordinaten definiert. Zusätzlich gibt es noch das Kommando `closepath`, das den aktuellen Pfad beendet und eine direkte Linie zum Startpunkt des Pfades zeichnet.

Text Zum Einbinden und Strukturieren von Text stehen drei Elemente zur Verfügung:

- Das Element `<text />` dient zum Einbinden von Text. Es können verschiedene Attribute angegeben werden, zum Beispiel die Position und die Länge des Textes, Größe, Schriftart und Farbe.
- Mit dem Element `<tspan />` können Bereiche innerhalb eines Textes verändert werden. So können beispielsweise einzelne Wörter in anderer Farbe oder Größe dargestellt werden.
- Das Element `<tref />` bietet die Möglichkeit auf definierte Texte zu referenzieren. Dies ist vor allem dann sinnvoll, wenn der gleiche Text mehrfach verwendet werden soll.

Image Das Element `<image />` bietet die Möglichkeit Rastergrafiken in ein SVG-Dokument einzubinden. Es müssen die Koordinaten, Länge und Breite des Bildes und ein Verweis auf die externe Datei angegeben werden. Es ist nicht möglich, auf Elemente innerhalb des SVG-Dokuments zu verweisen. Mit diesem Element werden im Prototypen die grafischen Elemente in die Visualisierungen eingebunden.

Alle Elemente, sowie Gruppen von Elementen können durch affine Transformationen verändert werden. Es stehen Parallelverschiebung, Skalierung, Rotation und Scherung zur Verfügung. Diese Transformationen können entweder einzeln, unter Angabe der benötigten Parameter, oder kombiniert in einer 3x3-Matrix definiert werden.

Probleme mit SVG-Editoren Bei durch Editoren erzeugten SVG-Dokumenten tritt oft das Problem auf, dass die Dateien unnötig groß sind, da die SVG-Editoren in vielen Fällen das Pfad-Element, anstatt der verschiedenen Grundformen verwenden. So ist bei zwei vom Aussehen her identischen Kreisen, von denen einer mit dem SVG-Editor Inkscape und der andere manuell mit einem Texteditor erstellt wurde, ein Unterschied beim Speicherverbrauch von ca. Faktor 9 festzustellen.
Da SVG auf XML basiert, können eigene Elemente hinzugefügt werden. Diese Elemente werden von gängigen SVG-Renderern ignoriert. Daher eignet sich SVG gut dafür, um die Grafiken mit semantischen Annotationen anzureichern.

2.8. Verwendete Technologien

Die Ausarbeitung dieser Diplomarbeit wird mit LaTeX erstellt. Zum Erstellen der Ontologie wird der Ontologie-Editor Protégé in der Version 3.4.7, entwickelt von der *Stanford University*, verwendet. Der Prototyp wird in der Programmiersprache C# entwickelt. Als Entwicklungsumgebung wird Visual Studio 2010 Ultimate eingesetzt. Damit werden auch die Klassen- und Sequenzdiagramme erstellt.
Das im Prototyp benötigte Rendering der PNG-Dateien wird durch das Kommandozeilentool von Inkscape durchgeführt. Inkscape ist ein Open-Source-Vektorgrafikprogramm, das auf `www.inkscape.org` heruntergeladen werden kann. Die in Kapitel 5 verwendeten Mockups werden mit der Web Demo des Design-Tools Balsamiq[6] erstellt.

[6] `http://www.balsamiq.com/`

3. Aufgabenstellung und Lösungsansatz

Im folgenden Kapitel wird in Abschnitt 3.1 zunächst die Aufgabenstellung der vorliegenden Diplomarbeit beschrieben. In Abschnitt 3.2 wird anschließend der Lösungsansatz erläutert, der zur Lösung der gestellten Aufgaben ausgewählt wurde.

3.1. Aufgabenstellung

Übergeordnetes Ziel der Diplomarbeit „Interaktive, verständnisorientierte Optimierung von semantisch-annotierten Visualisierungen“ ist es, den Gulf of Evaluation im Bereich von Visualisierungen zu verkleinern. Um dieses Ziel zu erreichen, soll ein Konzept entworfen werden, mit dem Visualisierungen mit semantischen Metainformationen annotiert werden können, um sie anschließend interaktiv optimieren zu können. Um dies zu ermöglichen soll eine Grafik-Ontologie sowie ein Regelwerk zum Annotieren von grafischen Elementen in einer Visualisierung entworfen werden. Bevor die Ontologie entworfen wird, soll überprüft werden, ob im Semantic Web bereits Ressourcen, beziehungsweise Ontologien vorhanden sind, die verwendet oder erweitert werden können. Anschließend soll ein die grundlegenden Bereiche des Konzepts umfassender Kernbereich herausgearbeitet und ein Prototyp entwickelt werden, der diesen Kernbereich implementiert. Durch den Prototypen sollen Benutzer die Möglichkeit erhalten, die grafische Repräsentation visualisierter Daten individuell anpassen zu können. Es soll ebenso möglich sein, die einzelnen grafischen Elemente, als auch das gesamte Visualisierungskonzept auszutauschen. Dadurch kann der Benutzer die Visualisierung an seine persönlichen Bedürfnisse und Vorlieben anpassen und Verständnisprobleme verhindern. Der Prototyp soll es dem Benutzer ermöglichen, die Anpassungen an seiner Visualisierung mittels einer grafischen Oberfläche durchzuführen. Zum Abschluss sollen das Konzept und seine Umsetzbarkeit evaluiert werden.
Um die Aufgabenstellung zu lösen, wurden folgende Teilaufgaben identifiziert:

1. Recherche und Einarbeitung in die Themen Semantic Web und Ontologien (Abschnitte 2.1, 2.2, 2.3 und 2.4).
2. Recherche und Einarbeitung in die Themen Visualisierung, Human-Computer Interaction und Vektorgrafik (Abschnitte 2.5, 2.6 und 2.7).
3. Untersuchen, ob geeignete Ontologien aus dem Semantic Web verwendet werden können, gegebenenfalls Entwurf eigener Ontologien (Abschnitt 4.2).
4. Entwicklung eines Konzepts zur Optimierung von Visualisierungen (Kapitel 4).
5. Erarbeiten eines Kernbereichs des Konzepts, der im Prototypen umgesetzt werden soll (Kapitel 5).
6. Implementieren des Prototyps (Kapitel 6).
7. Evaluierung des Konzepts und seiner Umsetzbarkeit (Kapitel 7).

3.2. Lösungsansatz

Um diese Diplomarbeit leichter verständlich zu machen, wird ein beispielhaftes Einsatzszenario aus dem Bereich Bevölkerungsentwicklung verwendet. Damit ein möglichst breites Spektrum von Anwendungsdomänen und Visualisierungskonzepten abgedeckt wird, soll zusätzlich ein Einsatzszenario aus der Automobilbranche entworfen werden.
Es sollen zwei verschiedene Ontologien, eine Grafik-Ontologie und eine Anwendungsfall-spezifische Domänen-Ontologie, entwickelt werden. Der Ansatz, zwei Ontologien, eine global einsetzbare Grafik-Ontologie und eine Anwendungsfall-spezifische Domänen-Ontologie, zu erstellen, wurde gewählt, um mit möglichst hoher Arbeitseffizienz eine möglichst flexible Lösung zu erhalten. So kann die Grafik-Ontologie für alle Anwendungsfälle verwendet werden (Effizienz), während die Domänen-Ontologie auf den konkreten Anwendungsfall zugeschnitten ist (Flexibilität). Die Ontologien sollen in RDF erstellt werden, da dies die einzige Ontologiesprache ist, für die eine C#-Bibliothek gefunden wurde. Beim Entwerfen der Ontologien soll das in 2.4 vorgestellte Schema verwendet werden.
Im Anschluss daran soll ein Interaktionskonzept entwickelt werden, wie die erstellten Ontologien verwendet werden können, um Visualisierungen mit semantischen Metainformationen zu annotieren. Diese Visualisierungen sollen interaktiv optimiert werden können. Da es sich um verschiedene Anwendungsbereiche handelt - Erstellen und Optimieren - soll das Interaktionskonzept in zwei Teile unterteilt werden.
Die Architektur des Prototyps soll an das Model-View-Controller Paradigma angelehnt werden. Der Prototyp soll in der Programmiersprache C# implementiert werden. Da in SVG-Dateien weitere Elemente, die nicht Teil der SVG-Spezifikation sind, hinzugefügt werden können, ohne dass diese das Aussehen der Grafik verändern, wurde entschieden, die Visualisierungen in SVG zu erzeugen. Da C# das Verändern von SVG-Dateien nicht unterstützt, müssen die notwendigen Methoden selber implementiert werden. Bei der Implementierung des Prototyps soll versucht werden, möglichst auf bereits vorhandene Technologien und Bibliotheken zurückzugreifen, um den Zeitaufwand zu minimieren. Daher soll nach Möglichkeit kein eigener Renderer für SVG-Dateien geschrieben werden. Die dadurch bei der Implementierung eingesparte Zeit soll für die Erarbeitung des theoretischen Konzepts verwendet werden.

4. Konzept

Häufig werden Visualisierungen von Menschen mit technischer Ausbildung (Ingenieure, Informatiker) erstellt, aber von Menschen mit anderem Hintergrund (beispielsweise Marketingexperten oder Manager) verwendet. Oder sie werden durch Marketingexperten erstellt und sollen von Kunden verstanden werden. Durch das unterschiedliche Hintergrundwissen von Ersteller und Nutzer kann es leicht zu Missverständnissen kommen. Um diese Missverständnisse möglichst zu vermeiden, wird im Folgenden ein Konzept vorgestellt, bei dem der Benutzer einer Visualisierung diese so weit wie möglich an seine eigenen Bedürfnisse und Vorlieben anpassen kann.
Im weiteren Verlauf der Diplomarbeit werden die Begriffe Visualisierung und Visualisierungskonzept verwendet. Der Begriff 'Visualisierung' wird als eine grafische Repräsentation verstanden. Eine Visualisierung kann aus verschiedenen Visualisierungskonzepten bestehen, zwischen denen gewechselt werden kann. Ein Visualisierungskonzept ist beispielsweise eine Landkarte, eine Tree-Map oder ein Balkendiagramm.
Dabei wird bei der Erstellung der Visualisierung die Bedeutung festgelegt, die durch die Visualisierung transportiert werden soll, damit diese Bedeutung stets erhalten bleibt. Alles andere, was die Bedeutung nicht beeinträchtigt, soll durch den Benutzer austauschbar sein. Um das Konzept besser zu veranschaulichen, werden im folgenden Abschnitt zwei Einsatzszenarien als Beispiele eingeführt. Eines davon stammt aus dem Bereich Bevölkerungsentwicklung und das andere aus der Automobilbranche. Diese beiden Beispiele werden dann im weiteren Verlauf dieser Diplomarbeit verwendet.

4.1. Einsatzszenarien

Für ein besseres Verständnis des erarbeiteten Konzeptes und des Prototyps werden im Folgenden zwei Einsatzszenarien eingeführt. Durch sie soll zum Einen gezeigt werden, in welchen verschiedenen Bereichen das zu entwerfende Konzept eingesetzt werden kann. Zum Anderen soll ein möglichst breites Spektrum an Visualisierungen abgedeckt werden. Aus diesem Grund wurden zwei Einsatzszenarien aus völlig verschiedenen Anwendungsdomänen gewählt. Die beiden Szenarien werden in den folgenden beiden Unterabschnitten erläutert.

4.1.1. Einsatzszenario Bevölkerungsentwicklung

Das erste Beispiel, das auch später im Prototyp zum Einsatz kommt, ist aus der Demografieforschung. Ziel ist es, die Bevölkerungsentwicklung in Deutschland zwischen 2007 und 2009 zu visualisieren. Dabei soll neben den absoluten Zahlen auch das Wachstum, beziehungsweise der Rückgang der Bevölkerungszahlen veranschaulicht werden. Die Entwicklung soll in verschiedenen Granularitäten (zum Beispiel auf kommunaler, regionaler und Landesebene) veranschaulicht werden können. Es sollen verschiedene Visualisierungskonzepte zur Verfügung stehen. Hierfür eignen sich beispielsweise Landkarten, Balkendiagramme oder Tree-Maps.

In Abbildung 13 wird das Konzept für das Einsatzszenario Bevölkerungswachstum schematisch dargestellt. Aufgrund der Bevölkerungszahlen wird eine Visualisierung erstellt, die verschiedene Visualisierungskonzepte unterstützt. Der Benutzer kann zwischen den Konzepten wechseln und jedes Konzept durch den Austausch einzelner grafischer Elemente interaktiv optimieren.

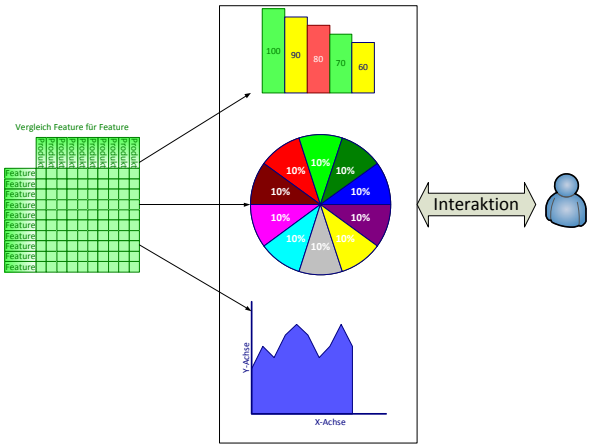


Abbildung 13: Auf Basis der Daten (links) werden die Visualisierungen (in der Mitte) erstellt. Der Benutzer (rechts) kann die Visualisierungen interaktiv anpassen.

4.1.2. Einsatzszenario Automobilvisualisierung

Das zweite Beispiel stammt aus der Automobilbranche. Ziel ist es ein Auto zu visualisieren. Die Visualisierung soll für verschiedene Aufgaben verwendet werden können. Ein Visualisierungskonzept soll eine technische Zeichnung sein, ein anderes ein fotorealistisches Bild des Autos. Dieses Abbild kann von Kunden individuell angepasst werden. Die Kunden sollen sowohl einzelne Elemente (Spiegel, Räder, Radio) als auch ganze Ausstattungslinien oder -pakete austauschen können. Durch Wechseln des Visualisierungskonzeptes kann eine technische Zeichnung des individuell angepassten Autos erstellt werden. Diese kann für die Produktion des Autos verwendet und an die Werkstatt übermittelt werden, die später dieses Auto warten und reparieren soll.

4.2. Visualisierungen interaktiv optimieren

Um die steigende Menge an Daten, die ständig erzeugt wird, sinnvoll verarbeiten und verstehen zu können, ist es in vielen Fällen nützlich, diese Daten in Form von Visualisierungen grafisch darzustellen. Da grafische Darstellungen, genau wie natürlichsprachliche Texte, häufig keine eindeutige Bedeutung haben, werden sie von verschiedenen

Menschen unterschiedlich interpretiert (siehe 2.6.3). Um die Bedeutung einer grafischen Darstellungen eindeutig zu definieren, können deren Elemente auf einer Metaebene semantisch annotiert werden. Durch diese Annotationen kann der Benutzer interaktiv und verständnisorientiert Visualisierungen anpassen. Ein Konzept, das die Annotationen und späteren Anpassungen ermöglicht, ist in Abbildung 14 schematisch dargestellt und wird im Folgenden vorgestellt.

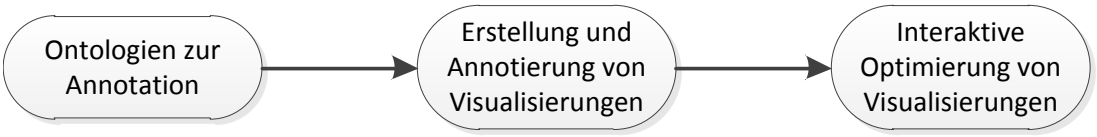


Abbildung 14: Workflow, der die Erstellung von optimierbaren Visualisierungen ermöglicht. Zuerst werden die Ontologien benötigt (links), dann werden die Visualisierungen erstellt und annotiert (Mitte), die anschließend optimiert werden können (rechts).

Abschnitt 4.2.1 beschreibt die Erstellung der zur Annotation benötigten Ontologien. In 4.2.2 wird beschrieben, wie die erstellten Ontologien bei der Erstellung von Visualisierungen verwendet werden, um diese mit semantischen Metadaten anzureichern. Abschnitt 4.2.4 schließt das Kapitel mit der Beschreibung eines Interaktionskonzepts zur interaktiven Optimierung der erstellten Visualisierungen.

4.2.1. Ontologien zur Annotation

Bevor die Visualisierungen annotiert werden können, muss zuerst geklärt werden, aus welchen Ontologien die semantischen Informationen kommen sollen. Um unnötigen Aufwand beim Erstellen von Ontologien zu vermeiden, wird zuerst versucht, bereits existierende Ontologien aus dem Semantic Web zu verwenden. Die Recherchen im Semantic Web werden mit Hilfe einer Suchmaschine durchgeführt, die das Web nach RDF-Dokumenten und HTML-Seiten mit eingebetteten Metadaten durchsucht[7]. Da auch nach aufwendigen Recherchen keine brauchbaren Ontologien zur Annotation von Visualisierungen gefunden werden konnten, wurde der Entschluss gefasst, eigene Ontologien zu entwerfen. Es werden zwei verschiedene Ontologien, eine Grafik-Ontologie und eine anwendungsfallspezifische Domänen-Ontologie, entwickelt, die beide in den nächsten Abschnitten beschrieben werden. Der Ansatz, zwei Ontologien, eine global einsetzbare Grafik-Ontologie und eine anwendungsfallspezifische Domänen-Ontologie, zu erstellen, wurde gewählt, um mit möglichst hoher Arbeitseffizienz eine möglichst flexible Lösung zu erhalten. So kann die Grafik-Ontologie für alle Anwendungsfälle verwendet werden (Effizienz), während die Domänen-Ontologie auf den konkreten Anwendungsfall zugeschnitten ist (Flexibilität).

[7] `http://www.swoogle.umbc.edu/`

Grafik-Ontologie
Die Grafik-Ontologie wird verwendet, um verschiedene grafische Primitive und die daraus aufgebauten Visualisierungskonzepte zu beschreiben. Die Ontologie enthält zu diesem Zweck die Visualisierungskonzepte und Informationen zu atomaren grafischen Bausteinen (im Folgenden als 'grafische Primitive' bezeichnet), aus denen Visualisierungen in den verschiedenen Konzepten bestehen können. Grafische Primitive sind die kleinsten sinnvoll zusammengehörenden grafischen Bausteine einer Visualisierung. Theoretisch könnten alle Elemente letztendlich bis auf Pixelebene zerlegt werden. Dies ist aber aus zwei Gründen nicht sinnvoll. Da die Benutzer auch Menschen ohne technische Ausbildung sein können, würde die große Anzahl an Pixeln ein effizientes Arbeiten unmöglich machen. Außerdem sollen nicht einzelne Pixel ausgetauscht werden, sondern Pixelmengen die zu logisch zusammengehörigen Objekten gehören. Aus diesem Grund werden die Visualisierungen in geometrische Objekte wie zum Beispiel Kreise, Rechtecke oder Linien zerlegt.

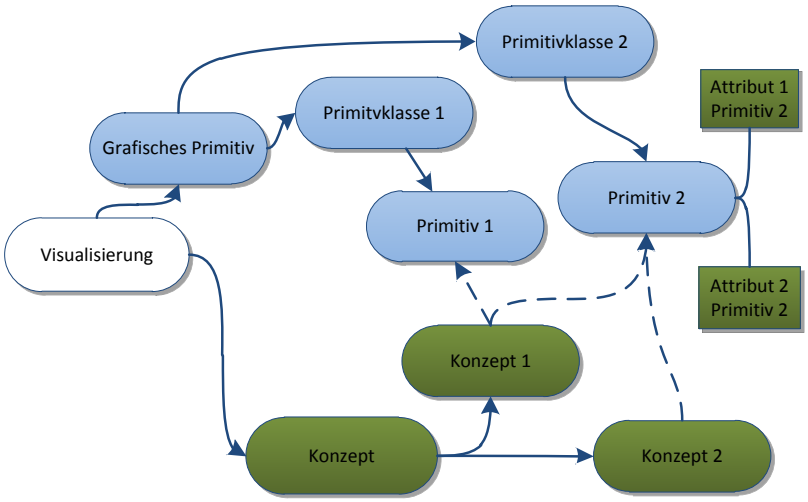


Abbildung 15: Konzeptueller Entwurf einer Grafik-Ontologie. Es werden verschiedene Konzepte definiert und verschiedene grafische Primitive in Klassen unterteilt. Die gestrichelten Pfeile veranschaulichen, aus welchen Primitiven die Konzepte bestehen, die durchgezogenen Pfeile stehen für 'Ist-Unterklasse-Von'. Die rechteckigen, optionalen Attribute können Primitive genauer spezifizieren.

