Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Visualisierung von Ontologien für MUSE4Anything

Jonas Baireuther

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Dr. h. c. Frank Leymann
Betreuer/in:	Fabian Bühler, M.Sc. Daniel Vietz, M.Sc.
Beginn am:	26. Mai 2021
Beendet am:	26. November 2021

Kurzfassung

Die vorliegende Arbeit stellt eine Visualisierung für die Ontologien in MUSE4Anything vor. MUSE4Anything ist ein generisches Werkzeug der Werkzeugumgebung Muster Suchen und Erkennen (MUSE). Mit diesem Werkzeug lassen sich Wissensrepräsentationen des Seienden in die Informatik übertragen um darin noch unerkannte Zusammenhänge zu erkennen. Diese Definition des Wissens erfolgt mit Hilfe von Ontologien und Taxonomien. Um bei diesen einen guten Überblick zu behalten und Relationen zu verstehen ist eine Visualisierung eine hilfreiche Möglichkeit. Für diese Visualisierung werden verschiedene Varianten vorgestellt und untersucht. Anschließend werden die Anforderungen an eine solche Visualisierung vorgestellt. Im letzten Schritt wird eine erste Version der Anforderungen implementiert, um einen Grundstein für die Ontologievisualisierung in MUSE4Anything zu setzen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface. 56

CSS Cascading Style Sheets. 61

EU Europäische Union. 34

FA Finale Anforderung. 47

GUI Graphical User Interface. 53

HATEOAS Hypermedia as the Engine of Application State. 56

HTTP Hypertext Transfer Protocol. 56

IMT Inferential Modeling Technique. 35

MUSE Muster Suchen und Erkennen. 2

PDF Portable Document Format. 69

PNG Portable Network Graphics. 69

SOVA Simple Ontology Visualization API. 41

SVG Scalable Vector Graphics. 69

UI User Interface. 33

UML Unified Modeling Language. 14

VOWL Visual Notation for OWL Ontologies. 41

1. Einleitung

1.1. Motivation

Im Rahmen der Digital-Humanities-Projekte MUSE [Bar18], MUSE4Music [BBE+17] und MUSE4Anything [Büh21] werden unter anderem reale Objekte und Daten auf Ontologien abgebildet. Alle drei Projekte haben als Grundlage Ontologien [Bar], welche die Struktur der Daten bestimmen.

MUSE und MUSE4Music haben einen bestimmten Zielkontext. Hier wurde die Ontologie vor der Entwicklung in einer Mind-Map Anwendung erstellt. Diese wurde anschließend während der Entwicklungsphase direkt mit integriert. Daher existiert für diese Ontologien eine rudimentäre Visualisierung.

MUSE4Anything hat keinen bestimmten Zielkontext. Das Ontologie-Management ist ein Teil von MUSE4Anything. Die Ontologien in MUSE4Anything werden zur Laufzeit mit einer Eingabemaske definiert. Diese baut auf einer JSON Schema Definition auf. Die Ontologie wird in einem generischen Datenbankschema abgespeichert. Da dies eine rein textliche Darstellung ist können hier unter anderem die Zusammenhänge einzelner Elemente nicht schnell erkannt werden, was wiederum eine höhere Hürde der Verständlichkeit mit sich bringt. Daher ist eine graphische Visualisierung hilfreich.

1.2. Ziel der Arbeit

Ziel der Masterarbeit ist die Entwicklung einer grafischen Visualisierung der in MUSE4Anything verwalteten Ontologien, inklusive der enthaltenen Taxonomien. Diese Darstellung soll für eine intuitive Verständlichkeit der Datenzusammenhänge bei den Wissenschaftlern der Digital Humanities sorgen und damit deren Arbeit erleichtern.

Katifori et al. schreiben, dass es nicht einfach ist eine Visualisierung zu erstellen, die alle Informationen effektiv darstellt [KHL+07, S. 2]. Deshalb werden in dieser Arbeit verschiedene Ontologie-Darstellungstools auf deren unterschiedliche Darstellungen untersucht und nach wissenschaftlichen Kriterien zur Bewertung von Visualisierungen verglichen. Diese Ergebnisse bilden die Grundlage für die Implementierung der Visualisierung in MUSE4Anything.

1.3. Gliederung

Kapitel 2 - Hintergrund und Grundlagen: Am Anfang der Arbeit werden grundlegende Begriffe für das Verständnis der Arbeit erörtert.

Kapitel 3 - Anforderungen an die Visualisierung: In diesem Kapitel werden auf Basis einer Literaturrecherche Anforderungen an die Visualisierung erarbeitet.

Kapitel 4 - Toolvergleich: In diesem Kapitel werden verschiedene Ontologievisualisierungen analysiert und mit den Anforderungen aus dem vorherigen Kapitel in Bezug gesetzt. Anschließend werden die finalen Anforderungen an die Visualisierung definiert.

Kapitel 5 - Prototyp der Visualisierung: Dieses Kapitel erstellt aus den erarbeiteten Anforderungen Mockups und anschließend wird ein erster Prototyp implementiert.

Kapitel 6 - Zusammenfassung und Ausblick: Die Ergebnisse der Arbeit werden in diesem Kapitel zusammengefasst und weitere Schritte vorgestellt.

2. Hintergrund und Grundlagen

Das folgende Kapitel erläutert das dieser Arbeit zugrundeliegenden Werkzeug Muster Suchen und Erkennen (MUSE) und die darauf basierenden Projekte MUSE4Music und MUSE4Anything. Daneben werden noch Ontologien und Taxonomien erläutert, da diese die Datengrundlage für MUSE bilden.

2.1. MUSE und MUSE4Music

MUSE ist ein neuer Ansatz für die Identifikation und Extraktion von Mustern in Kostümen und den dazu benötigten Werkzeugen. Es geht darum, die verschiedenen Einsatzmöglichkeiten von Kostümen in Filmen zu erfassen, damit daraus Konventionen ermittelt werden können, die in Form von Mustern dokumentiert werden und diese Kostümbildnern zugutekommen.

Mit Mustern lassen sich bewährte Lösungenansätze, in diesem Fall Datenstrukturen, als wiederverwendbare Vorlage definieren. Dieser Ansatz ist nicht auf einen bestimmten Bereich begrenzt, sondern kann für jegliches Wissen eingesetzt werden [AAI+77]. Somit sind gleiche Daten immer nach derselben Struktur abgelegt und können weiterverarbeitet werden. Um diese Muster in der digitalen Welt zu repräsentieren und verarbeitbar zu machen, wird das Konzept der Ontologien (Abschnitt 2.3) verwendet [Bar18]. Mit MUSE4Music wird das MUSE Konzept auf Musikstücke aus dem 19. Jahrhundert angewendet, um bisher unerkannte Ähnlichkeiten in der Musik in verschiedenen Stücken zu entdecken [BBE+17].