Abbildung 15 zeigt, wie eine Grafik-Ontologie im Grundsatz aufgebaut sein soll. Es werden darin verschiedene Visualisierungskonzepte definiert. Die verschiedenen grafischen Primitive werden in verschiedene Klassen unterteilt. Jedem Konzept werden dann die grafischen Primitive zugewiesen, aus denen es bestehen kann. In Abbildung 15 wird dies durch die gestrichelten Linien dargestellt. 'Konzept 1' kann also aus Instan-

zen von ’Primitiv 1’ und ’Primitiv 2’ bestehen, ’Konzept 2’ dagegen nur aus Instanzen von ’Primitiv 2’. Die durchgezogenen Pfeile stehen für ’Ist-Unterklasse-Von’. ’Konzept 1’ und ’Konzept 2’ sind also Unterklassen von ’Konzept’. Zusätzlich kann die Grafik-Ontologie für jedes Primitiv Attribute enthalten, die zur Darstellung des Primitivs definiert werden müssen. In Abbildung 15 benötigt nur ’Primitiv 2’ Attribute. Wie das im Detail aussehen kann, ist in Abbildung 16 zu sehen und wird in Abschnitt 4.2.2 beschrieben.

Domänen-Ontologie
Die Domänen-Ontologie enthält domänenspezifische Informationen zur Annotation grafischer Elemente und der dazugehörigen Daten. Ebenso enthält diese Ontologie, falls notwendig, die zu visualisierenden Daten. Für das Einsatzszenario Bevölkerungsentwicklung werden die verwendeten Entitäten sowie die benötigten Bevölkerungszahlen definiert. Eine Entität kann in diesem Fall beispielsweise ein Stadtteil, eine Stadt, ein Landkreis oder ein Bundesland sein, je nach Granularität, die visualisiert werden soll. Zusätzlich werden verschiedene Klassen definiert, mit denen grafische Elemente annotiert werden können. Beim Einsatzszenario Bevölkerungsentwicklung sind das verschiedene Eigenschaften, die die Bevölkerungsentwicklung charakterisieren, zum Beispiel ’Ansteigend’, ’Gleichbleibend’ oder ’Sinkend’.
Für das Einsatzszenario Automobilvisualisierung werden hier die verschiedenen Bauteile, aus den das Auto zusammengesetzt ist, beispielsweise ’Rad’, ’Spiegel’ oder ’Auspuff’ definiert. Damit können dann verschiedene grafische Elemente, die ein Rad repräsentieren, annotiert werden. Wenn der Ersteller der Visualisierung einen Platzhalter für Rad-Elemente hinzufügt, kann der Benutzer diesen Platzhalter durch alle grafischen Elemente ersetzen, die mit ’Rad’ annotiert sind.
Die Art der Visualisierungen in den beiden Einsatzszenarien unterscheidet sich. Während bei der Bevölkerungsentwicklung verschiedene Eigenschaften beschrieben sind, sind es bei der Automobilvisualisierung konkrete Teile des Autos.
Darüber hinaus werden in der Domänen-Ontologie verschiedene Detailstufen festgelegt. Mit diesen kann jedes Element einer Visualisierung annotiert werden. In Abschnitt 4.2.4 wird der Nutzen dieser Detailstufen erläutert.

4.2.2. Annotieren von Visualisierungen

Um Visualisierungen durch den Benutzer anpassbar zu machen, können sie mit Informationen aus den beiden Ontologien annotiert werden. Während die Annotationen aus der Grafik-Ontologie dazu dienen, das Visualisierungskonzept austauschbar zu machen, werden die Annotationen aus der Domänen-Ontologie zum Austauschen einzelner grafischer Elemente benötigt.

Annotationen aus der Grafik-Ontologie
Beim Erstellen einer Visualisierung annotiert der Ersteller die Visualisierung mit Visualisierungskonzepten aus der Grafik-Ontologie. Eine Annotation mit einem Visualisie-

rungskonzept bedeutet, dass diese Visualisierung in dem Visualisierungskonzept dargestellt werden kann. Die Visualisierung eines Autos wird daher mit Visualisierungskonzepten wie 'Auto-Frontansicht' oder 'Auto-Seitenansicht' annotiert, die Visualisierung von Bevölkerungsentwicklungen dagegen mit 'Balkendiagramm', 'Landkarte' oder 'Tree-Map'. Ein Visualisierungsprogramm weiß, wie die Visualisierungen aussehen können, weil in der Ontologie für jedes Visualisierungskonzept gespeichert ist, aus welchen Primitiven es zusammengesetzt ist.

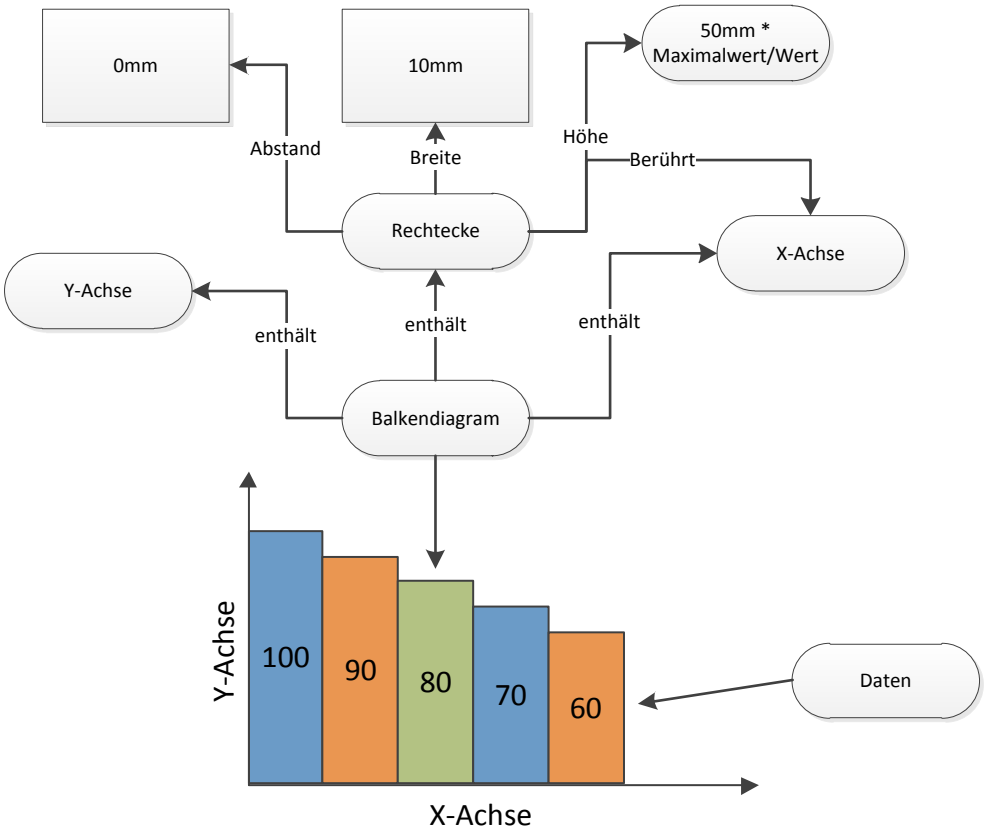


Abbildung 16: Ein Balkendiagramm kann mit Hilfe von Informationen aus der Ontologie und den Daten erzeugt werden. Der Ausschnitt aus der Ontologie (oberer Teil) definiert, dass ein Balkendiagramm aus X-Achse, Y-Achse und Rechtecken besteht. Rechtecke berühren die X-Achse und haben die Parameter Höhe, Breite und Abstand.

Abbildung 16 veranschaulicht am Beispiel eines Balkendiagramms, wie die Informationen aus der Ontologie eine Visualisierung definieren. Der obere Teil der Abbildung ist ein Ausschnitt aus der Grafik-Ontologie. Es ist definiert, dass ein Balkendiagramm aus einer X- und einer Y-Achse, sowie beliebig vielen Rechtecken besteht. Die Rechtecke berühren alle die X-Achse, haben einen Abstand untereinander von 0 mm und

eine Breite von 10 mm. Die Daten definieren die Höhe der Rechtecke. Die Höhe der Rechtecke wird durch die Formel *50mm * M/W* berechnet, wobei *M* der Maximalwert der Daten und *W* der aktuelle Wert ist. Das Visualisierungsp rogramm kann mit den Informationen aus der Ontologie und den Daten das Balkendiagramm visualisieren. Der untere Teil der Grafik stellt das Balkendiagramm dar, das durch die Kombination der Daten und der Informationen aus der Grafik-Ontologie, entstanden ist.

Annotationen aus der Domänen-Ontologie
Einzelne grafische Elemente einer Visualisierung werden mit Klassen aus der Domänen-Ontologie annotiert. So werden zum Beispiel alle Bilder von Rädern mit der Klasse 'Rad' und alle grafischen Elemente, die eine steigende Bevölkerungszahl repräsentieren (zum Beispiel ein Plus oder ein Pfeil nach oben) mit der Klasse 'Steigend' annotiert. Abbildung 17 verdeutlicht die Verwendung der Annotationen aus der Domänen-

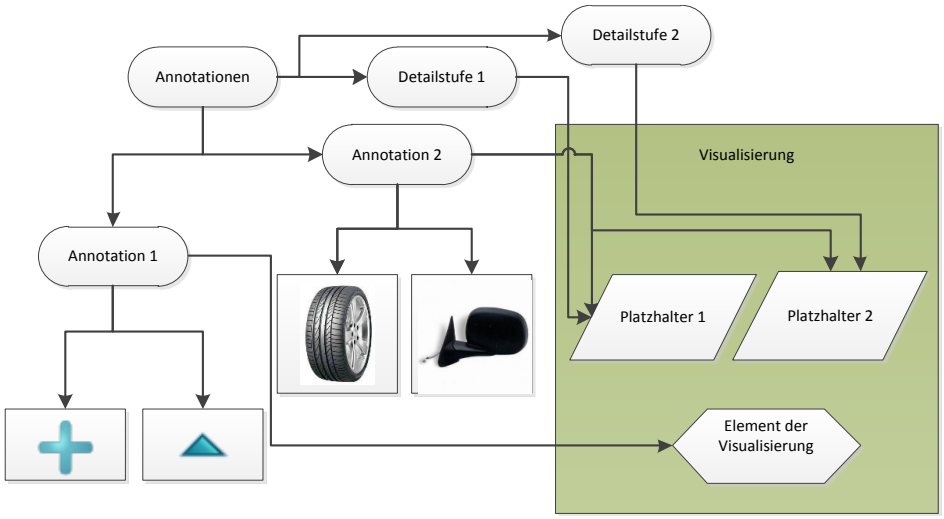


Abbildung 17: Die zur Verfügung stehenden grafischen Elemente werden annotiert (links). Der Visualisierung werden mit Annotation und Detailstufe annotierte Platzhalter hinzugefügt oder vorhandene Elemente werden annotiert (rechts).

Ontologie. Die viereckigen Elemente stehen für grafische Elemente, zum Beispiel Plus, Pfeil, Rad oder Spiegel. Die Elemente werden mit Annotationen aus der Domänen-Ontologie annotiert. Der grüne Kasten stellt eine Visualisierung dar. Die Visualisierung kann mit Platzhaltern für grafische Elemente (in der Abbildung durch Parallelogramme dargestellt) angereichert werden. Diese Platzhalter werden ebenfalls mit Annotationen aus der Domänen-Ontologie annotiert. Anstatt selber Platzhalter einzufügen, können auch vorhandene Elemente der Visualisierung annotiert werden (in der Grafik durch ein Sechseck dargestellt). Die eingefügten Platzhalter und die annotierten Elemente können darüber hinausgehend mit Detailstufen annotiert werden. Der Benutzer kann

dadurch die Visualisierung dahin gehend verändern, dass er nur Elemente mit bestimmten Detailstufen anzeigen lassen möchte.
Wenn Daten visualisiert werden sollen, dann können auch diese Daten annotiert werden. Jedes einzelne Datum eines Datensatzes wird annotiert. Bei dem Einsatzszenario Bevölkerungsentwicklung wird jede Entität annotiert. Je nach dem ob sich die Bevölkerungszahlen positiv, negativ oder neutral entwickelt haben, werden unterschiedliche Annotationen verwendet. Anschließend können der Visualisierung Platzhalter hinzugefügt werden, die mit den Daten verknüpft sind. Die Daten können entweder manuell oder automatisch annotiert werden. Beim Einsatzszenario Bevölkerungsentwicklung könnten beispielsweise Schwellwerte definiert werden, anhand derer automatisch annotiert werden kann. So wird jede Entität, deren Bevölkerungswachstum zwischen -0,5% und +0,5% liegt, mit der Annotation 'Gleichbleibend' annotiert. Bei automatisch generierten Daten kann die Annotation schon beim Erzeugen der Daten vorgenommen werden.

4.2.3. Von der Ontologie zur Visualisierung

Die Informationen aus der Grafik-Ontologie reichen nicht aus, um daraus eine konkrete Visualisierung zu erstellen. Die Information, dass eine Tree-Map oder ein Balkendiagramm aus Rechtecken besteht, sagt nicht zwangsläufig etwas darüber aus, wie diese Rechtecke angeordnet sein sollen. In Kombination mit den Informationen und Daten aus der Domänen-Ontologie ließen sich die entsprechenden Visualisierungen theoretisch erzeugen. Es besteht zwar die Möglichkeit, diese Informationen über zusätzliche Attribute zu liefern, der Aufwand, die Visualisierung zu erzeugen, ist trotzdem beträchtlich. Dies wird nun am Beispiel Tree-Map erläutert:
In Abbildung 18 ist ein Beispiel für eine Tree-Map zu sehen. Diese besteht aus lauter Rechtecken, die in Abhängigkeit der jeweils zugehörigen Daten verschieden groß sind. Höhe und Breite lassen sich also in Abhängigkeit der Daten berechnen. Die Anordnung der Rechtecke, so dass die Tree-Map insgesamt auch die Form eines Rechtecks hat, ist dagegen nicht trivial und müsste algorithmisch gelöst werden. Solche ontologiebasierte Algorithmen müssten für die verschiedenen Visualisierungskonzepte entwickelt und implementiert werden. Der Aufwand, sowohl beim Erstellen der Ontologien, als auch beim Entwickeln der Algorithmen wäre beträchtlich. In dieser Diplomarbeit wurde zur Entwicklung des Prototyps deshalb ein vereinfachtes Konzept verwendet, bei dem die Visualisierungen ein Hintergrundbild verwenden, das mit Platzhaltern angereichert werden kann (siehe Abschnitt 5.1.3).

4.2.4. Interaktionskonzept zur interaktiven Optimierung von Visualisierungen

Die in Abschnitt 4.2.1 vorgestellten Ontologien sowie die in Abschnitt 4.2.2 vorgestellten Konzepte zur Annotation von Visualisierungen sind die Grundlage des in diesem Abschnitt beschriebenen Interaktionskonzepts zur interaktiven Optimierung von Visualisierungen. Das Interaktionskonzept ist in zwei Bereiche unterteilt, auf der einen Seite

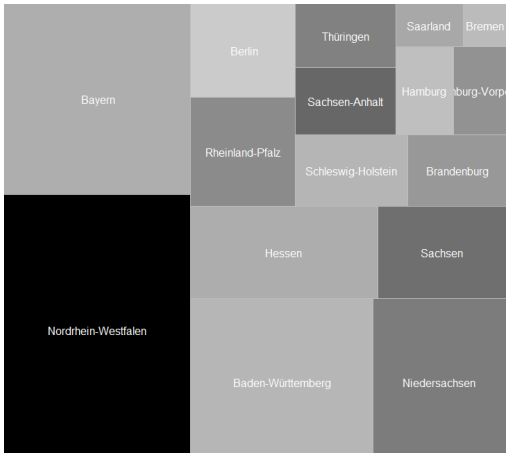


Abbildung 18: Zu sehen ist eine Tree-Map, die die Bevölkerung der deutschen Bundesländer darstellt. Je mehr Einwohner ein Bundesland hat, desto größer das jeweilige Rechteck.

der Austausch einzelner grafischer Elemente und auf der anderen Seite der Austausch des gesamten Visualisierungskonzepts.

Austausch grafischer Elemente
Wie in Abschnitt 4.2.2 beschrieben, kann eine Visualisierung beim Erstellen mit annotierten Platzhaltern angereichert werden, oder es können Elemente der Visualisierung annotiert werden. Platzhalter und annotierte Elemente sind aus Sicht des Benutzers gleichwertig. Um Verwechslungen zwischen annotierten Elementen der Visualisierung und grafischen Elementen, die diese ersetzen können, zu vermeiden, wird im Folgenden von Platzhaltern die Rede sein. Diese Platzhalter können durch alle grafischen Elemente ersetzt werden, die mit der selben Annotation annotiert sind wie der Platzhalter. Der Ersteller kann festlegen, welche grafischen Elemente er für welche Platzhalter verwenden will. Wenn der Benutzer mit einzelnen grafischen Elementen unzufrieden ist, kann er sie durch alle anderen Elemente, die mit der selben Klasse annotiert sind, austauschen. So könnte er beispielsweise ein Rad mit Stahlfelgen durch eines mit Alufelgen oder ein Plus durch einen Pfeil nach oben ersetzen.
Der Austausch kann entweder pro Platzhalter erfolgen oder pro Annotation. Falls er pro Annotation erfolgt, wird das gewählte Element für alle Platzhalter mit dieser Annotation verwendet. Auf diese Weise kann der Benutzer die Visualisierung an seinen persönlichen Bedarf anpassen, ohne dabei das Visualisierungskonzept zu wechseln. Auch anfänglich unverständliche Darstellungen können damit zu einer verständlicheren Darstellung umgebaut werden.

Austausch des Visualisierungskonzepts
Wie in Abschnitt 4.2.2 beschrieben, kann die Visualisierung mit verschiedenen Visuali-

sierungskonzepten aus der Grafik-Ontologie annotiert werden. Der Benutzer kann zwischen diesen Visualisierungskonzepten, zum Beispiel zwischen Balkendiagramm und Tree-Map, wechseln. Aufgrund der Informationen über das Visualisierungskonzept aus der Grafik-Ontologie und den zu visualisierenden Daten können die verschiedenen Visualisierungskonzepte automatisch erstellt werden.

Anpassung der Detailstufe

Der Benutzer hat die Möglichkeit, sich die Visualisierung in verschiedenen Detailstufen anzuschauen. So kann er sich beispielsweise zuerst einen Überblick über die gesamte Visualisierung verschaffen und sich anschließend auf einen bestimmten Bereich konzentrieren und diesen vergrößern. Beim Verschaffen des Überblicks kann es sinnvoll sein, dass nicht alle Details angezeigt werden, da die Visualisierung sonst leicht unübersichtlich wird. Bei der Vergrößerung eines Bereichs sollten aber alle Details angezeigt werden (siehe [Shn96]). Um dies zu ermöglichen, muss der Ersteller die Platzhalter mit Detailstufen aus der Domänen-Ontologie annotieren.
Der Benutzer kann eine Detailstufe, die er gerne hätte, auswählen. Es werden dann nur diese Elemente angezeigt, die mit einer Detailstufe annotiert sind, die geringer ist, als die ausgewählte. Abbildung 19 veranschaulicht das Konzept anhand von zwei Land-

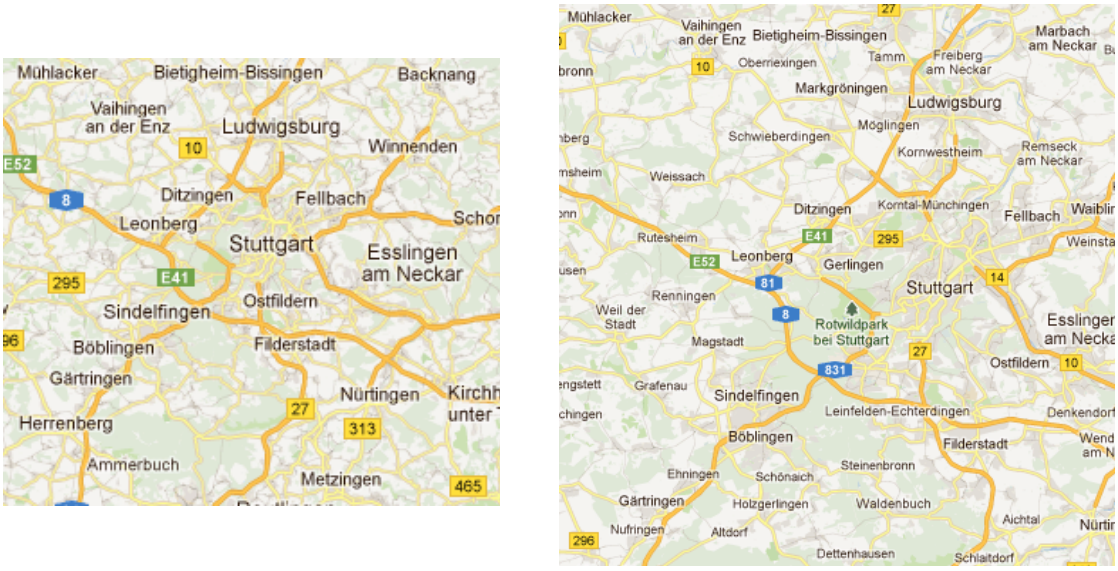


Abbildung 19: Zwei Kartenausschnitte der Region Stuttgart mit unterschiedlichem Maßstab. Auf dem linken ist der Detailgrad deutlich niedriger, daher sind viele Städte nur rechts zu sehen [map].

kartenausschnitten. Der linke Ausschnitt hat einen deutlich kleineren Maßstab. Damit die Karte trotzdem übersichtlich bleibt, wurden einige Informationen weggelassen. So sind einige kleinere Städte, die im rechten Ausschnitt zu sehen sind, im linken nicht vorhanden.

5. Umsetzung des Konzepts

Dieses Kapitel beschreibt die Bereiche des in Kapitel 4 vorgestellten Konzepts, die in einem Prototypen umgesetzt werden sollen. Abschnitt 5.1 beschreibt, welche Ontologien entworfen werden, während Abschnitt 5.2 das im Prototyp implementierte Interaktionskonzept zur interaktiven, verständnisorientierten Optimierung von semantisch annotierten Visualisierungen erläutert. Der Prototyp kann für das Einsatzszenario Bevölkerungsentwicklung (siehe 4.1.1) verwendet werden. Die Granularität wurde so festgelegt, dass die Daten für Bundesländer vorliegen. Auf eine feinere Granularität (zum Beispiel Landkreise oder Städte) wurde verzichtet, da eine feinere Granularität nicht nötig ist, um zu zeigen, dass das Konzept grundsätzlich funktionieren kann.

5.1. Erstellung der Ontologien

Im Rahmen dieser Diplomarbeit sollen zwei Ontologien entworfen werden. Einerseits eine sogenannte Domänen-Ontologie und andererseits eine Grafik-Ontologie. Durch Annotation mit Ressourcen aus der Domänen-Ontologie soll die Bedeutung, die durch die Visualisierung vermittelt werden soll, beschrieben werden. Die Bedeutung kann nur der Ersteller der Visualisierung kennen. Deshalb muss er auch im Zuge der Erstellung der Visualisierung die Annotationen vornehmen. Wie das Erstellen einer Ontologie in der Literatur beschrieben wird, kann in Kapitel 2.4 nachgelesen werden. Obwohl das Erstellen einer Grafik-Ontologie nicht besonders viel mit der Erstellung einer Wein-Ontologie zu tun hat, soll hier trotzdem versucht werden, dem vorgestellten Schema zu folgen.
In den folgenden beiden Abschnitten werden die beiden Ontologien und ihre Erstellung beschrieben.

5.1.1. Domänen-Ontologie

In der folgenden Aufzählung wird das Schema zur Erstellung von Ontologien für die Domänen-Ontologie Schritt für Schritt umgesetzt.

Anwendungsgebiet fokussieren
Diese Ontologie soll verwendet werden, um zu visualisierende Daten und grafische Elemente, die diese Daten repräsentieren, zu annotieren. Als Domäne für die Umsetzung wurde die Bevölkerungsentwicklung zwischen 2007 und 2009 in den deutschen Bundesländern gewählt.

Bestehende Ontologien wiederverwenden
Im Semantic Web konnte für dieses Einsatzgebiet keine bestehende Ontologie gefunden werden.

Relevante Begriffe identifizieren
Als relevante Begriffe wurden die 16 deutschen Bundesländer und ihre Bevölkerungszahlen aus den Jahren 2007 und 2009 identifiziert. Darüber hinaus werden Annotationen in der Ontologie definiert, die die Bevölkerungsentwicklung beschreiben, zum Beispiel 'Steigend', 'Gleichbleibend' und 'Sinkend'.

Klassenhierarchie festlegen
Als Wurzelklasse wurde die Klasse 'Bevölkerungsdomäne' definiert. Sie hat zwei Unterklassen. Die eine Unterklasse ist 'Bundesland', die für jedes der 16 deutschen Bundesländer eine weitere Unterklasse hat. Die andere Unterklasse ist 'Bevölkerungsentwicklung', die in fünf Unterklassen unterteilt ist, die die Entwicklung der Bevölkerungszahlen beschreiben.

Relationen definieren
Jede Bundesland-Klasse hat Relationen zu zwei Literalen. Dies sind jeweils die Bevölkerungszahlen aus den Jahren 2007 und 2009.

Abbildung 20 zeigt einen Ausschnitt der Ontologie. Der hellere, blaue Bereich der Ontologie repräsentiert die Daten zur Bevölkerungsentwicklung in den deutschen Bundesländern. Zu jedem Bundesland gehören zwei Integer-Literale, die jeweils die Anzahl der Bewohner in diesem Bundesland in den Jahren 2007 und 2009 angeben[8]. In der Abbildung ist dies beispielhaft für die beiden Bundesländer Baden-Württemberg und Bayern dargestellt. In der vollständigen Ontologie, die durch den Prototypen verwendet wird, sind diese Informationen für alle 16 deutschen Bundesländer enthalten (siehe Abbildung 45).
Der dunklere, grüne Bereich stellt fünf verschiedene Tendenzen dar, die zeigen, wie sich die Bevölkerung entwickelt haben kann. Mit diesen Tendenzen kann der Ersteller einer Visualisierung jedes Bundesland annotieren. In Abhängigkeit von der Annotation stehen dem Benutzer der Visualisierung dann verschiedene grafische Elemente zur Verfügung, mit denen die Bevölkerungsentwicklung des Bundeslandes in der Visualisierung dargestellt werden kann.

5.1.2. Grafik-Ontologie

In der folgenden Aufzählung wird das Schema zur Erstellung von Ontologien für die Grafik-Ontologie Schritt für Schritt abgearbeitet.

Anwendungsgebiet fokussieren
Die Grafik-Ontologie soll verwendet werden, um Visualisierungskonzepte zu beschreiben. Daher wurden im ersten Schritt verschiedene Visualisierungstechniken[9] analy-

[8] https://de.wikipedia.org/wiki/Demografie_Deutschlands#Aktuelle_Entwicklung
[9] http://www.visualcomplexity.com

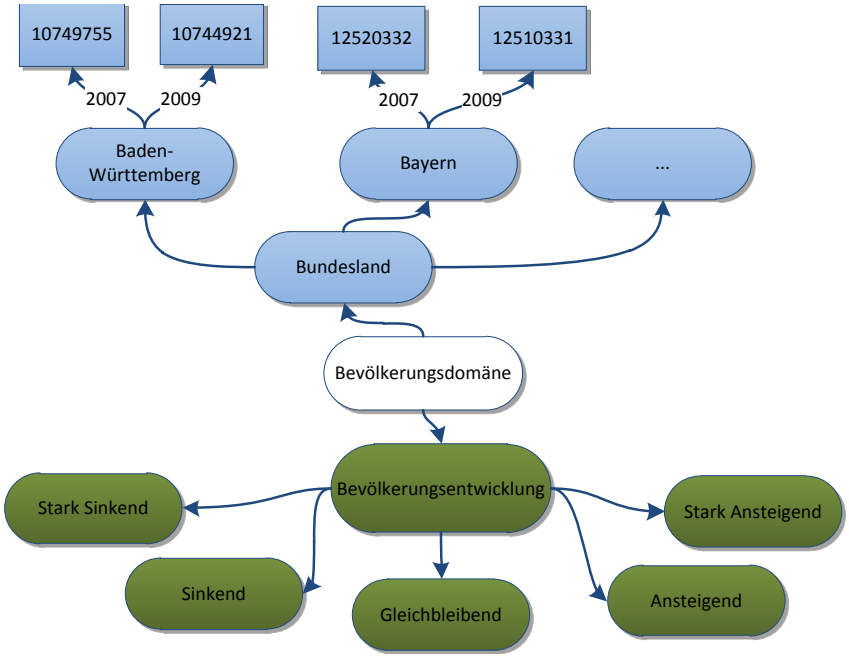


Abbildung 20: Ausschnitt aus der Domänen-Ontologie über die Bevölkerungsentwicklung deutscher Bundesländer zwischen 2007 und 2009. Zu sehen sind: fünf Tendenzen, die die Bevölkerungsentwicklung charakterisieren, zwei Bundesländer und ihre Bevölkerungszahlen.

siert. Dabei wurden einerseits die verwendeten grafischen Primitive, aus denen Visualisierungen bestehen, als auch die verwendeten Visualisierungskonzepte analysiert. Abbildung 21 zeigt beispielhaft die Zerlegung eines Scatter-Plots in einzelne grafische Primitive.

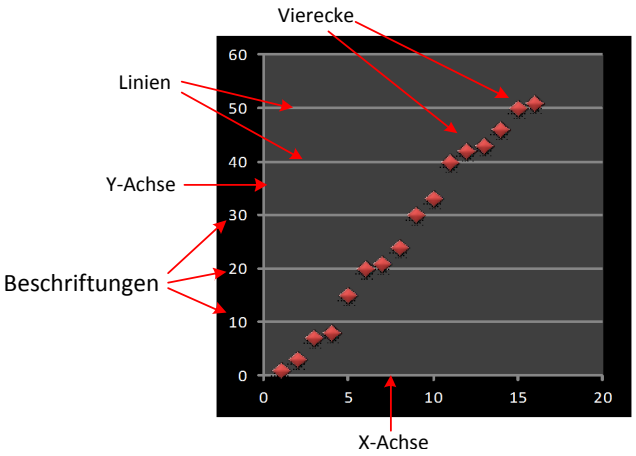


Abbildung 21: Zerlegung eines Scatter-Plots in grafische Primitive

Bestehende Ontologien wiederverwenden
Auch für die Grafik-Ontologien wurden keine wiederverwendbaren Ontologien gefunden.

Relevante Begriffe identifizieren
Die grafischen Primitive wurden in vier verschiedene Klassen aufgeteilt. Die vier Klassen sind im Folgenden mit Beispielen aufgelistet:

Koordinatensystem-Elemente: Achse

Text-Elemente: Beschriftung

Graphen-Elemente: Knoten, Kante

Geometrische Objekte: Linie, Dreieck, Viereck, Kreis, Punkt, Ellipse, Polygon

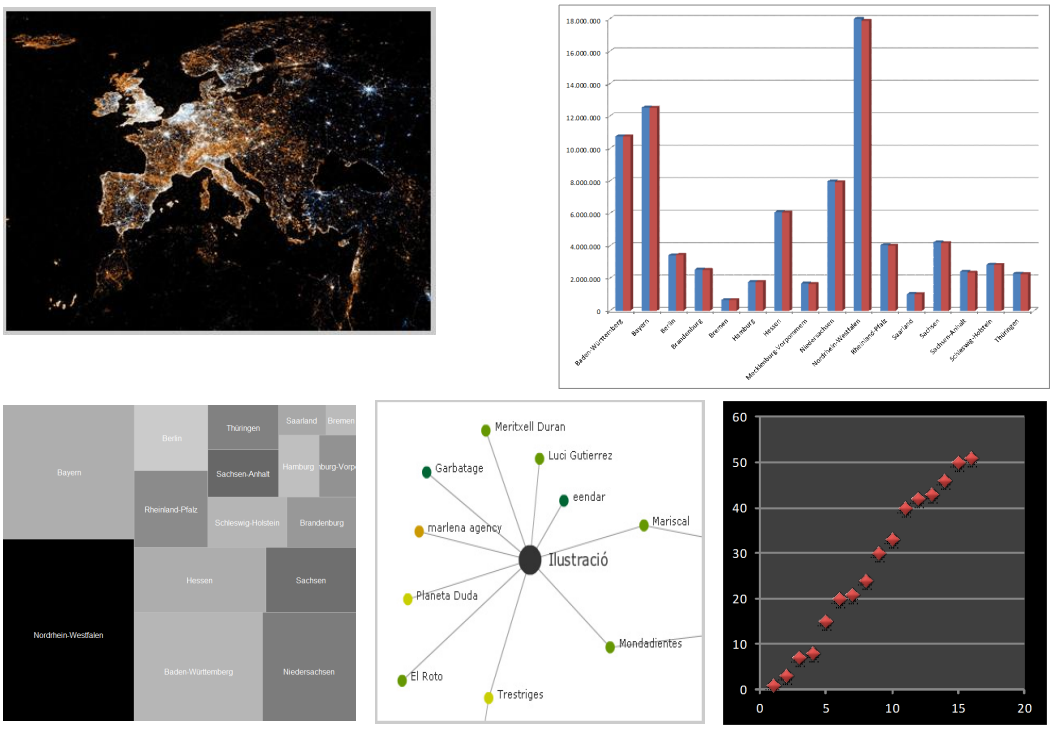


Abbildung 22: Beispiele verschiedener Visualisierungskonzepte: Landkarte (links oben), Balkendiagramm (rechts oben), Tree-Map (links unten), Graph (unten Mitte), Scatter-Plot (rechts unten)

Zusätzliche wurden fünf häufig verwendete Visualisierungskonzepte in die Ontologie aufgenommen, für die in Abbildung 22 jeweils ein Beispiel zu sehen ist:

- Balkendiagramm

- Landkarte
- Tree-Map
- Graph
- Scatter-Plot

Klassenhierarchie festlegen
Als Wurzelklasse wurde die Klasse 'Visualisierung' mit zwei Unterklassen definiert. Eine davon ist 'Konzept', deren Unterklassen die verschiedenen Visualisierungskonzepte sind. Die andere ist 'Grafisches Primitiv', deren Unterklassen Klassen von grafischen Primitiven sind, deren Unterklassen wiederum die verschiedenen grafischen Primitive selbst sind.

Relationen definieren
Den Visualisierungskonzepten wurden die grafischen Primitive zugewiesen, aus denen sie bestehen, beziehungsweise bestehen können. Tabelle 1 veranschaulicht diese Zuordnung.

Visualisierungskonzept	Grafische Primtive
Balkendiagramm	Beschriftungen, Vierecke, X-Achse, Y-Achse
Landkarte	Beschriftungen, Polygone
Tree-Map	Beschriftungen, Vierecke
Graph	Beschriftungen, Kanten, Knoten
Scatter-Plot	Beschriftungen, Linien, Vierecke, X-Achse, Y-Achse

Tabelle 1: Zuordnung der Visualisierungskonzepte zu den enthaltenen grafischen Primitiven.

Primitive können durch zusätzliche Attribute genauer spezifiziert werden. In Abbildung 23 ist dies am Beispiel des Primitivs „Rechteck“ dargestellt. Es verfügt über die Attribute „Höhe“, „Breite“ und „Position“. Diese Attribute können absolut oder relativ zu bestimmten Datenwerten angegeben werden.

Bei der Implementierung des Prototyps wurde die verwendete Ontologie auf die in Abbildung 23 gezeigten Visualisierungskonzepte und die dafür benötigten grafischen Primitive eingeschränkt. Der helle, blaue Bereich der Abbildung stellt die Primitive und ihre Einteilung in verschiedene Klassen dar, der dunkle, grüne Bereich die verschiedenen Konzepte. Ein gestrichelter Pfeil zwischen einem Konzept und einem Primitiv bedeutet, dass dieses Konzept Instanzen dieses Primitivs enthalten kann. Die durchgezogenen Pfeile stehen für die Relation 'Ist-Unterklasse-Von'. Die Klasse „Balkendiagramm“ ist also eine Unterklasse von Konzept. Die gesamte Grafik-Ontologie ist im Anhang zu finden (Abbildung 44).

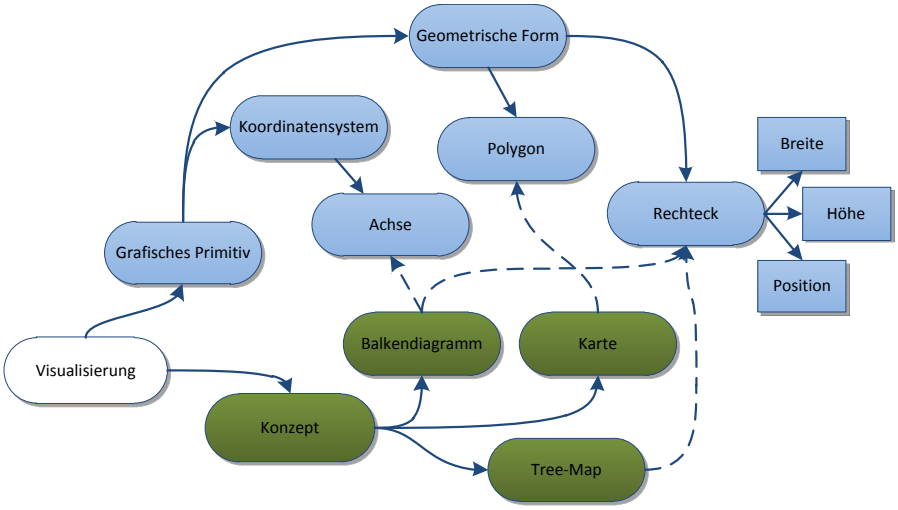


Abbildung 23: Ausschnitt der verwendeten Grafik-Ontologie. Es werden drei Konzepte definiert (unterer, grüner Bereich) und verschiedene grafische Primitive in zwei Klassen unterteilt. Die gestrichelten Pfeile veranschaulichen, aus welchen Primitiven die Konzepte bestehen, die durchgezogenen Pfeile stehen für 'Ist-Unterklasse-Von'. Das Primitiv „Rechteck" wird durch zusätzliche Attribute beschrieben (ganz rechts).

5.1.3. Lücke zwischen Ontologie und geometrischer Anordnung

Wie in Abschnitt 4.2.3 beschrieben, wäre der Aufwand, um Visualisierungen nur auf Basis von Ontologien zu erzeugen, beträchtlich. Da dieser Aufwand aber den Rahmen einer Diplomarbeit sprengen würde und außerdem nicht Aufgabe dieser Diplomarbeit ist, müssen diese Visualisierungen mit anderen Tools erstellt werden. Balkendiagramme können zum Beispiel mit Excel erzeugt werden, Tree-Maps mit der Programmiersprache *R*.[10] Die erstellten Visualisierungen werden mit Informationen aus der Grafik-Ontologie semantisch annotiert und vom Prototypen als Hintergrundbilder verwendet. Diese Hintergrundbilder können dann mit Platzhaltern für verschiedene grafische Elemente angereichert werden.

5.2. Interaktionskonzept

In den nächsten beiden Abschnitten wird das Interaktionskonzept erläutert, das dem in Kapitel 6 beschriebenen Prototypen zugrunde liegt. Für die Verwendung des Prototyps gibt es zwei Rollen, den Ersteller und den Benutzer. Aus diesem Grund besteht der Prototyp aus einem Hauptfenster mit zwei Tabs. Der eine Tab dient dem Ersteller von Visualisierungen, der andere dem Benutzer. Damit ist eine klare Trennung zwischen Erstellen und Anpassen der Visualisierungen möglich. Abschnitt 5.2.1 zeigt, welche

[10] `http://flowingdata.com/2010/02/11/an-easy-way-to-make-a-treemap/`

Möglichkeiten beim Erstellen einer Visualisierung bestehen, Abschnitt 5.2.2 erläutert, wie eine erstellte Visualisierung an die individuellen Vorlieben und Bedürfnisse eines Benutzers angepasst werden kann. Zusätzlich zeigt in beiden Abschnitten jeweils ein Mockup die Benutzeroberfläche des Prototyps zur Ausführung der Interaktionen.