2.2. MUSE4Anything

MUSE4Anything ist die generische Weiterentwicklung von MUSE und MUSE4Music. MUSE und MUSE4Music haben, wie im vorherigen Abschnitt beschrieben, für die jeweilige Ontologie ein speziell angepasstes Datenbankschema. Daneben ist die Ontologie auch direkt in der Entwicklung in das Backend und bei MUSE auch ins Frontend eingeflossen. Diese Ontologien wurden teilweise mithilfe von Mind-Map Anwendungen erstellt. Das Problem hierbei ist das sehr statische Datenbankschema, da dieses während der Entwicklungsphase fest in die jeweilige Anwendung implementiert wurde. Sobald sich an der Ontologie etwas ändert, wird eine neue Version der Anwendung benötigt. Zudem sind die Anwendungen jeweils nur für ihren speziellen Anwendungsfall verwendbar. Um diese beiden Probleme zu lösen, wurde MUSE4Anything entwickelt. MUSE4Anything bietet ein generisches Datenbankschema inklusive Versionsverwaltung der einzelnen Formate. Die Datenformate werden mithilfe einer Eingabemaske in MUSE4Anything definiert. Mit dieser Eingabemaske ist immer nur ein Element der Ontologie zu sehen. Hierbei fehlt aber die Visualisierung der Ontologie, was bei MUSE und MUSE4Music durch die Definition in einer Mind-Map Anwendung gegeben war. [Büh21]

Es ist durchaus möglich die in MUSE4Anything definierten Ontologien und Taxonomien mit in einer externen Mind-Map Anwendung, in welcher die Daten visuell dargestellt sind, synchron zu halten. Dies ist aber sehr zeitaufwendig und auch fehleranfällig. Daher ist die externe Verwendung einer existierenden Mind-Map Anwendung keine Lösung.

Die in dieser Masterarbeit erarbeitete und implementierte Visualisierung liefert den fehlenden Baustein der Visualisierung der Ontologie in MUSE4Anything und bietet somit den Benutzern einen besseren Überblick auf die definierten Ontologien.

2.3. Ontologien und Taxonomien

Eine Ontologie dient der Wissensrepräsentation des Seienden [Fur14]. Ontologien spielen in der Informationsverarbeitung eine wichtige Rolle. Auf Basis einer Ontologie können Teile der realen Welt systematisch erfasst werden, damit diese mit maschinellen Prozessen weiterverarbeitet werden können. Grundlage für die Ontologie in MUSE bilden Taxonomien. Diese bieten mehrere Vorteile für die Datenstrukturierung. Neben einem einheitlichen Vokabular und der systematischen Erfassung der Daten bieten Taxonomien auch die Wiederverwendbarkeit an. Taxonomien stellen eine hierarchische Struktur für die Daten einer Domäne bereit. Ontologien stellen zusätzlich zu den Taxonomien und Datentypen deren jeweilige Beziehungsrelationen dar. [Fur14; HKRS08; Stu09] In Abbildung 2.1 ist eine Beispielontologie zu sehen. Die Ontologie stammt aus dem Bereich der Kunst. Blaue Ellipsen sind real existierende Elemente der Welt. Mit den grünen Rechtecken werden diesen Elementen Eigenschaften hinzugefügt. Die Pfeile stellen die Relationen zwischen den Elementen dar.

Hierarchie - Relationen zwischen Elementen Elemente einer Ontologie können in einer Relation zueinander stehen. Wie in Abbildung 2.1 zu sehen gibt es zwischen den einzelnen Elementen der Ontologie verschiedene Relationen. Hierarchische Relationen sind durch grüne Pfeile dargestellt. Diese bedeuten, dass das Material eine Untergruppe einer Hose oder eines Hemdes ist. Daneben sind nicht-hierarchische Relationen mit roten Pfeilen dargestellt (ein Hemd wird am Körper eines Menschen getragen).

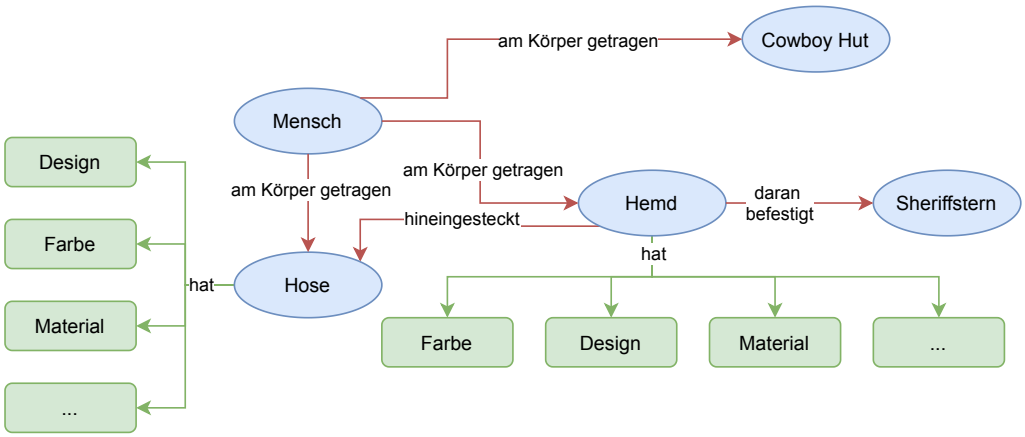


Abbildung 2.1.: Beispielontologie aus MUSE

Bei Taxonomien gibt es diese Unterscheidung nicht. Die Elemente einer Taxonomie stehen in einer rein hierarchischen Relation zueinander.

2.4. Ontologien in MUSE4Anything

Im folgenden Abschnitt geht es um das Datenbankschema von MUSE4Anything und dessen Struktur, da dies die Grundlage der vorliegenden Arbeit bildet. Mit diesem Schema lassen sich alle möglichen Ontologien abbilden. In Abbildung 2.2 ist die generische Datenstruktur von MUSE4Anything dargestellt. Wie zu sehen besteht die Datenstruktur aus vier Hauptbereichen: Namespace (deut.: Namensbereich), Object (deut.: Objekte), Type (deut.: Typen) und Taxonomy (deut.: Taxonomien).