5.2.1. Interaktionskonzept zum Erstellen einer Visualisierung

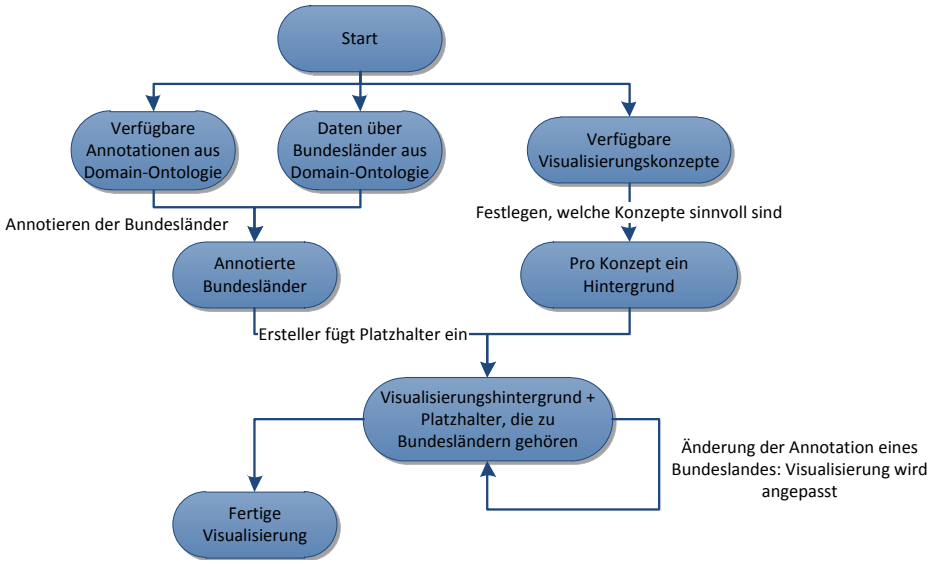


Abbildung 24: Workflow-Diagramm zur Erstellung einer Visualisierung. Der Ersteller muss die Bundesländer annotieren, sinnvolle Visualisierungskonzepte auswählen und diese anschließend mit Platzhaltern anreichern.

Abbildung 24 zeigt die Erstellung einer Visualisierung in einem Workflow-Diagramm. Der Workflow beginnt mit dem Knoten 'Start'. Die Bedeutung der drei nächsten Knoten wird im Folgenden erläutert:

Verfügbare Annotationen Die verfügbaren Annotationen stammen aus der Domänen-Ontologie. Da beispielhaft die in Abschnitt 5.1.1 vorgestellte Domänen-Ontologie verwendet wird, sind dies die Tendenzen der Bevölkerungsentwicklung 'Stark Sinkend', 'Sinkend', 'Gleichbleibend', 'Ansteigend' und 'Stark Ansteigend'.

Daten über Bundesländer Diese Daten sind ebenfalls in der Domänen-Ontologie gespeichert. Konkret sind dies die Bevölkerungszahlen für jedes Bundesland aus den Jahren 2007 und 2009.

Verfügbare Konzepte Die verfügbaren Visualisierungskonzepte sind in der Grafik-Ontologie definiert. Für jedes Konzept muss ein Hintergrundbild vorliegen. Im Prototyp gibt es die Konzepte Landkarte, Tree-Map und Balkendiagramm. Für das Visualisierungskonzept Landkarte ist das Hintergrundbild eine Landkarte von

Deutschland, für Tree-Map ist es eine Tree-Map, die in Abhängigkeit der Bevölkerungsgröße der einzelnen Bundesländer erstellt wurde. Das Balkendiagramm stellt für jedes Bundesland die Bevölkerungsgröße 2007 und 2009 dar. Diese Hintergrundbilder sind mit den zugehörigen Informationen aus der Grafik-Ontologie annotiert.

Um die Bundesländer mit den verfügbaren Annotationen annotieren zu können, soll es im Prototyp eine Tabelle geben, in der der Name und die beiden Bevölkerungskennzahlen für jedes Bundesland stehen (siehe Abbildung 26, Bereich 2). In einer zusätzlichen Spalte soll der Ersteller für jedes Land die seiner Meinung nach passende Annotation auswählen können. Zur Verfügung stehen sollen die Unterklassen der Klasse 'Bevölkerungsentwicklung' aus der Grafik-Ontologie (siehe Abbildung 20). Als Darstellungsform wurde eine Tabelle ausgewählt, da Daten in Datenbanken in aller Regel auch in Tabellenform gespeichert sind. Außerdem ermöglicht eine Tabelle eine einfache visuelle Zuordnung der Annotationen zu den Daten.
Aus den verfügbaren Visualisierungskonzepten soll der Ersteller auswählen können, welche er für seinen aktuellen Anwendungsfall sinnvoll findet. Wenn er zum Beispiel der Meinung ist, dass sich die Bevölkerungsentwicklung nicht mittels eines Balkendiagramms darstellen lässt, dann wählt er dieses Visualisierungskonzept nicht aus.
Nachdem die Bundesländer annotiert und die zu verwendenden Hintergrundbilder ausgewählt sind, kann eine Visualisierung erstellt werden, die Instanzen der verschiedenen Visualisierungskonzepte enthält.

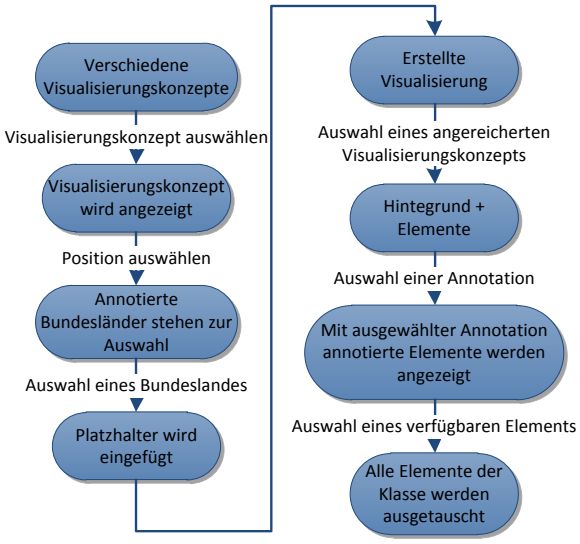


Abbildung 25: Workflow-Diagramm zur Erstellung einer Visualisierung. Der Benutzer wählt ein Visualisierungskonzept aus und kann in diesem grafische Elemente austauschen.

Der Ablauf der Erstellung der Visualisierung ist im linken Teil des Workflow-Diagramms in Abbildung 25 dargestellt und wird im Folgenden erklärt:

1. Es wird ein Visualisierungskonzept ausgewählt.
2. Der zu diesem Konzept gehörende Hintergrund wird angezeigt.
3. Durch einen Klick an eine Position auf dem angezeigten Hintergrund kann dort ein Platzhalter für grafische Elemente hinzugefügt werden.
4. Es wird ein Bundesland ausgewählt, für das der Platzhalter an der angeklickten Position hinzugefügt werden soll.
5. Der Platzhalter wird in der Visualisierung durch ein grafisches Element repräsentiert. Es wird ein beliebiges Element verwendet, das mit derselben Annotation annotiert ist, wie der Platzhalter.

Durch beliebiges Wiederholen der Punkte 3. bis 5. kann der Visualisierungshintergrund mit mehreren Platzhaltern angereichert werden. Durch Wiederholen des gesamten Ablaufs (1. - 5.) können die Instanzen der verschiedenen Konzepte erstellt werden.
Falls im Nachhinein die Annotation eines Bundeslandes verändert wird, dann werden alle grafischen Elemente, die Platzhalter repräsentieren, die zu diesem Bundesland gehören, in der erstellten Visualisierung ausgetauscht.
Abbildung 26 zeigt einen Mockup des Ersteller-Tabs. In Bereich 1 wird die Visualisierung angezeigt. Wenn der Ersteller einen Platzhalter zur Visualisierung hinzufügen möchte, muss er an die Stelle in der Visualisierung klicken, an der er den Platzhalter einfügen will. Anschließend wählt er im sich öffnenden Dialog das Bundesland aus, für das der Platzhalter eingefügt werden soll. In Bereich 2 werden die Bundesländer aus der Domänen-Ontologie in Tabellen-Form angezeigt. Um ein Bundesland zu annotieren, wählt der Ersteller in der DropDown-Box in der Spalte 'Annotation' die gewünschte Annotation aus. In Bereich 3 ist eine Gruppe von CheckBoxes zu sehen. Hier kann der Ersteller auswählen, in welchen Visualisierungskonzepten die Daten visualisiert werden sollen. Intern wird beim Anwählen einer CheckBox der Datensatz mit der Annotation aus der Grafik-Ontologie annotiert. In Bereich 4 kann das Visualisierungskonzept ausgewählt werden, das bearbeitet werden soll.

5.2.2. Interaktionskonzept zur interaktiven Optimierung einer Visualisierung

Der rechte Teil der Abbildung 25 stellt die interaktive Optimierung einer Visualisierung in einem Workflow-Diagramm dar. Der Benutzer verwendet eine vom Ersteller bereitgestellte Visualisierung und passt diese an seine persönlichen Bedürfnisse an. Dazu wählt er ein zur Verfügung stehendes Visualisierungskonzept aus, das ihm daraufhin angezeigt wird. Das Konzept besteht aus dem entsprechenden Hintergrundbild und den die eingefügten Platzhalter repräsentierenden grafischen Elementen. Nun können Elemente ausgetauscht werden. Dazu wird eine Annotation ausgewählt. Durch die Verwendung der Domänen-Ontologie des Einsatzszenarios Bevölkerungsentwicklung stehen die fünf Tendenzen der Bevölkerungsentwicklung als Annotationen zur Verfügung. Nach dieser Auswahl werden alle Elemente angezeigt, die mit dieser Annotation annotiert wurden (siehe Abbildung 27, Bereich 2). Wenn der Benutzer ein anderes Element

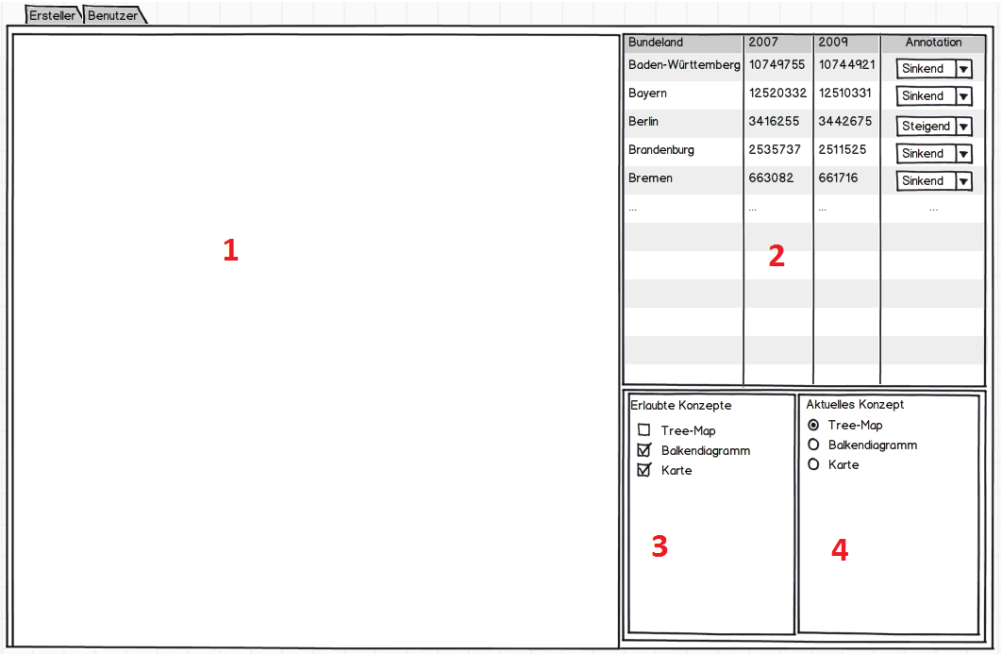


Abbildung 26: Mockup der Ersteller-Tabs. In Bereich 1 wird die Visualisierung angezeigt, in der Tabelle im Bereich 2 können die Bundesländer annotiert werden, in Bereich 3 werden die sinnvollen Visualisierungskonzepte ausgewählt und in Bereich 4 wird das zu bearbeitende Visualisierungskonzept ausgewählt.

auswählt, dann werden alle zu dieser Annotation gehörenden Elemente in der Visualisierung ausgetauscht.
Abbildung 27 zeigt einen Mockup des Benutzer-Tabs. In Bereich 1 wird die durch den Ersteller zur Verfügung gestellte Visualisierung angezeigt. Falls der Benutzer die Visualisierung anpasst, sei es durch den Austausch von grafischen Elementen oder durch den Wechsel des Visualisierungskonzepts, wird die Visualisierung automatisch aktualisiert. In Bereich 2 können grafische Elemente ausgetauscht werden. Dazu wird als Erstes in der DropDown-Box eine Annotation und dann das gewünschte Element in der Gruppe der RadioButtons ausgewählt. In Bereich 3 steht eine DropDown-Box zur Verfügung, mittels derer das Visualisierungskonzept ausgetauscht werden kann.

5.2.3. Allgemeine Funktionen

Zusätzlich zu den bereits beschriebenen Funktionen können sowohl Ersteller als auch Benutzer die aktuelle Visualisierung speichern und wieder laden.

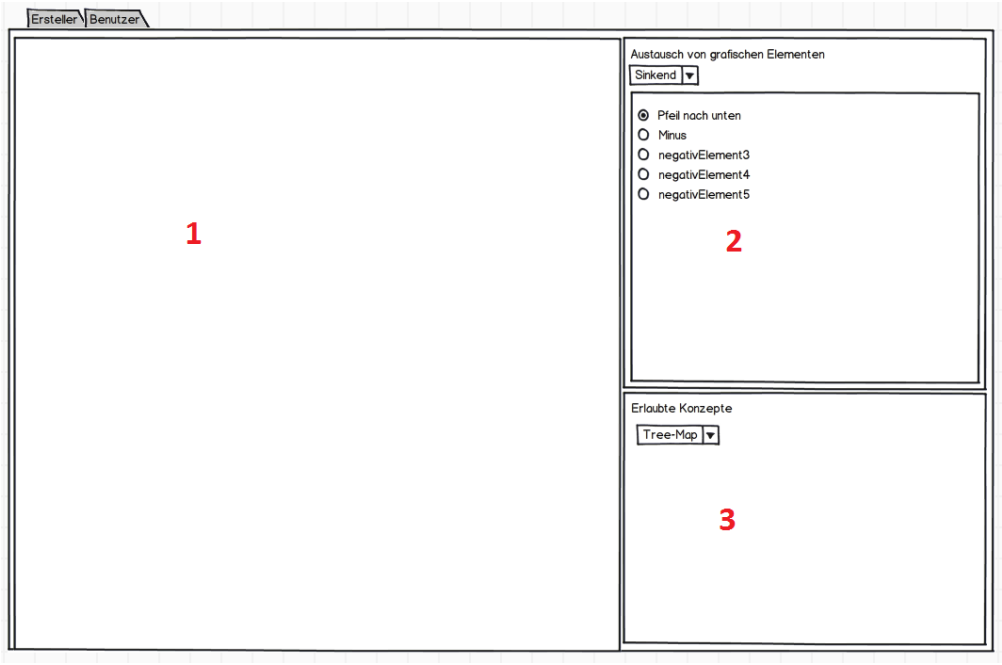


Abbildung 27: Mockup der Benutzer-Tabs. In Bereich 1 wird die Visualisierung angezeigt, in Bereich 2 können grafische Elemente ausgetauscht werden und in Bereich 3 kann das Visualisierungskonzept ausgetauscht werden.

6. Prototyp

Dieses Kapitel beschreibt den im Verlauf dieser Diplomarbeit implementierten Prototyp. Dieser Prototyp soll die grundlegende Umsetzbarkeit des vorgestellten Konzepts zeigen. Abschnitt 6.1 gibt eine Übersicht über die Software-Architektur des Prototyps, während Abschnitt 6.2 die tatsächliche Umsetzung des Konzepts im Prototypen anhand der Benutzeroberfläche beschreibt. Zur Darstellung der Visualisierungen wurde das Grafikformat SVG verwendet, da sich SVG gut eignet, um Grafiken mit semantischen Informationen zu annotieren [YHC03]. Der Prototyp wurde in C# entwickelt. Dies führte bei der Verwendung von SVG zu einigen Schwierigkeiten, da C# keine Unterstützung für SVG mitbringt. Aus diesem Grund musste die Manipulation der SVG-Dateien selbst implementiert werden.
Zum Erstellen der Ontologien wurde RDF verwendet, da dies die einzige Ontologiesprache ist, für deren Verarbeitung eine Bibliothek gefunden werden konnte.

6.1. Architektur

Die für den Prototyp entwickelte Architektur orientiert sich an dem Model-View-Controller Paradigma [KP88] (MVC). Somit besteht der Prototyp aus den drei Komponenten Model, View und Controller. Abbildung 28 illustriert die Aufteilung der verschiedenen Aufgaben, die durch die drei Komponenten implementiert werden.

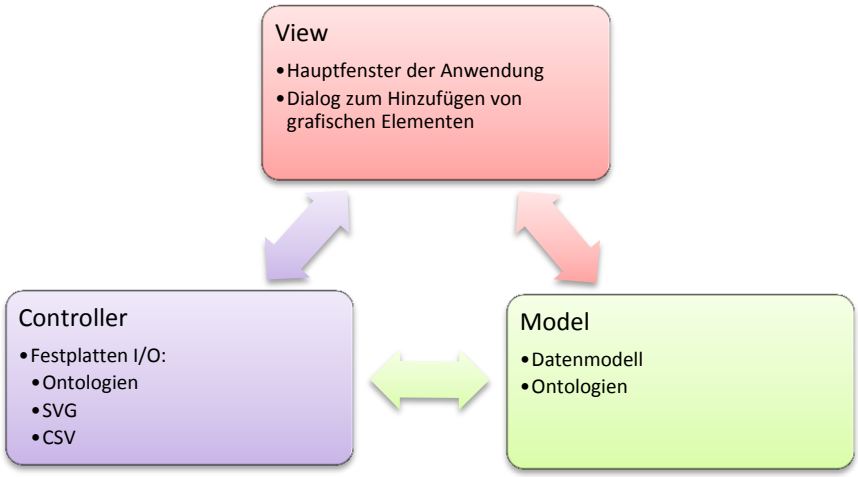


Abbildung 28: Architekturüberblick, angelehnt an MVC Paradigma. Zu sehen sind die drei Komponenten Model, View und Controller und die Aufgaben, die sie erfüllen.

6.1.1. Model

Die Komponente Model hält die von der Anwendung benötigten Daten. Die Hauptklasse des Modells ist die Klasse VisModel. In dieser Klasse sind sämtliche zur Darstellung benötigten Informationen gespeichert. Abbildung 29 gibt einen Überblick über die Klassenstruktur der Komponente. Sämtliche Model-Klassen sind im Namespace InteractiveOptimization.Model zu finden. Die Informationen sind in Objekten der im Folgenden vorgestellten Klassen gespeichert.

IAnnotation
IAnnotation ist ein Interface, das eine Annotation repräsentiert. Es besteht aus den Eigenschaften `Label` und `Uri`, die aus einer der beiden Ontologien stammen. IAnnotation wird von den Klassen VisConcept und SnippetClass implementiert, die in den beiden folgenden Abschnitten beschrieben werden.

SvgSnippet
Beim Programmstart wird für jede SVG-Datei, die im Ordner „SvgSnippets“ liegt und eine gültige Annotation enthält, ein SvgSnippet-Objekt angelegt. Die Annotation wird in *Annotation* gespeichert. Zusätzlich werden der Name und der Pfad der Datei gespeichert. Alle eingelesenen SvgSnippets können vom Benutzer in der Visualisierung verwendet werden, vorausgesetzt die SvgSnippets sind mit einer Annotation aus der Domain-Ontologie annotiert.

SnippetClass
Eine SnippetClass ist eine Abbildung der in der Domänen-Ontologie vorhandenen Annotationen. Bei der verwendeten Domänen-Ontologie aus dem Einsatzszenario Bevölkerungsentwicklung sind das die Unterklassen von 'Bevölkerungsentwicklung', also 'Stark Sinkend', 'Sinkend', 'Gleichbleibend', 'Steigend' und 'Stark Steigend'. Die Annotationen werden beim Programmstart aus der Domänen-Ontologie ausgelesen und jeweils in ein Objekt vom Typ SnippetClass gespeichert. SnippetClass erbt die beiden Eigenschaften *Label* und *Uri* vom Interface IAnnotation. Zusätzlich ist die Methode `ToString()` überschrieben. Sie gibt *Label* zurück.

Dataset
Ein Objekt der Klasse Dataset dient dazu, einen Datensatz mit Informationen aus der Domänen-Ontologie zu speichern. Bei der für den Prototyp gewählten Domänen-Ontologie sind das die Bundesländer und die Informationen über die Bevölkerungsentwicklung. Das Objekt enthält die Eigenschaften *Dates*, eine Liste mit Objekten der Klasse Date und *Annotations*, eine Liste mit Objekten der Klasse VisConcept. Die Liste *Annotations* dient dazu, den Datensatz mit semantischen Informationen anzureichern. Jedes Element dieser Liste steht für ein Visualisierungskonzept, mit dem dieser Datensatz visualisiert werden kann.

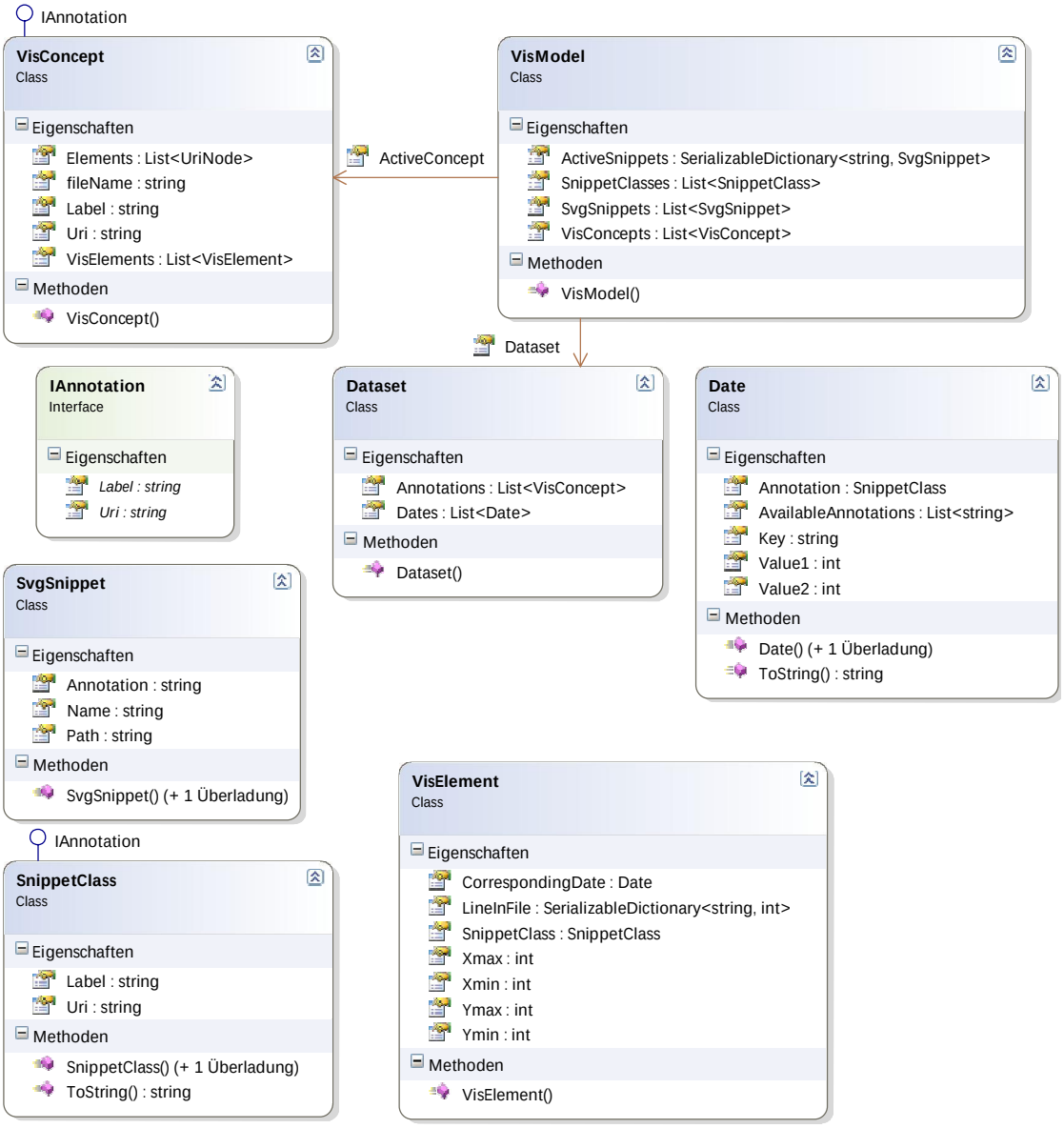


Abbildung 29: Klassendiagramm der Komponente Model

Date

Diese Klasse repräsentiert einzelne Daten eines Datensatzes. In der Domäne 'Bevölkerungsentwicklung' sind das die Bundesländer. Ein Datum besteht aus *Key* vom Typ string und *Value1* und *Value2* vom Typ int. In *Key* wird der Name des Bundeslandes gespeichert, in *Value1* und *Value2* die Bevölkerungszahlen aus den Jahren 2007 und 2009. Darüber hinaus enthält das Datum eine Liste (*AvailableAnnotations*), die angibt, mit welchen Informationen dieses Datum annotiert werden kann. Das Objekt *Annotati-*

on gibt an, mit welcher SnippetClass das Datum annotiert ist. Durch Elemente dieser SnippetClass kann dieses Datum in der Visualisierung repräsentiert werden. In Date wird die Methode `ToString()` überschrieben und gibt den Key des Datums zurück.
Abbildung 30 zeigt beispielhaft, wie ein Date-Objekt mit Informationen aus der Domänen-Ontologie gefüllt wird. Der linke Bereich ist ein Ausschnitt aus der Domänen-Ontologie, der rechte ist das Klassendiagramm der Klasse Date. *Annotation* wird gefüllt, wenn das Date durch den Ersteller annotiert wird. Es wird mit einer Annotation gefüllt, die eine Unterklasse von 'Bevölkerungsentwicklung' ist. Die Liste *AvailableAnnotations* ist mit den Labels aller zur Verfügung stehenden Annotationen gefüllt. *Key* ist das Label (also der Bezeichner) des Bundeslandes, *Value1* und *Value2* sind die beiden Bevölkerungszahlen.

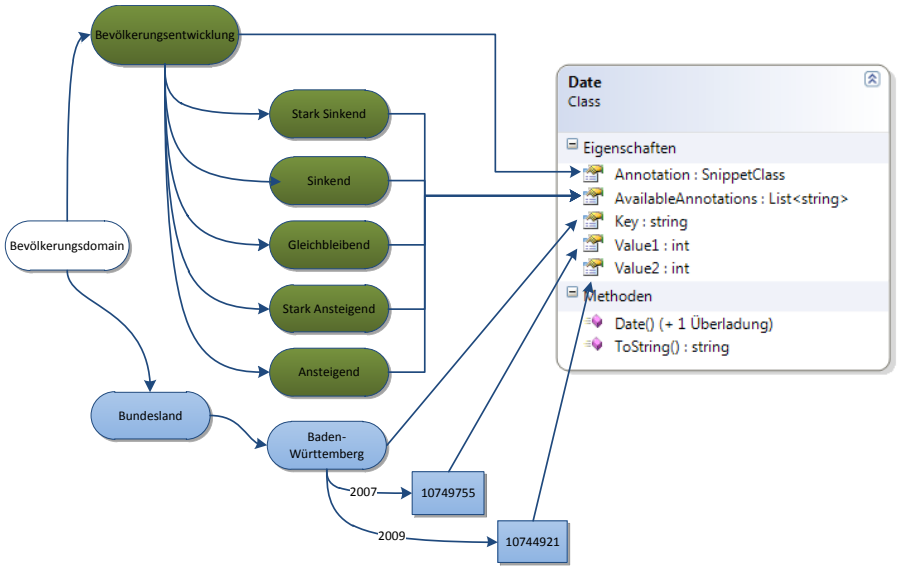


Abbildung 30: Links ist ein Ausschnitt aus der Domänen-Ontologie, rechts die Zuordnung zur Klasse Date zu sehen.

VisConcept
Ein Objekt dieser Klasse steht für ein Visualisierungskonzept, das zur Visualisierung verwendet werden kann. VisConcept erbt vom Interface IAnnotation die Eigenschaften `Label` und `Uri`, da Dataset-Objekte mit VisConcepts annotiert werden können. Um das Visualisierungskonzept anzeigen zu können, wird eine SVG-Datei benötigt. Diese SVG-Datei dient als Hintergrund der Visualisierung. Die durch den Ersteller hinzugefügten Platzhalter werden darin eingefügt. Die zum Konzept gehörende Datei wird durch die Eigenschaft *FileName* angegeben. *VisElements* gibt an, welche VisElements dem Visualisierungskonzept hinzugefügt wurden. *Elements* definiert, welche grafischen Primitive dieses Visualisierungskonzept enthalten kann. Die Informationen, die in *Elements*, *Label* und *Uri* gespeichert sind, stammen aus der Grafik-Ontologie.

Abbildung 31 zeigt beispielhaft das Mapping von Grafik-Ontologie auf die Klasse VisConcept. Die beiden gestrichelten Pfeile bedeuten, dass ein Balkendiagramm aus Achsen und Rechtecken bestehen kann. *Label* und *Uri* stammen aus 'Balkendiagramm', die Liste *Elements* enthält Informationen darüber, aus welchen grafischen Primitiven das VisConcept bestehen kann, beim Beispiel Balkendiagramm also aus Achsen und Rechtecken.

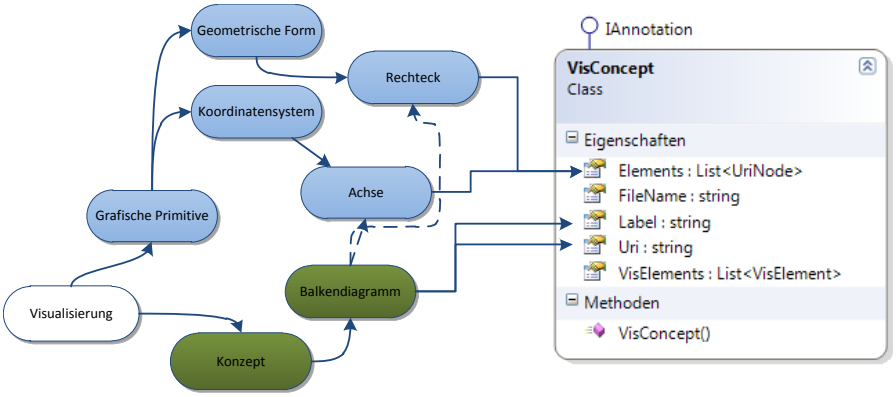


Abbildung 31: Links ist ein Ausschnitt der Grafik-Ontologie, rechts die Zuordnung zur Klasse VisConcept zu sehen.

VisElement

Ein VisElement repräsentiert einen durch den Ersteller zur Visualisierung hinzugefügten Platzhalter. Aufgrund der nicht vorhandenen Unterstützung von SVG in C# ist es notwendig zu wissen, in welcher Zeile der SVG-Datei das VisElement eingebunden ist, damit der Platzhalter später verändert oder gelöscht werden kann. Diese Information ist in *LineInFile* gespeichert. Jedes VisElement gehört zu einem Date-Objekt. Dieses Objekt ist in *CorrespondingDate* gespeichert. *SnippetClass* definiert die SnippetClass, durch deren Elemente der Platzhalter in der Visualisierung ersetzt werden kann. *SnippetClass* ist davon abhängig, mit welcher SnippetClass das zugehörige Date-Objekt annotiert ist. *Xmin*, *Xmax*, *Ymin* und *Ymax* definieren die Fläche, die das VisElement in der Visualisierung einnimmt.

VisModel

VisModel (siehe Abbildung 32) ist, wie bereits erwähnt, die zentrale Klasse des Models. Ein Objekt dieser Klasse wird im Controller gehalten und enthält Objekte der bisher vorgestellten Klassen. Im Folgenden werden die Eigenschaften der Klasse erläutert.
ActiveConcept ist das VisConcept, das gerade aktiv ist. Das *ActiveConcept* wird, falls der Ersteller-Tab ausgewählt ist, angezeigt und kann mit grafischen Elementen angereichert werden. Falls der Benutzer-Tab aktiv ist, wird das *ActiveConcept* ebenfalls angezeigt. Wenn der Benutzer grafische Elemente austauscht, werden diese nur im *ActiveConcept* ausgetauscht.

ActiveSnippets ist ein Dictionary, das jeder SnippetClass ein SvgSnippet zuweist. Dieses SvgSnippet repräsentiert in den Visualisierungen alle VisElements, die mit der SnippetClass verbunden sind.
Dataset ist ein Objekt vom Typ Dataset und speichert die 16 deutschen Bundesländer und ihre Bevölkerungszahlen.
SnippetClasses ist eine Liste mit allen verfügbaren SnippetClasses, *SvgSnippets* ist eine Liste mit allen verfügbaren SvgSnippets und *VisConcept* ist eine Liste mit allen zur Verfügung stehenden Visualisierungskonzepten.

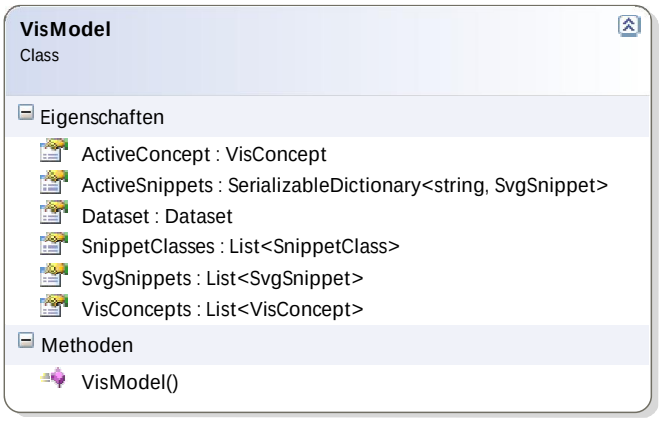


Abbildung 32: Klassendiagramm der Klasse VisModel.

6.1.2. View

Die Komponente View besteht aus den beiden Klassen MainWindow und AddElementForm. Beide Klassen sind im Namespace InteractiveOptimization.UI zu finden.
MainWindow definiert das Hauptfenster der Anwendung. Die Anwendung soll sowohl zum Erstellen von Visualisierungen als auch zum interaktiven Optimieren der erstellten Visualisierungen verwendet werden können. Aus diesem Grund ist das Hauptfenster in zwei Tabs aufgeteilt. Im Ersteller-Tab können Daten importiert und annotiert werden. Diese Daten können mit grafischen Elementen in der Visualisierung dargestellt werden. Im Benutzer-Tab können grafische Elemente und Visualisierungskonzepte ausgetauscht werden. MainWindow stellt neben dem Design der Benutzeroberfläche alle benötigten Event-Handler zur Verfügung.
AddElementForm definiert einen Dialog, mit dem der Ersteller die Visualisierung mit Platzhaltern anreichern kann. Jedes Element gehört zu einem Date-Objekt aus dem Model. Im Dialog wird das Date-Objekt (repräsentiert durch seinen Key) ausgewählt. AddElementForm wird auch dafür verwendet, um hinzugefügte Elemente im Nachhinein zu verändern oder zu löschen und stellt die hierfür benötigten Event-Handler zur Verfügung.

6.1.3. Controller

Die Komponente Controller enthält die Klasse Controller im Namespace InteractiveOptimization.Controller sowie die Klassen CsvReader, RdfReader und SvgReaderWriter im Namespace InteractiveOptimization.Controller.IO (siehe Abbildung 33).

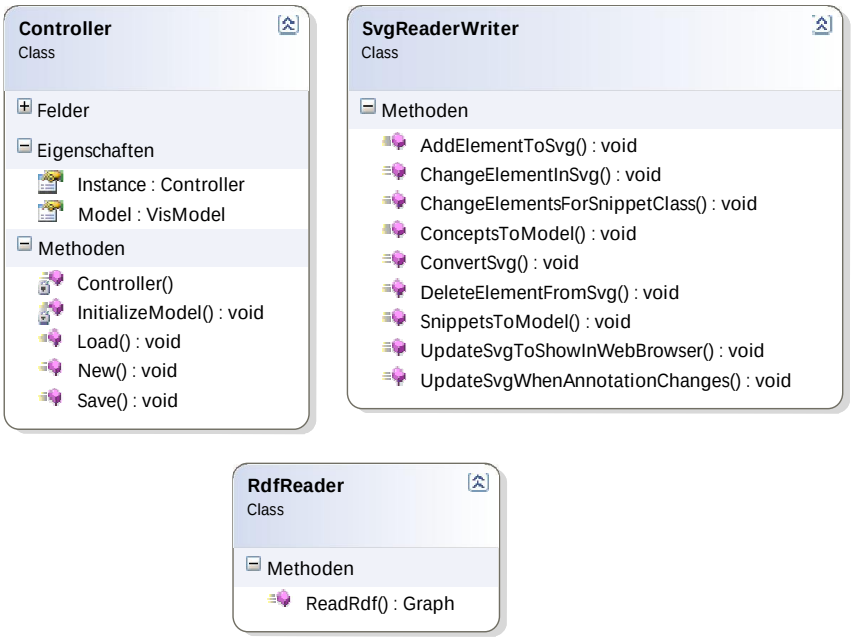


Abbildung 33: Klassendiagramm der Komponente Controller mit den Klassen Controller, SvgReaderWriter und RdfReader. Die Parameter der Methoden wurden in der Abbildung aus Platzgründen weggelassen.

Controller
Die Klasse Controller ist nach dem Entwurfsmuster Singleton [Gam09] implementiert. Das bedeutet, dass es nur eine Instanz der Klasse geben kann. Dadurch ist gewährleistet, dass es nur einen Controller und ein Model gibt und es beim Laden und Speichern keine Konflikte gibt. Controller stellt die Methoden `New()`, `Save()` und `Load()` zur Verfügung. `New()` erzeugt eine neue Version des Models. Um das Model mit Informationen zu füllen, werden die beiden Ontologien eingelesen. Die im Ordner 'SVG' vorhandenen SVG-Dateien werden als Hintergrundbilder für mögliche Visualisierungen in den Ordner 'GraphicSnippets' kopiert, da das Einbinden externer Dateien in SVG-Dateien nur funktioniert, wenn diese im selben Ordner gespeichert sind. `Save()` serialisiert das Model und speichert es in der Datei 'model.xml' im Ordner 'Save' und kopiert die erstellten Visualisierungen in den Ordner 'Save'. `Load()` deserialisiert das Model aus 'model.xml' und kopiert die gespeicherten Visualisierungen in den Ordner 'GraphicSnippets'.

RdfReader
Die Methode `ReadRdf(String path)` wird verwendet, um die beiden Ontologien einzulesen. Da es in C# keine Unterstützung zum Arbeiten mit Ontologien gibt, wird die externe Bibliothek *DotNetRDF* verwendet. DotNetRDF stellt Objekte und Methoden zur Verfügung, mit denen RDF-Ontologien eingelesen und verarbeitet werden können. `ReadRdf()` liest eine angegebene RDF-Datei ein und gibt die Ontologie in einem Graph-Objekt zurück.

DotNetRDF
DotNetRDF[11] ist eine in C# entwickelte Bibliothek zum Verarbeiten und Erstellen von Ontologien in RDF. Sie steht unter der *GNU General Public License*[12] (GPL). DotNetRDF orientiert sich dabei an RDF. Die zentralen Klassen haben selbsterklärende Namen und repräsentieren dabei die Konzepte von RDF wie Graphen (IGraph), Knoten (INode) und Tripel (Triple). Es können die gängigsten RDF-Syntaxen eingelesen und geschrieben werden. Außerdem stehen Methoden zur Verfügung, um Ontologien zu durchsuchen und zu verändern.

SvgReaderWriter
SvgReaderWriter ist die wichtigste und größte Klasse von Controller.IO. Die vorhandenen Methoden werden im Folgenden beschrieben.

AddElementToSvg(Date date, int x, int y): In die SVG-Datei des derzeit aktiven Visualisierungskonzepts wird ein Platzhalter eingefügt. Da es in C# keine Unterstützung für SVG gibt, wird in der Datei nach dem Element `</svg>` gesucht. Der Platzhalter wird vor `</svg>` per Image-Tag eingebunden. Jeder hinzuzufügende Platzhalter gehört zu dem Date-Objekt, das als Parameter `date` übergeben wird. Stellvertretend für den Platzhalter wird das grafische Element eingefügt, das als ActiveSnippet der SnippetClass, mit der das Date-Objekt annotiert ist, gespeichert ist. Die Parameter `x` und `y` geben die Position an, an der der Platzhalter eingefügt werden soll. Der Platzhalter nimmt eine Fläche von 50 x 50 Pixeln ein, mit der angegebenen Position als Mittelpunkt.