Da in MUSE4Anything die Ontologie erst in der Anwendung definiert wird, wächst die Anzahl der Ontologien im Laufe der Zeit. Deshalb wird jede Ontologie einem Namensbereich zugeordnet, damit die Ordnung bewahrt wird. Hinter dem Taxonomie Element befinden sich, wie der Name sagt, die Taxonomien. Typen sind die Elemente einer Ontologie, welche in einem späteren Schritt bei der Dateneingabe als Grundlage für die Objekte verwendet werden. In den Objekten werden wie beschrieben von den Typen Instanzen erzeugt, welche mit realen Daten gefüllt sind. Daher ist der Teil der Objekte in diesem Fall für die Ontologiedefinition, und damit auch deren Visualisierung, nicht relevant. Wie zu sehen, gibt es für die Objekte, Typen und Taxonomien Versionsnummern, was das vorher genannten Problem der Datenänderung im Laufe der Zeit realisierbar macht. Die Hauptdefinition von Elementen der Ontologie passiert mithilfe der Typen. Diese können verschiedene Eigenschaften besitzen. Zum einen können dies einfache Datentypen wie Zahlen, Textfelder, boolesche Variablen oder Aufzählungen sein. Zum anderen werden auch hier die Taxonomien verwendet. Die letzte Möglichkeit für Eigenschaftstypen sind andere Typen. Dadurch kann es zu komplexen Datenkonstrukten kommen, da es unter anderem zu langen Ketten von

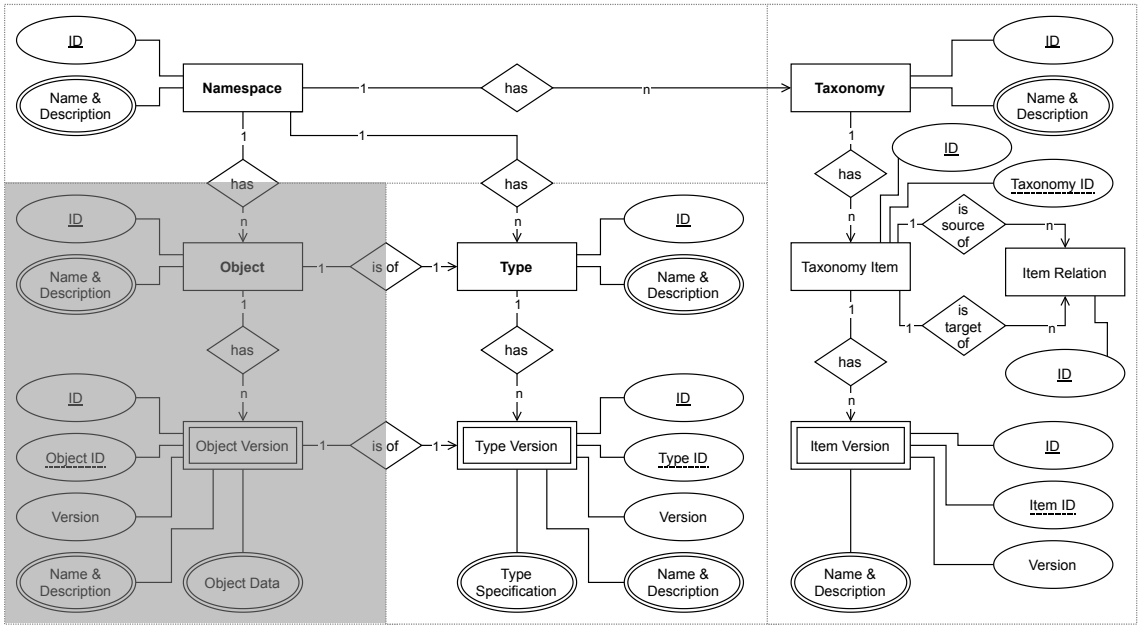


Abbildung 2.2.: Datenstruktur von MUSE4Anything [Büh21]

Typen oder zu einem Kreis führen kann. Ein Kreis bedeutet, dass eine Eigenschaft eines Typs auf einen andern Typen verweist, dieser wiederum hat auch eine Eigenschaft, welche auf den vorherigen selben Typen verweist. Typen können zusätzlich noch als „abstract“ gekennzeichnet werden, was bedeutet, dass diese nur als Eigenschaft eines anderen Typs verwendet werden können und nicht selber ein Objekt erzeugen können. So gekennzeichnete Typen sollen die Dateneingabe übersichtlicher lassen, da diese dann nicht zur Auswahl stehen. Diese Eigenschaft kann bei der Visualisierung der Ontologie berücksichtigt werden.

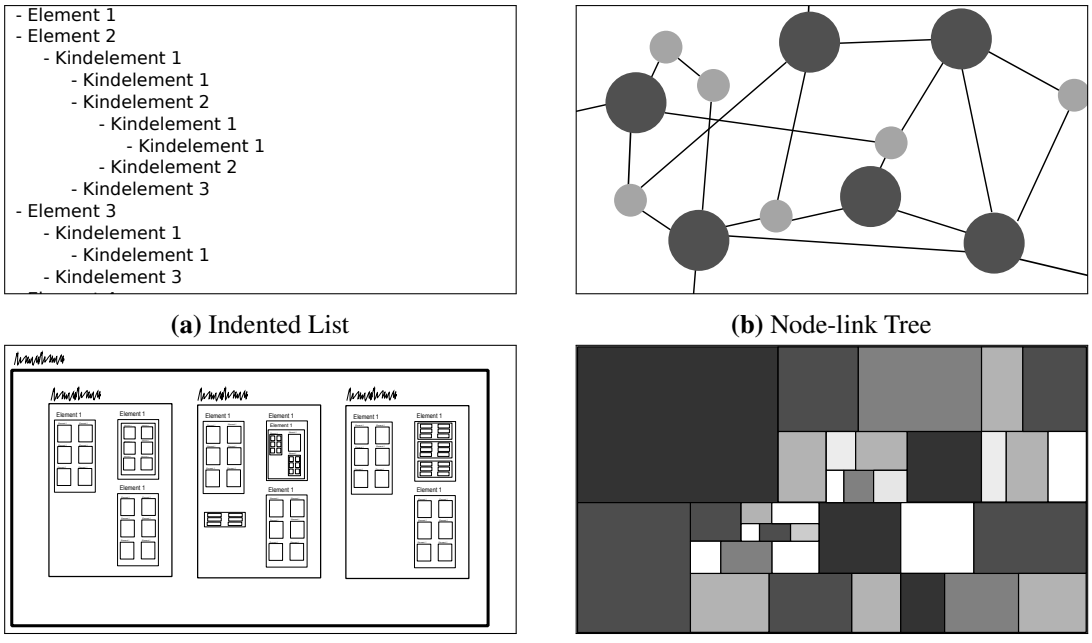
3. Anforderungen an die Visualisierung

In diesem Kapitel werden verschiedene Darstellungsmöglichkeiten und Interaktionsmöglichkeiten für die Visualisierung von Ontologien erläutert. Anschließend werden aus diesen Möglichkeiten Features erarbeitet, welche die Ontologievisualisierung besitzen kann.