ChangeElementInSvg(Date date, VisElement oldElement) Ein bereits in der Svg-Datei des aktiven Visualisierungskozepts eingefügtes Element (`oldElement`) kann mit dieser Methode einem anderen Date-Objekt (`date`) zugeordnet werden. Falls das neue Date-Objekt mit einer anderen Annotation versehen wurde als das alte, wird die entsprechende Zeile in der Datei verändert, damit nun wieder das richtige Element angezeigt wird.

ChangeElementsForSnippetClass(SnippetClass selectedClass) Diese Methode wird verwendet, wenn der Benutzer für eine bestimmte Element-Klasse (`selectedClass`)

[11] `http://www.dotnetrdf.org`
[12] `https://www.gnu.org/licenses/gpl-3.0.html`

die grafischen Elemente austauschen will. Für alle betroffenen Elemente wird in der SVG-Datei die entsprechende Zeile dahin gehend geändert, dass das vom Benutzer ausgewählte Element angezeigt wird.

ConceptsToModel() Es werden die 'leeren' Vorlagen der Visualisierungskonzepte aus dem Ordner 'SVG' in den Ordner 'GraphicSnippets' kopiert. Anschließend werden die SVG-Dateien nach der Annotation, zu welchem Visualisierungskonzept welche Datei gehört, durchsucht. Die entsprechenden Informationen werden im Model abgespeichert.

ConvertSvg(string folderPath) Alle im Ordner mit dem Pfad `folderPath` liegenden SVG-Dateien werden zu PNG-Dateien konvertiert. Dazu wird das Kommandozeilen-Tool von Inkscape verwendet.

DeleteElementFromSvg(VisElement oldElement) Das Element `oldElement` wird aus der SVG-Datei des aktiven Visualisierungskonzepts entfernt.

SnippetsToModel(string snippetFolderPath) Diese Methode durchsucht einen angegebenen Ordner nach grafischen Elementen. Diese müssen als SVG-Dateien vorliegen und mit einer gültigen Annotation aus der Domänen-Ontologie versehen sein. Pfad und Annotation werden in Model.SvgSnippets gespeichert.

UpdateSvgToShowInWebBrowser() Da das PictureBox-Steuerelement keine SVG-Dateien anzeigen kann, werden die SVG-Dateien zu PNG-Dateien konvertiert. Diese Konvertierung funktioniert bei SVG-Dateien, in die andere SVG-Dateien eingebunden sind, nicht. Deshalb werden die grafischen Elemente als PNG eingebunden. Im Benutzer-Tab kommt ein WebBrowser-Steuerelement zum Einsatz, das SVG-Dateien anzeigen kann. Um die Vorteile von SVG nutzen zu können, ersetzt diese Methode alle PNG-Dateiendungen in der SVG-Datei durch SVG-Dateiendungen, so dass die SVG-Elemente eingebunden werden können.

UpdateSvgWhenAnnotationChanges(Date date) Wenn die Annotation eines Date-Objektes geändert wird, dann werden alle SVG-Dateien, in denen Elemente vorkommen, die zu diesem Date-Objekt gehören, angepasst, so dass Elemente der richtigen Klasse verwendet werden.

6.2. Benutzeroberfläche

Die Benutzeroberfläche besteht neben einem Menü und einer Toolbar aus zwei Tabs (siehe Abbildung 34 und 38), einen für das Erstellen der Visualisierungen und einen um diese Visualisierungen interaktiv optimieren zu können. Die beiden Tabs werden in den folgenden Abschnitten beschrieben. Die Benutzeroberfläche wurde mit Hilfe von Windows.Forms erstellt und setzt die View-Komponente der MVC-Architektur um.

6.2.1. Ersteller-Tab

Der Ersteller-Tab (siehe Abbildung 34) ist in zwei Bereiche, in der Abbildung durch die roten Kästen zu erkennen, unterteilt. Der linke Bereich (Bereich 1) ist eine PictureBox, in der eine Vorschau der Visualisierung angezeigt wird und in der Platzhalter hinzugefügt und entfernt werden können. Im rechten Bereich (Bereich 2), der wiederum in zwei Bereiche unterteilt ist, können die Annotationen der Daten und des Datensatzes bearbeitet werden. Die Annotationen der Daten können in einer Tabelle (siehe Abbildung 36) geändert werden. Diese Tabelle enthält pro Zeile eine DropDown-Box, in der eine der verfügbaren Annotationen ausgewählt werden kann. Um den Datensatz zu annotieren, wählt man die CheckBoxes aus, die für die zu verwendenden Visualisierungskonzepte stehen. Alle in diesem Tab ausgeführten Aktionen werden in der Rolle des Erstellers ausgeführt.

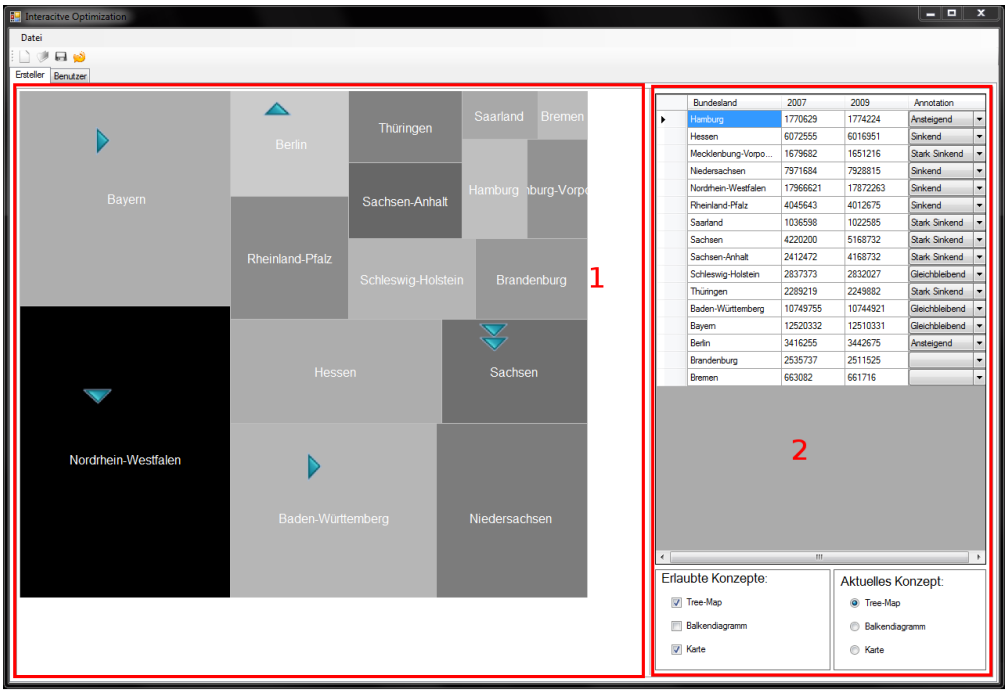


Abbildung 34: Zu sehen ist das Hauptfenster mit aktivem Ersteller-Tab. In diesem Beispiel ist eine Tree-Map zur Bearbeitung (1) und die Tabelle zum Annotieren der Bundesländer (2) zu sehen.

creatorPictureBox
Die creatorPictureBox dient als Vorschau der zu erstellenden Visualisierung. Gleichzeitig kann die Visualisierung darin bearbeitet werden. Die PictureBox in Windows.Forms kann keine SVG-Dateien anzeigen. Das einzige Windows.Forms-Steuerelement, das SVG-Dateien anzeigen kann, ist das WebBrowser-Steuerelement. Dieses kann hier

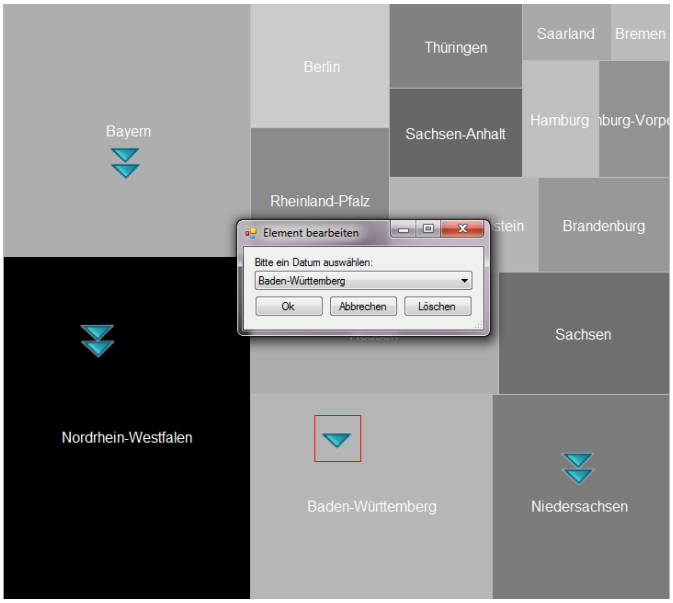
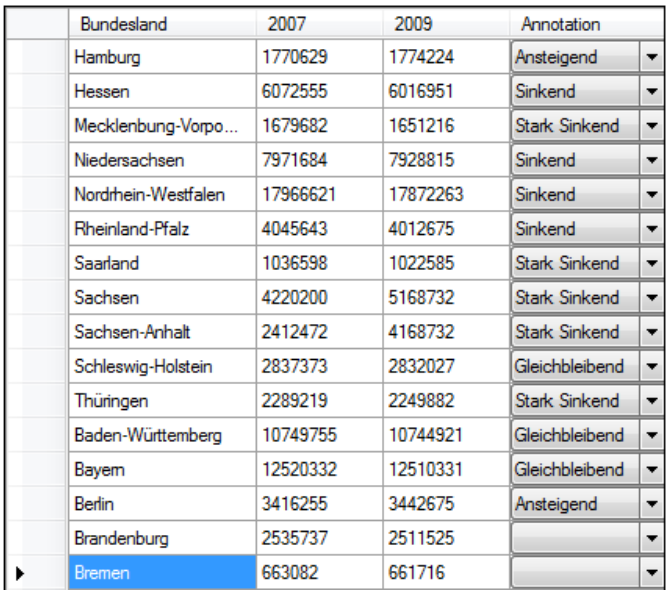


Abbildung 35: Zu sehen ist die creatorPictureBox. Aktuell ist das rot umrandete Element ausgewählt. Daher ist ein AddElementForm-Dialog geöffnet, in dem Baden-Württemberg ausgewählt ist.

nicht verwendet werden, da es einige Events nicht unterstützt. So ist es zum Beispiel bei einem Klick auf das Steuerelement nicht möglich, die Maus-Position zu erhalten. Dies ist aber elementar wichtig, damit der Ersteller die Positionen auswählen kann, an denen er Platzhalter hinzufügen möchte und um diese Platzhalter später wieder auswählen zu können. Aus diesem Grund wurde eine PictureBox verwendet. Um die Visualisierung darin anzeigen zu können, wird sie mit dem Kommandozeilen-Tool von Inkscape in eine PNG-Datei umgewandelt.
Durch einen Klick auf eine Position innerhalb der creatorPictureBox kann ein Platzhalter an dieser Position hinzugefügt werden. Platzhalter können für alle in Model.Dataset.Dates gespeicherten Date-Objekte, die annotiert wurden, hinzugefügt werden. Abbildung 35 zeigt eine Tree-Map-Visualisierung, die gerade bearbeitet wird. Den Hintergrund der Visualisierung (die eigentliche Tree-Map) stellt das Programm zur Verfügung, während die blauen Pfeil-Elemente durch den Ersteller hinzugefügt wurden. Aktuell hat der Ersteller auf den Pfeil im Feld Baden-Württemberg geklickt. Das Programm hebt den ausgewählten Platzhalter mit einem roten Rahmen hervor. Es öffnet sich ein Dialog vom Typ AddElementForm, in dem der Platzhalter entweder bearbeitet oder gelöscht werden kann. In der ComboBox wird der Key des zu diesem Element gehörenden Date-Objekts angezeigt. Auf diesem Weg kann der Ersteller herausfinden, welcher Platzhalter zu welchem Date-Objekt gehört. Welches konkrete grafische Element den Platzhalter repräsentiert, kann der Ersteller nicht direkt beeinflussen. Es wird ein beispielhaftes Element der SnippetClass angezeigt, mit der das zugehörige Date-Objekt annotiert ist. Welches Element dieser SnippetClass angezeigt werden soll, kann der Benutzer bestimmen.

Datentabelle



Bundesland	2007	2009	Annotation
Hamburg	1770629	1774224	Ansteigend
Hessen	6072555	6016951	Sinkend
Mecklenburg-Vorpo...	1679682	1651216	Stark Sinkend
Niedersachsen	7971684	7928815	Sinkend
Nordrhein-Westfalen	17966621	17872263	Sinkend
Rheinland-Pfalz	4045643	4012675	Sinkend
Saarland	1036598	1022585	Stark Sinkend
Sachsen	4220200	5168732	Stark Sinkend
Sachsen-Anhalt	2412472	4168732	Stark Sinkend
Schleswig-Holstein	2837373	2832027	Gleichbleibend
Thüringen	2289219	2249882	Stark Sinkend
Baden-Württemberg	10749755	10744921	Gleichbleibend
Bayern	12520332	12510331	Gleichbleibend
Berlin	3416255	3442675	Ansteigend
Brandenburg	2535737	2511525	
Bremen	663082	661716	

Abbildung 36: Datentabelle. Alle Bundesländer, außer Brandenburg und Bremen wurden bereits annotiert. 1.Spalte: Namen der Bundesländer, 2. Spalte: Bevölkerungszahl 2007, 3. Spalte: Bevölkerungszahl 2009, 4.Spalte: Bundesländer können annotiert werden.

Abbildung 36 zeigt die Datentabelle. In der Tabelle sind die 16 deutschen Bundesländer und ihre Bevölkerungszahlen aus den Jahren 2007 und 2009 zu sehen. Außer Brandenburg und Bremen wurden alle Bundesländer annotiert. Die Spalten 'Bundesland', '2007' und '2009' stammen aus der Domänen-Ontologie und zeigen die Bevölkerungsentwicklung der deutschen Bundesländer von 2007 bis 2009. In der Spalte 'Annotation' kann jede Zeile mit einer Annotation angereichert werden. Die in den ComboBoxen in der Spalte 'Annotation' auswählbaren Annotationen stammen aus der Domänen-Ontologie. Die zur Verfügung stehenden Annotationen stammen ebenfalls aus der Domänen-Ontologie. Der Visualisierung können nur für annotierte Zeilen Platzhalter hinzugefügt werden. Wenn der Ersteller eine Annotation in einer Zeile, für die es Platzhalter in der Visualisierung gibt, ändert, wird die Visualisierung angepasst. Das Anpassen wird in der zugehörigen SVG-Datei sofort vorgenommen. Dazu wird die Methode `UpdateSvgWhenAnnotationChanges()` aufgerufen. Da das Rendern der PNG-Datei für die creatorPictureBox einige Sekunden dauert, wird es erst bei einem Klick auf den Button 'Visualisierung neu laden' in der Toolbar durchgeführt.

Konzept-Panel Der letzte zu beschreibende Bereich des Ersteller-Tabs ist das Panel (Abbildung 37) rechts unten. Im linken Teil des Panels kann der Ersteller angeben, für welche Visualisierungskonzepte sich der aktuelle Datensatz eignet, in dem

er die entsprechenden CheckBoxes aktiviert. Intern wird der Datensatz jeweils mit der Annotation aus der Grafik-Ontologie für das vom Ersteller ausgewählte Visualisierungskonzept annotiert. In der Abbildung wurde der Datensatz also mit den beiden Annotationen `http://www.vis.uni-stuttgart.de/SemanticOptimization#Tree-Map` und `http://www.vis.uni-stuttgart.de/SemanticOptimization#Karte` annotiert.
Im rechten Teil des Panels kann der Ersteller auswählen, welches Visualisierungskonzept in der PictureBox zur Bearbeitung angezeigt werden soll.
Damit der Prototyp mit verschiedenen Domänen-Ontologien verwendet werden kann, werden sowohl die CheckBoxes im linken, als auch die RadioButtons im rechten Teil des Panels beim Start der Anwendung dynamisch erzeugt. Es werden stets die Labels der in der aktuellen Grafik-Ontologie vorkommenden Visualisierungskonzepte angezeigt. In Abbildung 37 enthält die Grafik-Ontologie drei Visualisierungskonzepte: Tree-Map, Balkendiagramm und Karte.

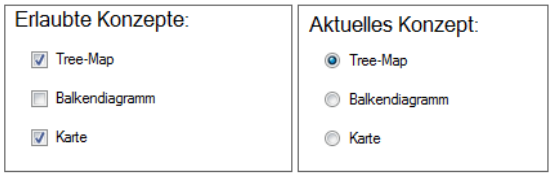


Abbildung 37: Im linken Bereich wurden Tree-Map und Karte als verwendbare Konzepte ausgewählt, im rechten ist Tree-Map ausgewählt, dadurch wird in der creatorPictureBox eine Tree-Map angezeigt.

6.2.2. Benutzer-Tab

Der Benutzer-Tab (Abbildung 38) ist, ähnlich wie der Ersteller-Tab, zweigeteilt. In der Abbildung ist dies durch die beiden roten Kästen verdeutlicht. Im linken Bereich (Bereich 1) befindet sich ein WebBrowser-Steuerelement, in dem die Visualisierung angezeigt wird. Im rechten Bereich (Bereich 2) kann der Benutzer einzelne grafische Elemente sowie das gesamte Visualisierungskonzept austauschen.
Da die grafischen Elemente nicht einzeln, sonder klassenweise ausgetauscht werden, ist es auf Benutzerseite nicht notwendig, die Mausposition in der Visualisierung zu bestimmen. Daher kann zur Anzeige ein WebBrowser-Steuerelement verwendet werden. In diesem können SVG-Dateien angezeigt werden. Das hat zwei Vorteile gegenüber der Anzeige als PNG in einer PictureBox:

1. Die Visualisierung muss nicht nach PNG konvertiert werden, was für einen erheblichen Geschwindigkeitsvorteil bei der Aktualisierung sorgt.

2. Die grafischen Elemente können als SVG eingebunden werden, wodurch sie deutlich besser dargestellt werden können.

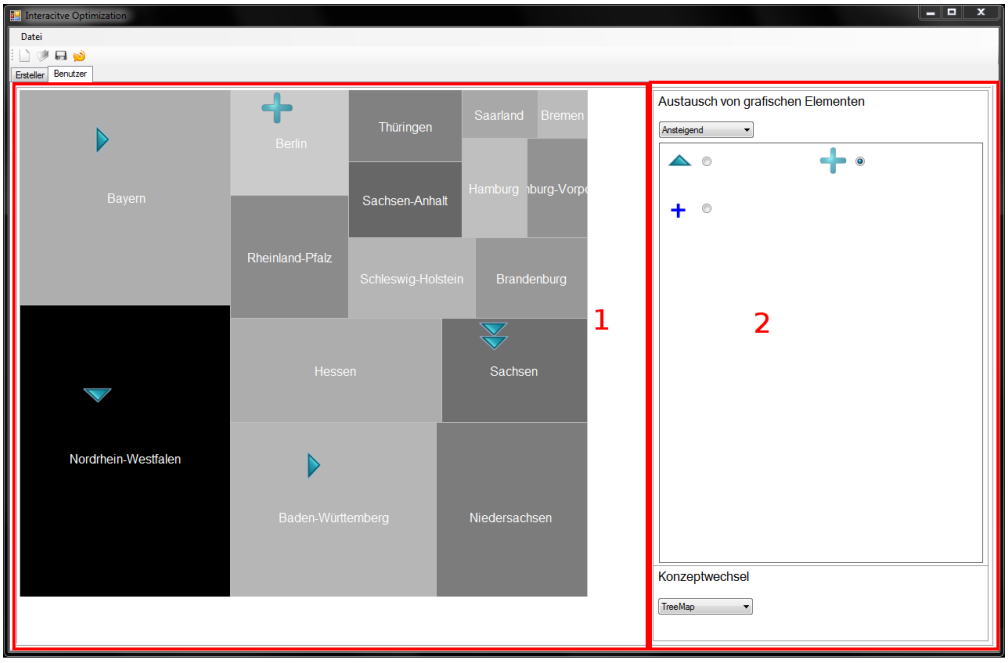


Abbildung 38: Zu sehen ist das Hauptfenster mit aktivem Benutzer-Tab. Tree-Map ist als Visualisierungskonzept ausgewählt (1), grafische Elemente für Annotation 'Ansteigend' können geändert werden (2).