3.1. Darstellungsmöglichkeiten

3.1.1. Arten von Visualisierungen

Für die Visualisierung gibt es mehrere Möglichkeiten wie sich die Ontologie darstellen lässt. Es gibt laut Sivakumar und Arivoli die Indented List (deutsch: eingerückte Liste), den Node-link Tree (deutsch: Knoten-Kanten-Baum) , die zoombare Visualisierung und die spacefilling Visualisierung (deutsch: platzfüllende Visualisierung) [RV14; SA11; SBN+10]. Teilweise lassen sich diese Varianten auch kombinieren. Im Folgenden wird auf die einzelnen Varianten genauer eingegangen und auch mögliche Kombinationen werden erläutert.



(a) Indented List

(b) Node-link Tree

(c) Zoombare Visualisierung

(d) Spacefilling Visualisierung

Abbildung 3.1.: Graphenarten

Indented List

Die Indented List kann als 1.5-dimensional verstanden werden, da die einzelnen Elemente hauptsächlich in einer Dimension, der horizontalen Einrückung, angeordnet werden, obwohl das Ergebnis aus zwei Dimensionen besteht [DLSP18]. Die anderen, in dieser Arbeit aufgeführten Möglichkeiten haben alle zwei oder drei Dimensionen. In dieser Visualisierung wird nur ein graphisches Element, die Beschriftung, verwendet [DLSP18]. Dies lässt sich aber durch Symbole und Farben erweitern. Die Darstellung ist nur für einen Ausschnitt der Ontologie geeignet, da durch die Liste viel Platz ungenutzt bleibt, weil das Layout nur in eine Richtung wächst [SBN+10].

Eine Indented List lässt sich einfach und intuitiv implementieren [RV14], da viele Frameworks diese Visualisierung schon implementiert haben.

Die Indented List (Abbildung 3.1a) ist vielen Menschen bekannt, da diese Darstellung an einen Dateimanager erinnert [KTH+06]. Durch die Bekanntheit dieser Abbildung lassen sich Aufgaben schneller lösen als mit einem Node-link Tree [FNS16]. In dieser Visualisierung kann es aber zur Verwirrung kommen, sobald ein Knotenelement mit mehreren Elementen verbunden ist, was nicht der hierarchischen Verbindung entspricht. Diese Art von Verbindungen lässt sich nur durch mehrfaches Aufführen von Elementen in der Liste lösen und kann dann zu Verwirrung führen [FNS16]. Abhängig von der Größe der Ontologie ist in einer Liste die Suche nach einzelnen Informationen leichter, da bei einem Node-link Tree die Suche ab einer gewissen Größe schwer werden kann. Dies liegt an den kürzeren Scanpfaden die das menschliche Auge hinlegen muss, um die gesuchte Information zu finden [FNS16]. Durch die Visualisierung der Ontologie in Listenform, lässt sich die Hierarchie sehr gut darstellen, da jedes Kindelement ein Unterknoten seines Elternelementes ist [SBN+10]. In einem Vergleich von Indented List und Node-link Tree, in welchem Studienteilnehmer die Struktur einer Ontologie ermitteln musste, wurde die Liste dem Baum bevorzugt. Studienteilnehmer haben hierbei die Aufgaben erfolgreicher gelöst, der große Platzverbrauch wurde aber auch hier als ein Nachteil genannt. Eine Kombination dieser beiden Visualisierungen erscheint den Teilnehmern am geeignetsten [FNS13].

Node-link Tree

Eine weitere Möglichkeit der Darstellung ist der Node-link Tree (Abbildung 3.1b). Hier wird jedes Element aus der Ontologie als ein Knoten in einem Graphen dargestellt. Kanten zwischen diesen Knoten stellen jegliche Art von Relation zwischen den einzelnen Knoten dar. Diese Art der Visualisierung kann in zwei Bereiche aufgeteilt werden [DLSP18].

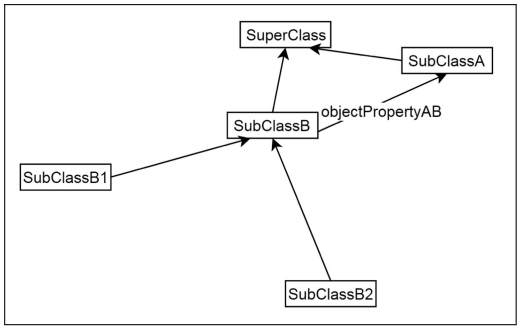
Zum einen sind es die Beschriftungsmöglichkeiten. Hier gibt es zwei Möglichkeiten. Die erste Variante ist, dass nur der Name des Elementes angezeigt wird. Die zweite Variante ist angelehnt an die Unified Modeling Language (UML) und stellt neben dem Namen noch zusätzliche Informationen des Elementes dar.

Zum anderen die Anordnungsmöglichkeit des Diagramms. Es gibt fünf häufig verwendete Layoutvarianten in einem Node-link Tree, welche mit beiden Beschriftungsmöglichkeiten kombinierbar sind. Diese Varianten sind in Abbildung 3.2 zu sehen.

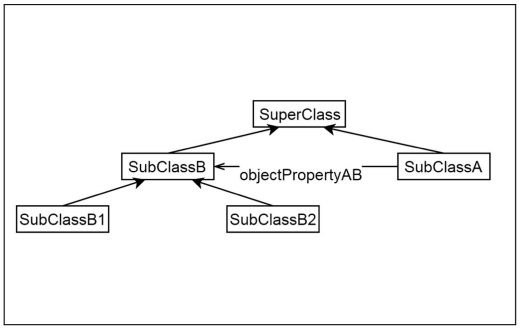
1. Die kraftgesteuerte Anordnung (engl. force-directed layout) (Abbildung 3.2a) ist eine der meist verwendeten Varianten. Jedes Element der Ontologie wird als Knoten dargestellt. Beziehungen werden durch Kanten zwischen den Knoten visualisiert. Das Element mit den meisten Verbindungen wird tendenziell eher mittig angeordnet. Kantenlängen können durch die Stärke der Verbindung beeinflusst werden. Die Knoten werden auf Basis eines Algorithmus möglichst ästhetisch angeordnet. Dieser Algorithmus basiert auf verschiedenen physikalischen Simulationen und benötigt normalerweise kein graphentheoretisches Wissen.
2. Die Baum-Anordnung (engl. tree layout) (Abbildung 3.2b) ist besonders gut geeignet für die Darstellung hierarchischer Relationen. Die Wurzel der Ontologie, das Elternelement, ist ganz oben dargestellt. Die Kinder sind jeweils eine Ebene weiter unter den Eltern angeordnet. Alle Kinder eines Elternelementes sind auf der gleichen Ebene. Nicht hierarchische Links kreuzen normalerweise hierarchische Relationskanten.
3. Die radiale Anordnung (engl. radial layout) (Abbildung 3.2c) ist ebenso für hierarchische Relationen geeignet. Die Spitze der Ontologie ist am Mittelpunkt des Kreises. Kinder sind immer einen Schritt weiter vom Kreis entfernt. Diese Anordnung ist etwas platzsparender als die Baumanordnung.
4. In der Kreisanordnung (engl. circle layout) (Abbildung 3.2d) sind alle Elemente auf einem Kreis angeordnet. Hierbei kann es vermehrt zu Kantenschneidungen kommen. Die Hierarchie der Ontologie ist hier nicht so leicht zu erkennen.
5. Der umgekehrter Radialbaum (engl. inverted radial tree layout) (Abbildung 3.2e) ist eine Kombination der spacefilling Visualisierung und des Node-link Tree. Elemente aus der Ontologie werden als ein Kreis dargestellt. Kindelemente davon werden als konzentrische Kreise innerhalb dargestellt. Elternelemente werden weiter außen dargestellt. Durch Kanten werden nicht-hierarchische Relationen visualisiert.

Bei der Wahl der Layoutvariante bietet die kraftgesteuerte Anordnung die passendste Anordnung für den Benutzer. Die Algorithmen hinter dieser Anordnung sind die flexibelsten Algorithmen zur Layoutberechnung und für unterschiedlichste Datengrundlagen zu benutzen. Für die Berechnung werden ausschließlich Informationen aus der Struktur des Graphen benötigt. Dies bedeutet, es sind nur die Knoten und Kanten notwendig, sodass der Graph sich an Änderungen in der Anordnung des Benutzers immer wieder anpassen kann. Das Ergebnis dieser Anordnung ist in der Regel ansprechend und neigt dazu kreuzungsfreie Graphen zu erzeugen. Früher waren diese Algorithmen nur für ein paar Hundert Knoten geeignet. Im Laufe der Zeit wurde an diesen Algorithmen weiter gearbeitet, und wird immer noch, sodass heute auch größere Graphlayouts berechnet werden können. [Kob12]

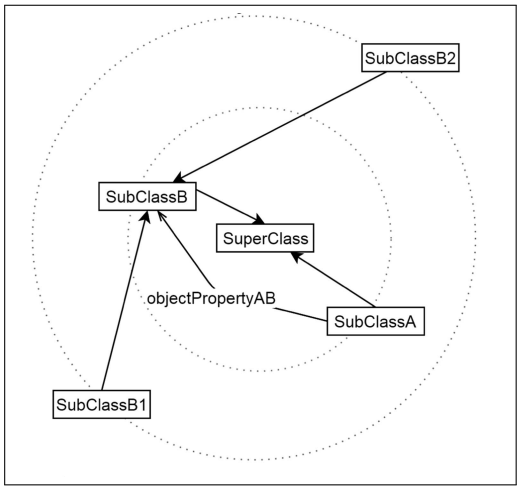
Die Variante des Node-link Tree hilft Nicht-Experten bei einem schnellen Einstieg [AWL] und wird in den meisten Anwendungen verwendet [DLSP18; RV14]. Bei zu vielen Knoten kann diese Art unübersichtlich werden [RV14]. Dies lässt sich unter anderem durch eine Kombination mit der zoombaren Visualisierung reduzieren. Der Vorteil eines Node-link Tree im Gegensatz zur Indented List ist, dass sich hier Mehrfachvererbungen und semantische Beziehungen darstellen lassen [SBN+10]. Trotzdem liegt der Hauptfokus bei den meisten Anwendungen auf der Hierarchievisualisierung [DLSP18]. Neben der Hierarchie lässt sich mithilfe des Node-link Tree auch sehr gut eine Übersichtsdarstellung realisieren [RV14]. Eine Übersichtsdarstellung stellt auf einen Blick die gesamte Ontologie dar.



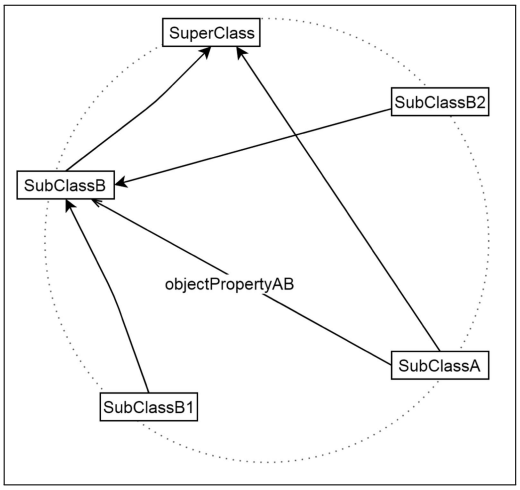
(a) Kraftgesteuerte Anordnung



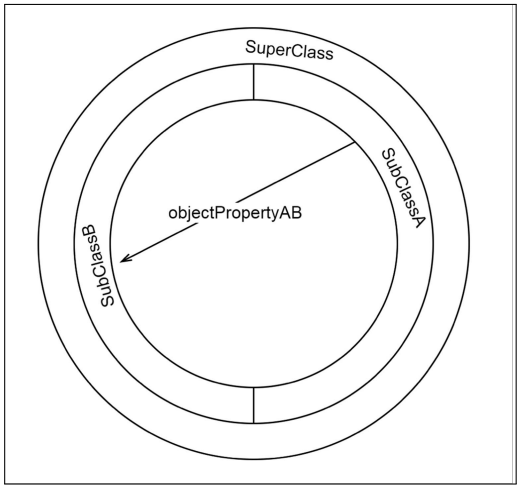
(b) Baum Anordnung



(c) Radiale Anordnung



(d) Kreisanordnung



(e) Umgekehrter Radialbaum

Abbildung 3.2.: Layout des Graphen [DLSP18]