Austausch grafischer Elemente

Abbildung 39 zeigt den Bereich des Benutzer-Tabs, in dem die grafischen Elemente ausgetauscht werden können. In der ComboBox kann der Benutzer die Element-Klasse auswählen, deren Elemente er austauschen will. Im Feld darunter kann er anschließend das passende Element auswählen. In Abbildung 39 hat der Benutzer entschieden, dass Elemente der Klasse 'Ansteigend' durch ein Plus dargestellt werden sollen. Die Auswahl an Element-Klassen stammt aus der Domänen-Ontologie. Es sind dieselben Klassen, mit denen der Ersteller die Bundesländer annotieren kann. Um alle verfügbaren

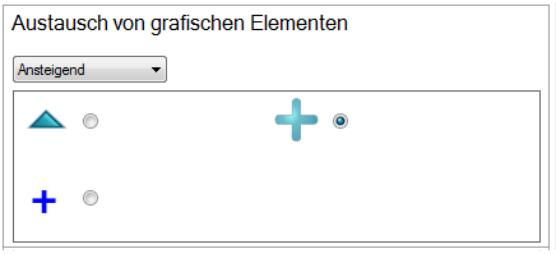


Abbildung 39: Bereich zum Austausch von grafischen Elementen. Aktuell wurde für die Klasse 'Ansteigend' ein Plus ausgewählt.

Elemente anzeigen zu können, wird bei Programmstart der Ordner 'GraphicSnippets' nach SVG-Dateien durchsucht, die mit den existierenden Element-Klassen annotiert wurden. Das Durchsuchen erfolgt durch den Controller, der dann die entsprechenden Objekte im Model anlegt. Die View-Komponente holt sich die benötigten Informationen aus dem Model und zeigt die verfügbaren Elemente an.

Auswahl des Visualisierungskonzepts
Mit der ComboBox in Abbildung 40 kann das Visualisierungskonzept ausgewählt werden. Es stehen die Visualisierungskonzepte zur Verfügung, mit denen der Ersteller den Datensatz annotiert hat.

Abbildung 40: Bereich zum Austausch des Visualisierungskonzepts

6.3. Sequenzdiagramme

In den beiden folgenden Abschnitten werden zwei grundlegende Funktionen des Programms anhand von Sequenzdiagrammen erläutert. Die Nummerierung der Aufzählungen entspricht den mit roten Zahlen markierten Bereichen in den Diagrammen.

6.3.1. Einfügen eines Platzhalters

Das Sequenzdiagramm in Abbildung 41 zeigt den internen Ablauf des Programms beim Hinzufügen eines Platzhalters zu einer Visualisierung durch den Ersteller.

1. Der Ablauf wird durch einen Klick des Erstellers in die PictureBox gestartet. Es wird der MouseDown-Event-Handler für die creatorPictureBox in MainForm aufgerufen. Dort wird als erstes überprüft, ob in der PictureBox im Moment eine Visualisierung angezeigt wird. Anschließend wird die Methode `MarkIfExists()` in der Klasse MainForm aufgerufen. In dieser Methode wird über Model.ActiveConcepts.VisElements die Liste der zum aktiven Visualisierungskonzept gehörenden Platzhalter geholt. Für jeden Platzhalter ist die Position in der Visualisierung gespeichert. So kann nun überprüft werden, ob der Ersteller auf einen bereits hinzugefügten Platzhalter geklickt hat oder nicht. Falls er auf einen existierenden Platzhalter geklickt hätte, würde der Platzhalter in der PictureBox mit einem roten Rahmen versehen. Dadurch soll verdeutlicht werden, auf welchen Platzhalter geklickt wurde. In dem im Sequenzdiagramm abgebildeten Ablauf hat der Ersteller auf eine Position in der Visualisierung geklickt, an der noch kein Platzhalter hinzugefügt worden ist. Aus diesem Grund wird in `MarkIfExists()` nur die Überprüfung, ob auf ein grafisches Element geklickt wurde, durchgeführt.

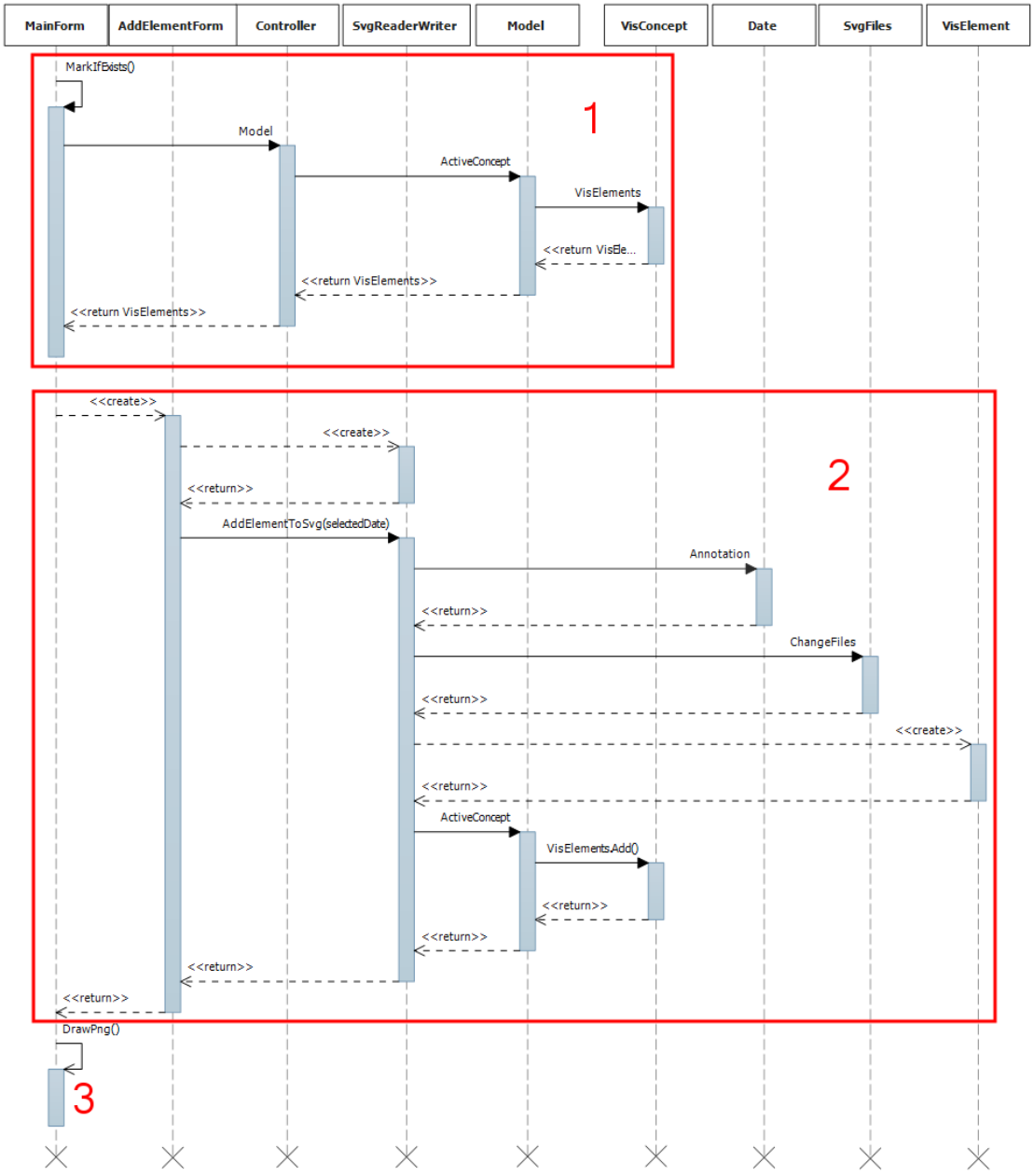


Abbildung 41: Das Einfügen eines Platzhalters erfolgt in drei Schritten: Überprüfen, ob ein vorhandener Platzhalter angeklickt wurde (1), Platzhalter in SVG-Datei einbinden (2), PNG neu zeichnen (3).

Anschließend wird ein Dialog vom Typ AddElementForm erzeugt und dem Ersteller angezeigt. In diesem muss der Ersteller ein Date-Objekt in einer ComboBox auswählen (die Date-Objekte werden durch ihren Key repräsentiert) und auf den Button OK klicken.

2. Als nächstes wird durch das AddElementForm-Objekt das SvgReaderWriter-Objekt `svg` erzeugt. Da ein neuer Platzhalter hinzugefügt werden soll, wird `svg.AddElementToSvg()` aufgerufen. Als Parameter wird das ausgewählte Date-Objekt sowie die Position des Mauszeigers beim Klick auf die PictureBox übergeben. In `AddElementToSvg()` wird dann die SnippetClass in Model.SnippetClasses gesucht, mit der das Date-Objekt annotiert ist. Anschließend wird die zum derzeit aktiven Visualisierungskonzept gehörende SVG-Datei bearbeitet. Das grafische Element wird per Image-Tag in der Zeile vor dem schließenden `</svg>` eingefügt. Listing 4 zeigt beispielhaft den SVG-Code, mit dem ein grafisches Element zur Visualisierung hinzugefügt wird. Zusätzlich wird ein neues VisElement-Objekt erzeugt, das zu Model.ActiveConcept.VisElements hinzugefügt wird. In diesem Objekt werden folgende Informationen gespeichert: Die Koordinaten der Fläche, die der Platzhalter in der Visualisierung belegt, das zugehörige Date-Objekt, die zugehörige SnippetClass und die Nummer der Zeile in der SVG-Datei, in der der Platzhalter hinzugefügt wurde. Anschließend wird das AddElementForm geschlossen.

3. Als letzter Schritt wird in MainForm die Methode `DrawPng()` aufgerufen. Darin wird die SVG-Datei als PNG gerendert und in der creatorPictureBox angezeigt.

```
<image x="581" y="288" width="50px" height="50px"
xlink:href="2downarrow.png">
<title>Stark_Ansteigend/title>
</image>
```

Listing 4: Image-Tag zum Einbinden eines grafischen Elements

6.3.2. Austauschen grafischer Elemente

Abbildung 42 zeigt den Programmablauf beim Austauschen grafischer Elemente zum Anpassen der Visualisierung durch den Benutzer an seine persönlichen Vorlieben und Bedürfnisse. Der Benutzer hat bereits eine SnippetClass ausgewählt und will nun alle Elemente dieser Klasse durch ein anderes Element ersetzen.

1. Zu diesem Zweck wählt er einen RadioButton neben dem gewünschten grafischen Element aus. Dadurch wird der zugehörige Event-Handler aufgerufen. Das Dictionary ActiveSnippets ordnet jeder SnippetClass ein SvgSnippet zu. Dieses wird dann verwendet um alle grafischen Elemente dieser SnippetClass in der Visualisierung zu repräsentieren. Im Event-Handler des RadioButtons wird dieses Dictionary aktualisiert, indem der SnippetClass das ausgewählte SvgSnippet zugeordnet wird.

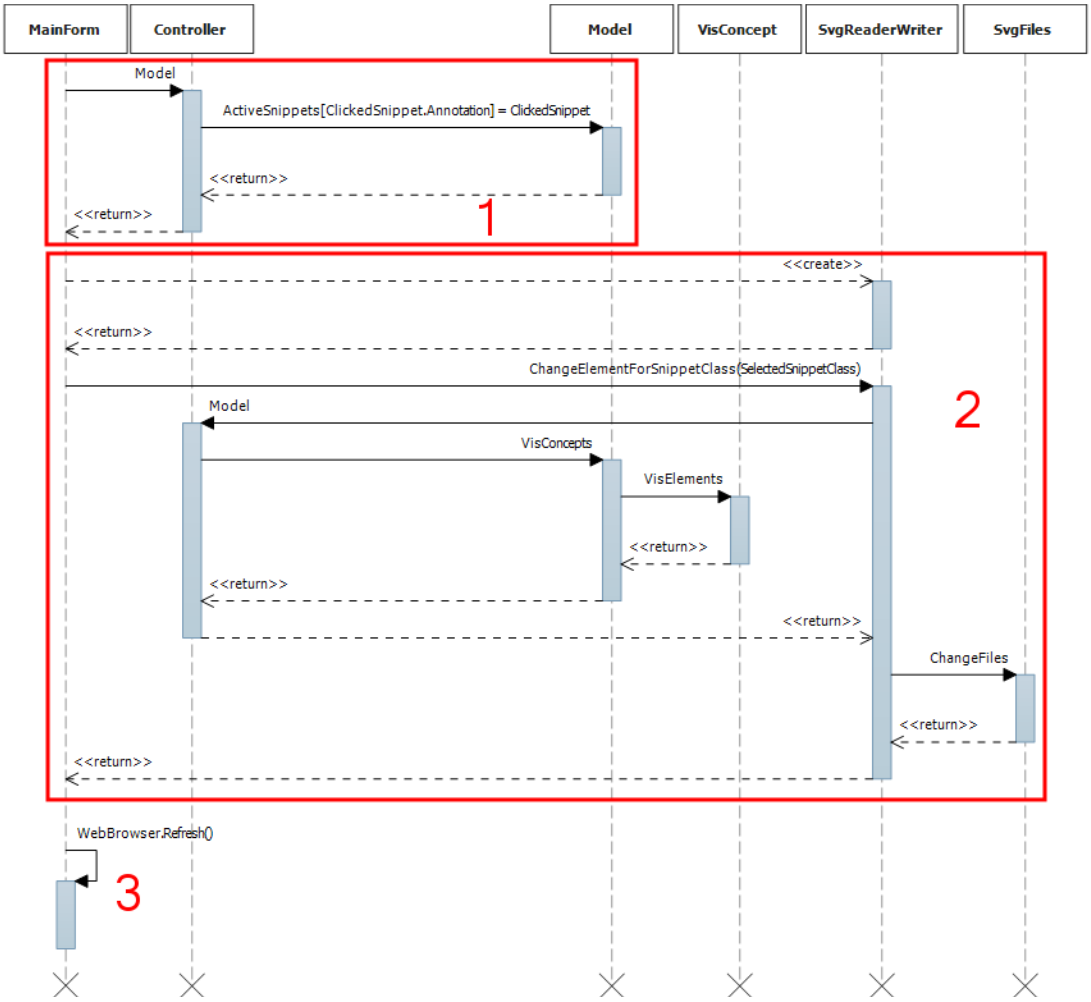


Abbildung 42: Der Austauschen grafischer Elemente erfolgt in drei Schritten: Dictionary *ActiveSnippets* im Model aktualisieren (1), SVG-Datei anpassen (2), WebBrowser aktualisieren (3).

2. Als nächstes wird das SvgReaderWriter-Objekt `svg` erzeugt und die Methode `svg.ChangeElementsForSnippetClass()` aufgerufen. Als Parameter wird die SnippetClass übergeben. Es wird aus dem Model die Liste mit allen Platzhaltern geholt. Für jeden dieser Platzhalter wird überprüft, ob der Platzhalter zur übergebenen SnippetClass gehört. Diese Überprüfung erfolgt anhand der aus der Domänen-Ontologie stammenden Uri der SnippetClass. Bei erfolgreicher Überprüfung wird die SVG-Datei der im Moment aktiven Visualisierung geändert. Das Image-Tag, mit dem das Element hinzugefügt wurde, bleibt grundsätzlich erhalten, nur der Dateiname wird ausgetauscht. Da jedes VisElement in der Eigenschaft `LineInFile` die Zeilennummer, an der es in der SVG-Datei eingefügt wurde, gespeichert hat, kann die entsprechende Zeile leicht gefunden werden.

3. Abschließend wird das WebBrowser-Steuerelement, in dem die Visualisierung angezeigt wird, aktualisiert.

6.4. Systemvoraussetzungen

Dieser Abschnitt erläutert, was benötigt wird, um den Prototypen verwenden zu können und in welcher Ordnerstruktur die Komponenten vorliegen müssen. Es werden die grafischen Elemente, die Visualisierungshintergründe und die Ontologien benötigt. Die folgenden Unterabschnitte erläutern die Komponenten im Detail. Alle Komponenten werden mit dem Prototypen ausgeliefert, können jedoch auch ersetzt werden.

6.4.1. Ordnerstruktur

Im Programmordner müssen die Ordner 'bin', 'GraphicSnippets', 'Ontologies', 'Save' und 'SVG' vorhanden sein. Der Ordner 'bin' enthält einen Ordner namens 'Release', in dem die ausführbare Exe-Datei und die benötigten DLLs liegen. Der Ordner 'Save' wird benötigt, um erstellte Visualisierungen speichern zu können. Der Inhalt der weiteren Ordner wird in den folgenden Abschnitten beschrieben.

6.4.2. Grafische Elemente

Die grafischen Elemente, die dem Benutzer bei der Anpassung der Visualisierungen zur Verfügung stehen sollen, müssen beim Programmstart im Ordner 'GraphicSnippets' vorliegen. Sie müssen im SVG-Format gespeichert sein und mit einer gültigen Annotation aus der Domänen-Ontologie annotiert sein. Listing 5 zeigt die Annotation eines grafischen Elements mit der Tendenz 'Sinkend'. Das bedeutet, dass dieses grafische Element verwendet werden kann, um Bundesländer zu repräsentieren, deren Bevölkerungswachstum sinkend ist.

```
<semantics>
    http://www.vis.uni-stuttgart.de/GermanStates\#Sinkend
</semantics>
```

Listing 5: Annotation eines grafischen Elements mit der Tendenz 'Sinkend'

Die Annotation wird innerhalb eines `<semantics>`-Tags angegeben. Da dieser Tag nicht Teil der SVG-Spezifikation ist, wird er durch SVG-Renderer einfach ignoriert. Wenn zusätzliche Elemente verwendet werden sollen, müssen sie im Programm mit der Funktion 'Grafische Elemente konvertieren' zu PNG-Dateien konvertiert werden, da die Elemente für die Anzeige in der Ersteller-PictureBox als PNG benötigt werden.

6.4.3. Visualisierungshintergründe

Die zu verwendenden Visualisierungshintergründe müssen als SVG-Dateien im Ordner 'SVG' liegen. Falls ein Hintergrund nur als Rastergrafik vorliegt, kann er einfach in eine leere SVG-Datei eingebunden werden. Die Hintergründe müssen mit einer Unterklasse

der Klasse 'Type' aus der Grafik-Ontologie annotiert sein. Diese Annotation dient dem Prototypen zur Zuordnung von Hintergrundbild und VisConcept.

6.4.4. Ontologien

Sowohl die Grafik- als auch die Domänen-Ontologie müssen als RDF-Datei vorliegen. Die beiliegenden Ontologien wurden mit Protégé 3.4.7 erstellt. Falls die Ontologien mit anderen Editoren oder anderen Versionen erstellt wurden, muss die Kompatibilität überprüft werden. Listing 6 zeigt einen beispielhaften Ausschnitt der Domänen-Ontologie. Es sind die drei Tendenzen 'Ansteigend', 'Gleichbleibend' und 'Sinkend' sowie die beiden Bundesländer Hamburg und Hessen enthalten. Die Ontologien müssen im Ordner 'Ontologies' gespeichert sein.

```
<rdf:RDF xmlns:rdf="&rdf;"
         xmlns:GermanStates="&GermanStates;"
         xmlns:rdfs="&rdfs;">
<GermanStates:Bevölkerungsentwicklung rdf:about="&GermanStates;Ansteigend"
         GermanStates:Label="Ansteigend"
         rdfs:label="Ansteigend"/>
<GermanStates:Bevölkerungsentwicklung rdf:about="&GermanStates;Gleichbleibend"
         GermanStates:Label="Gleichbleibend"
         rdfs:label="Gleichbleibend"/>
<GermanStates:Bevökerungsentwicklung rdf:about="&GermanStates;Sinkend"
         GermanStates:Label="Sinkend"
         rdfs:label="Sinkend"/>
<GermanStates:State rdf:about="&GermanStates;domain_Class10"
         GermanStates:Einwohner_2007="1770629"
         GermanStates:Einwohner_2009="1774224"
         GermanStates:Label="Hamburg"
         rdfs:label="Hamburg"/>
<GermanStates:State rdf:about="&GermanStates;domain_Class11"
         GermanStates:Einwohner_2007="6072555"
         GermanStates:Einwohner_2009="6016951"
         GermanStates:Label="Hessen"
         rdfs:label="Hessen"/>
</rdf:RDF>
```

Listing 6: Ausschnitt aus der Domänen-Ontologie. Zu sehen sind die Elemente für die Bundesländer Hamburg und Hessen, sowie die Tendezen 'Ansteigend', 'Gleichbleibend' und 'Sinkend'

7. Evaluierung

Das übergeordnete Ziel dieser Diplomarbeit war es, den Gulf of Evaluation im Bereich von Visualisierungen zu verkleinern. In diesem Kapitel wird evaluiert, ob die gewählten Lösungsansätze geeignet waren, um dieses Ziel zu erreichen.
Es wurde ein Konzept entworfen, mit dem Visualisierungen mit semantischen Metainformationen annotiert werden können, damit sie anschließend durch einen Benutzer interaktiv optimiert werden können. Nach der Konzeptionsphase wurde ein Prototyp implementiert, der zeigt, dass sich das Konzept auch praktisch umsetzen lässt. Dabei kam es zu folgenden Schwierigkeiten:

- Mit Hilfe der Grafik-Ontologie kann beschrieben werden, aus welchen grafischen Primitiven Visualisierungskonzepte bestehen können. Allerdings ist es nicht ohne Weiteres möglich, aus diesen Informationen eine Visualisierung zu erstellen, da die Informationen über die geometrische Anordnung der grafischen Primitive fehlen. Daher benötigt das Visualisierungsprogramm für jedes Visualisierungskonzept ein Hintergrundbild.

- Schwierigkeiten machte auch die nicht vorhandene Unterstützung der Programmiersprache C# für Ontologiesprachen. Da nur für RDF eine Bibliothek zur Verarbeitung von Ontologien gefunden werden konnte, mussten die Ontologien in RDF erstellt werden. RDF ist, im Gegensatz zu OWL, nur für die Modellierung einfacher Sachverhalte verwendbar. Für die Ontologien, die für diese Diplomarbeit benötigt wurden, reichen die Modellierungsmöglichkeiten von RDF jedoch aus.

Die Entscheidung, SVG zur Erstellung und Darstellung der Visualisierungen zu verwenden muss ambivalent betrachtet werden:
SVG hat sich sehr gut geeignet, um Visualisierungen und auch einzelne Elemente davon mit semantischen Metainformationen anzureichern.
Da C# keine Unterstützung zur Bearbeitung von SVG-Dateien bietet, mussten die dafür benötigten Methoden selbst geschrieben werden.
Die Entscheidung, keinen eigenen SVG-Renderer zu schreiben, sparte viel Zeit bei der Implementierung, die für die Erarbeitung des theoretischen Konzepts verwendet werden konnte. Das WebBrowser-Steuerelement ist in C# die einzige Möglichkeit, SVG-Dateien anzuzeigen. Da sich darin aber die Mausposition nicht bestimmen lässt, konnte es zum Erstellen und Annotieren der Visualisierungen nicht verwendet werden. Aus diesem Grund müssen die SVG-Dateien in PNG-Dateien konvertiert werden, um sie dann in einem PictureBox-Steuerelement anzuzeigen. Das führt zu einem erheblichen Performanceverlust. Außerdem können die Vorteile von SVG-Dateien nicht genutzt werden. Da bei der Optimierung der Visualisierung die Mausposition nicht benötigt wird, kann auf Benutzerseite das WebBrowser-Steuerelement verwendet werden.

Als vorteilhaft hat sich die Verwendung des in Abschnitt 2.4 vorgestellten Schemas herausgestellt. Ohne dieses Schema wäre vor allem der Einstieg in die Ontologieerstellung schwierig geworden.

Insgesamt konnte in der vorliegenden Diplomarbeit ein Konzept zur interaktiven, verständnisorientierten Optimierung von semantisch-annotierten Visualisierungen entworfen und dessen Umsetzbarkeit mittels eines Prototypen gezeigt werden. Abbildung 43 zeigt eine erweiterte Version der Abbildung 11. Der hinzugefügte, grüne Pfeil ist das Ergebnis des in dieser Diplomarbeit erarbeiteten Konzepts. Der Nutzer kann dadurch die Visualisierung so lange anpassen, bis er sie richtig verstanden hat. Durch die interaktive Optimierbarkeit können einmal erzeugte Visualisierungen durch verschiedene Menschen gut verstanden werden. Das Ziel, den Gulf of Evaluation im Bereich Visualisierung zu verringern konnte also erreicht werden.

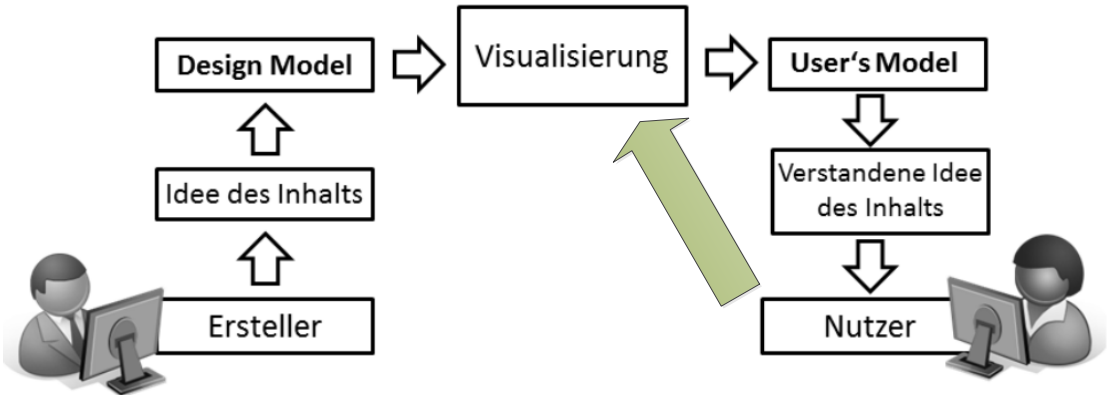


Abbildung 43: Der Grafik aus dem Abschnitt 2.6.3 konnte der grüne Pfeil hinzugefügt werden. Der Nutzer kann nun die Visualisierung so lange anpassen, bis er sie richtig verstanden hat.

8. Zusammenfassung und Ausblick

Übergeordnetes Ziel der Diplomarbeit „Interaktive, verständnisorientierte Optimierung von semantisch-annotierten Visualisierungen“ ist es, den Gulf of Evaluation im Bereich von Visualisierungen zu verkleinern. Um dieses Ziel zu erreichen, sollte ein Konzept entworfen werden, mit dem Visualisierungen mit semantischen Metainformationen annotiert werden können, um sie interaktiv optimieren zu können. Zusätzlich sollte die grundsätzliche Umsetzbarkeit des Konzeptes durch Implementierung eines Prototypen gezeigt werden.
Um die Visualisierungen annotieren zu können, wurden zwei Ontologien entworfen. Eine der beiden, die Grafik-Ontologie, beschreibt verschiedene Visualisierungskonzepte und die atomaren Bausteine, aus denen die Visualisierungskonzepte bestehen können. Die andere, die Domänen-Ontologie, dient dazu, den Inhalt von Visualisierungen mit semantischen Metainformationen zu annotieren und muss für den jeweiligen Anwendungsfall erstellt werden. Es wurde beispielhaft eine Domänen-Ontologie für das Einsatzszenario Bevölkerungsentwicklung erstellt. Anschließend wurde ein Interaktionskonzept entworfen, mit dem Visualisierungen annotiert und anschließend interaktiv optimiert werden können.

Sowohl das in dieser Diplomarbeit vorgestellte Konzept als auch der Prototyp könnten in Zukunft noch deutlich erweitert werden. In diesem Abschnitt werden einige der möglichen Erweiterungen beschrieben.
Die Wunschvorstellung wäre eine Software, die automatisch Visualisierungen in verschiedenen Visualisierungskonzepten erstellt. Dies müsste auf Basis einer um beliebige Visualisierungskonzepte erweiterten Grafik-Ontologie und einer Domänen-Ontologie für eine beliebige Anwendungsdomäne geschehen. Auf diesen Grundlagen könnten Daten visualisiert werden. Damit das erreicht werden kann, müssten die Ontologien deutlich erweitert werden. Vor allem müsste das Problem gelöst werden, wie aus den Informationen der Ontologien die genaue geometrische Anordnung der Elemente einer Visualisierung abgeleitet werden kann. Zusätzlich müssten Richtlinien definiert werden, die festlegen, welche Informationen erweiterte Ontologien enthalten müssen, damit die Software daraus automatisch Visualisierungen erzeugen kann. In diesen Visualisierungen könnten dann alle Elemente annotiert und ausgetauscht werden, nicht nur durch den Ersteller hinzugefügte Platzhalter.

Eine mögliche Erweiterung für den Prototypen wäre es, zu ermöglichen, dass Platzhalter mit Detailstufen annotiert werden können. Wie das konzeptionell umgesetzt werden kann ist in Abschnitt 4.2.4 beschrieben. Die Implementierung der Detailstufen wurde aus Zeitgründen nicht durchgeführt.

Eine weitere Möglichkeit zur Erweiterung des Prototypen wäre es, dass grafische Elemente nicht nur klassenweise, sondern auch einzeln durch den Benutzer, ausgetauscht werden können. Dann könnte man eine Visualisierung beispielsweise dahingehend verändern, dass die steigende Bevölkerungszahl eines Bundeslandes durch einen Pfeil

dargestellt wird und die eines anderen Bundeslandes durch ein Plus. Allerdings sollte vorher untersucht werden, ob dies das Verständnis des Benutzers erleichtert, oder ob es zu zusätzlicher Verwirrung führt.

Eine Möglichkeit, das Konzept zu erweitern wäre, dass grafische Elemente in verschiedenen Farben verwendet werden können. Dazu müsste die Grafik-Ontologie um die gewünschten Farben erweitert werden. Dann könnten eingefügte Platzhalter mit einer Farbe annotiert werden. Elemente, die diesen Platzhalter ersetzen können, müssten dieselbe Farbe haben.

Für den Fall, dass sich das Sematic Web durchsetzen wird, dürften sich in Zukunft deutlich mehr Daten und Informationen finden lassen, die bereits mit semantischen Metainformationen annotiert sind. Wenn diese Informationen genutzt werden könnten, müsste nicht für jeden Anwendungsfall eine eigene Domänen-Ontologie entworfen werden. Da sich das Semantic Web jedoch auch über zehn Jahre nach der Veröffentlichung durch Tim Berners-Lee noch nicht durchgesetzt hat, bleibt abzuwarten, ob der große Durchbruch noch gelingen wird.
Viele der heutzutage erstellten Daten werden automatisch generiert. Daher wäre es möglich, die Daten bei der Erstellung automatisch zu annotieren. Dies wäre eine weitere Möglichkeit, um im großen Stil annotierte Daten zu erhalten.

Interessant wäre außerdem zu überprüfen, ob sich das vorgestellte Konzept auch für Benutzeroberflächen eignen würde. Damit könnte vielleicht ein entscheidender Schritt in Richtung automatisch generierbarer Benutzeroberflächen getan werden.

A. Grafik-Ontologie

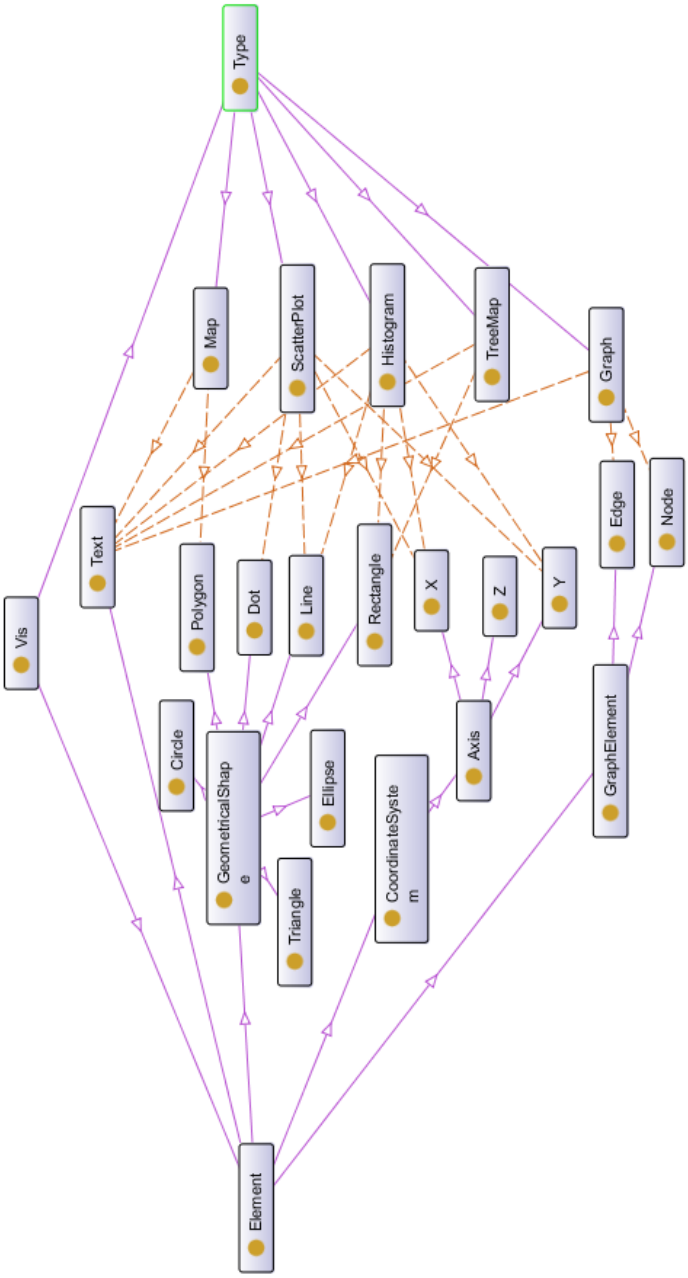


Abbildung 44: Die gesamte Grafik-Ontologie. Die durchgezogenen, violetten Pfeile stehen für 'Ist-Unterklasse-Von', die gestrichelten, orangen Pfeile für 'has-Element'. Sie ordnen den Visualisierungskonzepten die grafischen Primitive zu, aus denen sie bestehen.

B. Domänen-Ontologie

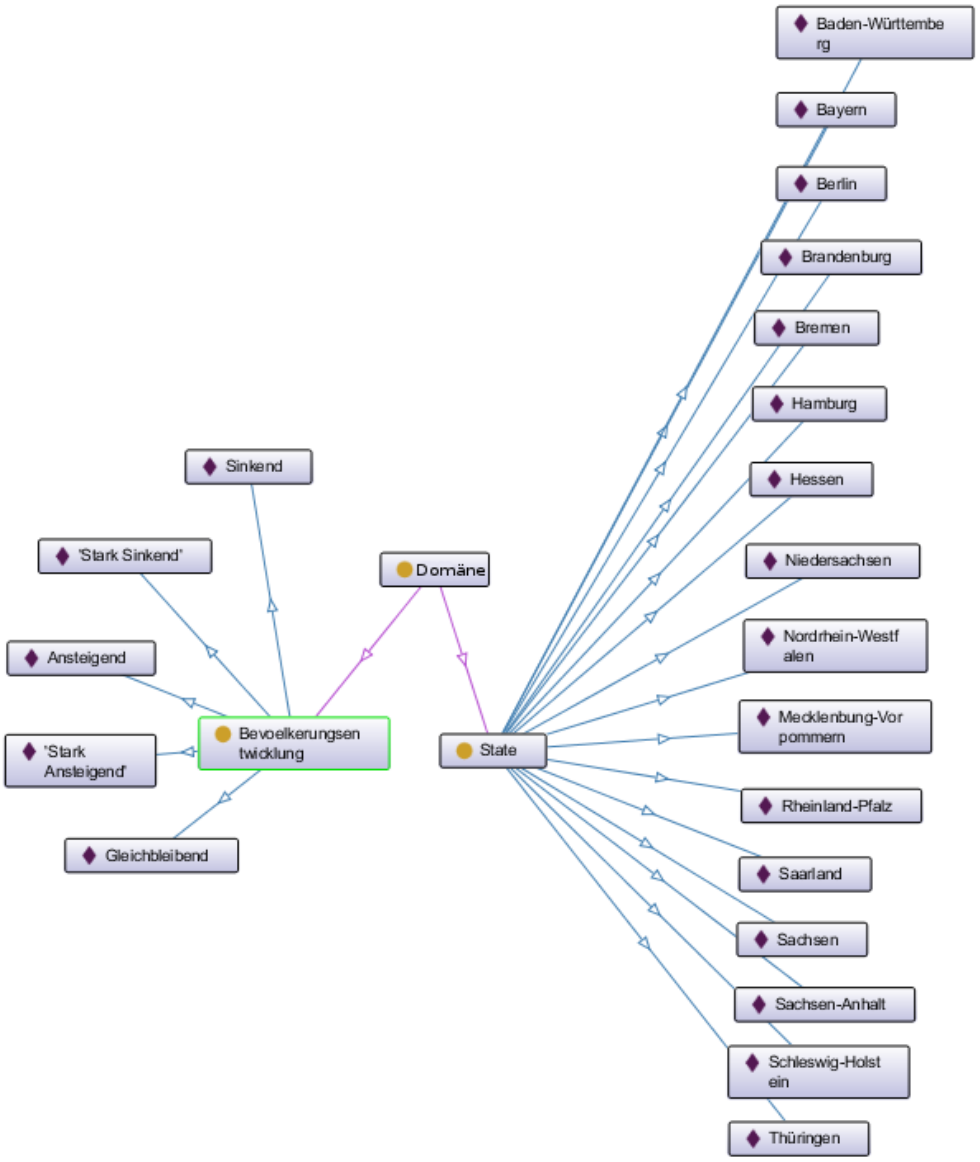


Abbildung 45: Die Domänen-Ontologie. Die violetten Pfeile stehen für 'Ist-Unterklasse-Von', die blauen Pfeile für 'Ist-Instanz-Von'. Zu jedem Bundesland gehören noch die Bevölkerungszahlen aus den Jahren 2007 und 2009, die aber aus Gründen der Übersichtlichkeit weggelassen wurden.

Abbildungsverzeichnis

Literatur

[Con09] World Wide Web Consortium. Owl 2 web ontology language. http://www.w3.org/TR/owl2-overview/, 2009.

[Ert11] Thomas Ertl. Visualization. 2011.

[Gam09] E. Gamma. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley professional computing series. Addison-Wesley, 2009.

[GPCFL04] Asuncion Gomez-Perez, Oscar Corcho, and Mariano Fernandez-Lopez. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, July 2004.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5:199–220, June 1993.

[Hor07] Ian Horrocks. Semantic web: the story so far. In *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, W4A '07, pages 120–125, New York, NY, USA, 2007. ACM.

[htta] http://www.duden.de/rechtschreibung/Annotation.

[httb] http://www.duden.de/rechtschreibung/Ontologie.

[httc] http://www.duden.de/rechtschreibung/Semantik.

[httd] http://www.webopedia.com/TERM/V/vector_graphics.html.

[JG11] David Reinsel John Gantz. Extracting value from chaos. http://www.emc.com/digital_universe 2011.

[KP88] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26 – 49, August 1988.

[map] maps.google.com.

[Mar06] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49:41–46, April 2006.

[MR11] Thomas Ertl Michael Raschke, Philipp Heim. Interaktive verständnisorientierte optimierung von semantisch-annotierten visualisierungen. 2011.

[Nor83] Donald Norman. Some observations on mental models. pages 7–14, 1983.

[Nor86] Donald Norman. *User Centered System Design*. Lawrence Erlbaum Associates, 1986.

[PH08a] Sebastian Rudolph York Sure Pascal Hitzler, Markus Krötzsch. Semantic web - grundlagen. pages 1–10, 2008.

[PH08b] Sebastian Rudolph York Sure Pascal Hitzler, Markus Krötzsch. Semantic web - grundlagen. pages 125 – 155, 2008.

[RBH90] D.A.McNabb R. B. Haber. Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in Scientific Computing*, pages 74–93, 1990.

[RF94] W. Ribarsky and J.D. Foley. Next-generation data visualization tools. pages 102–127, 1994.

[Sch09] Thomas Schlegel. Usability and interaction. 2009.

[Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, IEEE Symposium on*, 0:336, 1996.

[Stu09] Heiner Stuckenschmidt. Erstellen von ontologien. In *Ontologien: Konzepte, Technologien und Anwendungen*, Informatik im Fokus. Springer Berlin Heidelberg, 2009.

[TBL01] Ora Lassila Tim Berners-Lee, James Hendler. The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2001.

[w3c] http://www.w3c.org.

[wcw] www.rockingteam.com/wp-content/uploads/2009/05/semantic web.gif.

[YHC03] Tyng-Ruey Chuang Yi-Hong Chang. Embedding domain semantics in svg, 2003.

Alle Online-Quellen wurden zuletzt am 18.01.2011 geprüft.

Erklärung

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

(Hannes Pfannkuch)