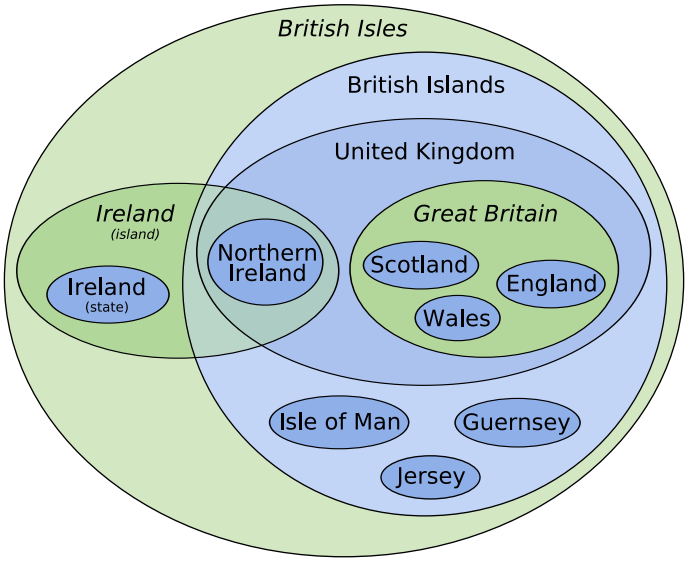


Abbildung 3.3.: In diesem Euler-Diagramm sind die Zusammenhänge der Britischen Inseln und deren Gebiete dargestellt. Jede Insel oder jedes Gebiet wird durch eine Ellipse dargestellt. Zusammengehörigkeiten werden mit einer gemeinsamen Ellipse dargestellt. Elemente können auch zu mehreren Gebieten gehören, wie „Northern Ireland“ [TWC11]

Zoombare Visualisierung

Bei der zoombaren Visualisierung (Abbildung 3.1c) sind Kindelemente in den Eltern dargestellt [RV14; SA11]. Diese Darstellungsmöglichkeit lässt sich mit dem semantischen Zoomen kombinieren (Abschnitt 3.2.2).

Eine Variante der zoombaren Visualisierung sind die Euler-Diagramme [DLSP18]. In Abbildung 3.3 ist ein Beispieldiagramm dargestellt. Zu sehen ist, wie die einzelnen britischen Inseln zusammengehören. Im Gegensatz zu den anderen Darstellungsmöglichkeiten können hier auch andere Beziehungsarten dargestellt werden, zum Beispiel Disjunktheit. Euler-Diagramme können mit Node-link Trees kombiniert werden.

Diese Visualisierung führt zu einer klaren Übersichtsdarstellung [SBN+10], da Kindelemente in der Anfangsdarstellung nicht sichtbar sind und somit die Anzahl der zu visualisierenden Elemente limitiert ist. Kindelemente werden erst durch Zoomen sichtbar. Hier kann der Kontext verloren gehen, da Elternelemente ab einem gewissen Zoomlevel nicht mehr im aktuell dargestellten Bereich sind [SBN+10].

Spacefilling Visualisierung

Die spacefilling Visualisierung (Abbildung 3.1d) ist darauf ausgelegt, den gesamten verfügbaren Platz zu verwenden. Die bekannteste Variante sind Treemaps. Dabei wird jedes Element der Ontologie als Rechteck repräsentiert.

Die spacefilling Visualisierung wird unter anderem für die Darstellung der Fesplattenauslastung benutzt. Hier werden alle Ordner als ein Rechteck dargestellt und Unterordner und darin enthaltene Dateien in diesem Rechteck. Die Größe des Rechteckes wird durch die Größe des belegten Festplattenplatzes bestimmt. Hier müssen keine nicht hierarchischen Beziehungen dargestellt werden, daher eignet sich diese Visualisierung hierfür.

In Abschnitt 2.4 wurde erläutert, dass die Ontologien von MUSE4Anything sehr komplex werden können. Diese Darstellung ist nicht für komplexe Ontologien geeignet, da hier nicht hierarchische Relationen nicht dargestellt werden können [KHL+07; SBN+10].

Dimensionalität der Visualisierung

Wie schon erläutert, lässt sich die Visualisierung mit verschiedenen Dimensionen darstellen. Dies fängt bei einer 1.5-dimensionalen Darstellung an. Hierunter fällt die Indented List. Zu den Visualisierung mit zwei Dimensionen zählen alle Darstellungen, welche auf einer Ebene dargestellt werden können und Gebrauch von mehreren Richtungen machen. Bei Visualisierungen mit 3 Dimensionen kommt noch die Tiefe der Darstellung dazu. Ab der dritten Dimension macht von den genannten Darstellungen nur der Node-link Tree Sinn. Weitere Dimensionen sind für das menschliche Auge schwer zu verarbeiten.

Laut Silva und Freitas ist eine 2.5D Visualisierung effektiver als eine 2D Visualisierung [SF11]. Eine mögliche 2.5D Visualisierung basiert auf einem 2D Node-link Tree, welcher nur die hierarchischen Beziehungen darstellt. In der weiteren halben Dimension werden nicht-hierarchische Relationen über den existierenden Baum gespannt.

Der Vorteil eines 3D Layouts ist die weitere Dimension. Durch diese weitere Dimension entsteht mehr Raum, dadurch gibt es mehr Platz und daher können mehr Informationen auf gleichem Platz dargestellt werden [FNS13; HMM00]. Das Hauptproblem bei der 3D Visualisierung ist, dass Standardmonitore eine 2D Darstellung haben und somit die 3D Visualisierung wieder auf eine 2D Visualisierung angepasst werden muss, was wieder zu einem Verlust der weiteren Dimension führt [DLSP18].

3.1.2. Darstellung der Elemente

Neben der grundsätzlichen Art der Visualisierung des Graphen gibt es bei jeder Variante noch zusätzliche Faktoren bei der Gestaltung der Elemente zu beachten. Durch spezielle Gestaltungselemente lassen sich die Elemente einer Ontologie besser unterscheiden.

Grafische Elemente

Das ist zum einen die Darstellung der Elemente im Graphen. Hier führt das Verwenden unterschiedlicher Farben und Formen für die Elemente zu einer leichteren Unterscheidbarkeit [KVWW07; LNB14; Moo09]. In der Definition von VOWL 2 haben Lohmann et al. [LNHE14] direkt Farben und Formen für Komponente definiert. Durch unterschiedliche Darstellungsformen und Farben lassen sich die Elemente leichter in der Hierarchie der Ontologie einordnen [CSM09]. Neben den genannten Merkmalen kann unter anderem noch zusätzlich auf die Größe und Textur der Elemente

geachtet werden. Zusätzlich lassen sich die Elemente mit Symbolen verknüpfen, was im Speziellen für farbenblinde Benutzer die Unterscheidung einzelner Elemente erleichtert, da diese sonst nur die Form der Elemente erkennen können [BJKK10]. Bei der Verwendung von Symbolen sollte darauf geachtet werden, dass ein Symbol genau eine Bedeutung und jede Bedeutung genau ein Symbol hat [Moo09].

Beschriftung

Neben der Darstellung der Elemente ist die Beschriftung mindestens genauso wichtig. Die Beschriftung hilft dem Benutzer das Element genauer zu erkennen [LNHE14; LNHE16]. Hierbei sollte aber darauf geachtet werden, dass die Beschriftungen sich nicht überlagern [KTH+06], da diese sonst schwerer zu lesen sind. Falls sich Beschriftungen überschneiden, sollten diese nicht einfach abgeschnitten werden, da dies ebenso zur Verwirrung führen kann, falls diese den gleichen Anfang haben [KW10]. Dies wird anhand der folgenden zwei Beschriftungen kurz erläutert. Die erste Beschriftung lautet „Node-link Tree Layout -> Force-directed Layout“. Die Zweite lautet „Node-link Tree Layout -> Tree Layout“. Wenn aufgrund der Größe der Elemente die Beschriftung nach 10 Zeichen abgeschnitten wird, wird bei beiden Elementen „Node-link T“ dargestellt und somit sind diese Elemente auf den ersten Blick nicht unterscheidbar. Dieses Problem kann in interaktiven Diagrammen durch das Darstellen von Tooltipps gelöst werden.

Bei den Überlegungen zu OWLeasyViz, einer benutzerfreundlichen Anwendung welche auch für Nicht-Experten gut bedienbar sein soll, wurden die technischen Begriffe der ontologischen Relationen durch menschlichere ersetzt, was in der durchgeführten Studie positiv bei den Benutzern angekommen ist. Catenazzi, Sommaruga und Mazza haben bei ihrer benutzerfreundlichen Ontologievisualisierung zum Beispiel „ObjectProperty“ durch „relationship“ und „DatatypeProperty“ durch „property“ ersetzt. [CSM09, S. 286] Die Beschriftung sollte trotzdem das Zielpublikum im Auge haben und dementsprechend angepasst sein.

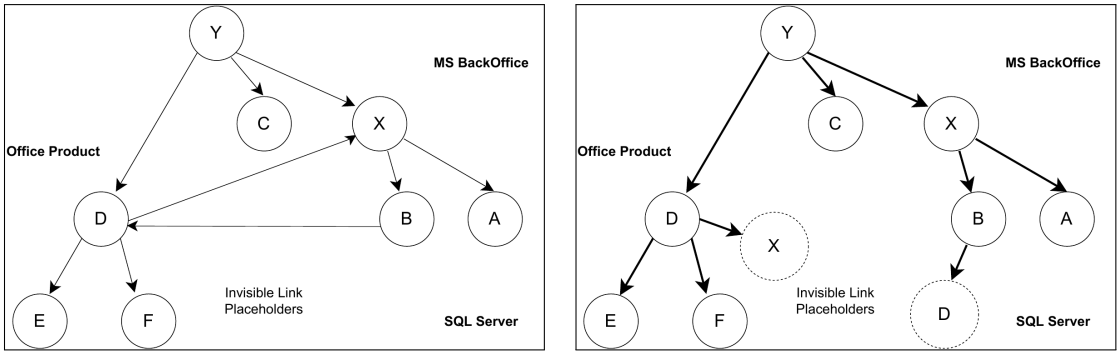
Positionierung

Lohmann et al. schreiben unter anderem, dass stark verbundene Komponenten, also die Komponenten mit den meisten gemeinsamen Kanten, nah beieinander positioniert werden sollen, da diese viele gemeinsame Kanten besitzen. Werden diese mit großem Abstand verteilt wird der Graph durch die vielen langen Kanten zwischen den Komponenten unübersichtlicher [LNHE16]. Dies wird unter anderem bei der Layoutgestaltung der kraftgesteuerten Anordnung (Punkt 1) berücksichtigt.

Kantenschneidung vermeiden

Ab einer gewissen Graphengröße kann es zu vielen Überschneidungen bei den Kanten kommen. Zur Vermeidung von Überschneidungen werden nun zwei Ansätze vorgestellt.

Bei dem ersten Ansatz geht es um die mehrfache Darstellung von Elementen. Hierbei wird ein Teilgraph, welcher mehrere Eltern, hat mehrfach im Graphen angeordnet. [ERG02] Dies führt ab einer gewissen Anzahl an mehrfach dargestellten Elementen zur Überladung des Graphen. Ebenso kann es zu Verwirrung führen, da Elemente mehrmals dargestellt werden. Sobald eine Rekursion im Graph vorhanden ist, führt das zu einer unendlich großen Menge an Elementen im Graph.



(a) Keine Verwendung von Unsichtbare Links **(b)** Unsichtbare Links [HHDK00]

Abbildung 3.4.: Unsichtbare Links

Die zweite Variante nennt sich unsichtbaren Links. Unsichtbare Links sind alle Verbindungen im Graphen, welche nicht in eine direkte azyklische Baumhierarchie passen. Jedes Element in diesem Graph kann einen Vorgänger haben und viele unsichtbare Links. In Abbildung 3.4b sind zwei unsichtbare Links dargestellt. Anstatt eine Linie von D nach X und von B nach D zu zeichnen, werden Platzhalter hinzugefügt, welche dann zu X und D zeigen. Wenn ein Benutzer ein Element mit einem oder mehreren unsichtbaren Links auswählt, werden die Zielelemente im Graphen hervorgehoben. [HHDK00] In Abbildung 3.4a ist dieselbe Situation ohne die Verwendung von unsichtbaren Links dargestellt. Hierbei sind die azyklischen Kanten zwischen D und X und die Kante zwischen B und D zu erkennen.

Hierarchische und nicht hierarchische Relationen

Zur Unterscheidung von hierarchischen und nicht hierarchischen Relationen gibt es zwei Darstellungsarten. Hierarchien lassen sich auf der einen Seite als Kante darstellen. Auf der anderen Seite kann dies durch ineinandergeschachtelte Elementen realisiert werden. Hierbei lassen sich nicht hierarchische Relationen mit Kanten darstellen, dadurch können diese unterschieden werden. Bei der Darstellung von hierarchischen Relationen mit Kanten werden auch nicht hierarchische Relationen so dargestellt.

In der Studie von Kriglstein und Wallner [KW10] kommt heraus, dass Knoocks (Abschnitt 4.1.2) und Jambalaya (Abschnitt 4.1.8) im Gegensatz zu TGViz (Abschnitt 4.1.8) zwischen hierarchischen und nicht-hierarchischen Relationen unterscheiden. Bei den ersteren beiden sind die Kindelemente, wie vorher erläutert, in den Eltern geschachtelt und nicht hierarchische Relationen durch Kanten visualisiert. Bei TGViz wird alles durch Kanten dargestellt, was die Unterscheidung erschwert.

3.1.3. Übersicht der genannten Darstellungsmöglichkeiten

In Abschnitt 3.1 wurden mögliche Arten der Visualisierung, Layoutvarianten und Elementgestaltungen vorgestellt. Diese werden nun kurz zusammengefasst.

Als erstes geht es um die Arten der Visualisierung.

Dar. 1 Indented List: Die Elemente der Ontologie werden als Liste aufgelistet.

Dar. 2 Node-link Tree: Elemente der Ontologie werden in einem Graphen angeordnet und Relationen mit Verbindungskanten dargestellt.

Dar. 3 Zoombare Visualisierung: Kindelemente werden in den Eltern angeordnet, was ab einer gewissen Tiefe zu hoher Komplexität führt.

Dar. 4 Spacefilling Visualisierung: Die Elemente der Ontologie werden als Rechtecke angeordnet. Kinder werden im Elternrechteck angeordnet. Nicht hierarchische Beziehungen sind hier schwer darstellbar.

Im folgenden geht es um die vorgestellten Layoutvarianten für den Node-link Tree.

Dar. 5 Kraftgesteuerte Anordnung: Elemente werden mithilfe eines Algorithmus angeordnet.

Dar. 6 Baumanordnung: Die Elemente werden hierarchisch von oben nach unten oder von links nach rechts angeordnet.

Dar. 7 Radialanordnung: Diese Anordnung ist ähnlich zur Baumanordnung, nur dass hier das Wurzelelement der Ontologie in der Mitte positioniert wird.

Dar. 8 Kreisanordnung: Alle Elemente werden auf einem Kreis angeordnet.

Dar. 9 Umgekehrter Radialbaum: Dies ist eine Kombination der spacefilling Visualisierung mit einem Node-link Tree.

Zuletzt werden die Darstellungsmöglichkeiten der Elemente zusammengefasst.

Dar. 10 Form: Verschiedene Formen sorgen für eine bessere Unterscheidung der Elemente im Graph.

Dar. 11 Farbe: Verschiedene Farben sorgen für eine bessere Unterscheidung der Elemente im Graph.

Dar. 12 Symbole: Verschiedene Symbole sorgen für eine bessere Unterscheidung der Elemente im Graph.

Dar. 13 Beschriftung: Beschriftungen sind ein wichtiger Bestandteil von Graphen.

Dar. 14 Positionierung: Bei der Positionierung ist es hilfreich, wenn stark verbundene Komponenten eng beieinander positioniert sind.

Dar. 15 Kantenschneidung: Zur Vermeidung von Kantenschneidungen wurden zwei Verfahren erläutert. Zum einen die mehrfache Komponentendarstellung und zum anderen die unsichtbaren Links zu den eigentlichen Komponenten.

3.1.4. Gruppierung der Ontologie

Im Hinblick auf eine übersichtlichere Darstellung geht es im folgenden Abschnitt um die Gruppierung von Elementen der Ontologie.

Da es sich hier bei den zu visualisierenden Informationen nur um Datenformate handelt und nicht um konkrete Daten ist das automatische Gruppieren anhand von Algorithmen nicht leicht realisierbar. Gruppierungsalgorithmen basieren unter anderem auf numerischen Werten [BR; GJZZ13] oder auf graphischer Analyse [SS08; TK18]. Da in diesem Fall aber keine Daten vorhanden sind, sind solche Algorithmen nicht anwendbar.

Ein Element in der Ontologie kann sehr viele Kindelemente besitzen, aber nicht immer müssen alle Kindelemente angezeigt werden. Der Benutzer kann Kindelemente gezielt ausblenden oder zusammenfassen, und somit die Komplexität des Graphen reduzieren. Im Beispiel der Ontologie von MUSE4Music hat das Opus (Kunstwerk) fünfzehn Eigenschaften, welche teilweise wieder auf weitere Datenobjekte verweisen. Diese Eigenschaften lassen sich zwar nicht automatisch gruppieren, sodass die Darstellung übersichtlicher wird, aber es gibt die Möglichkeit nur einen Teil der Daten darzustellen. Um den Benutzer vor einer überladenen Visualisierung im ersten Moment zu bewahren, ist das Ziel auf den ersten Blick nur einen Teil der Eigenschaften anzuzeigen. Der Nutzer kann mit einer geeigneten Interaktionsmöglichkeit nachträglich wählen wie viele Eigenschaften dargestellt werden sollen. Das Problem ist hierbei die Auswahl der anfangs dargestellten Elemente. Zwei einfache Möglichkeiten sind zum einen die Reihenfolge der Eigenschaften, in welcher sie aus der Datenbank geliefert werden, oder die alphabetische Reihenfolge. Eine aufwändigere Variante ist die Auswahl der Elemente mithilfe einer Heuristik. Die Heuristik beschreibt ein Verfahren zur Problemlösung mit begrenztem Wissen und geringer Zeit. Die Lösungen davon sind nicht perfekt, aber ausreichend. Das bekannteste Verfahren ist „Versuch und Irrtum“[Pil12].

Heuristische Verfahren beruhen auf Erfahrungen aus der Vergangenheit, was in diesem Fall aber nicht möglich ist, da es keine vergangenen Informationen der jeweiligen Ontologie gibt. Da mit diesen Ontologien auch nur das Datenformat definiert wird, gibt es auch keine inhaltlichen Informationen. Ein mögliches Ordnungsverfahren für die Heuristik sind die Attributsnamen der einzelnen Typen und Taxonomien. Hier kann auf Basis vieler Ontologien die Häufigkeit von gleichnamigen Attributen ermittelt werden und dadurch die Heuristik erstellt werden.

Bei der Erstellung und Bearbeitung von Taxonomien in MUSE4Anything kann der Benutzer die Reihenfolge der Eigenschaften eines Elementes selber definieren, da alphabetisch oder die Reihenfolge der Erstellung häufig nicht die passendste ist [Büh21, S. 15]. Daher sollte bei der Visualisierung der Ontologie auch auf diese Reihenfolge zurückgegriffen werden.

3.2. Interaktionsmöglichkeiten

Neben der Darstellung der Ontologie spielt die Interaktion eine genauso wichtige Rolle. Denn selbst die beste Darstellung wird durch geeignete Interaktionsmöglichkeiten noch besser. Die meisten Anwendungen haben dabei nur einen kleinen Teil aller möglichen Interaktionsmöglichkeiten implementiert [DLSP18]. Im Folgenden werden verschiedene Interaktionsmöglichkeiten erläutert.

3.2.1. Suchen und Filtern

Die Such- und Filterfunktionalität gehört zu den Basisfunktionalitäten einer Anwendung. Diese wurde in mehreren Studien von den Teilnehmern als positive Funktionalität gewertet oder, falls sie nicht vorhanden war, wurde es als eine fehlende Funktionalität angegeben [ALK+17; CSM09; Kri09; LNHE16]. Beim Filtern ist die Möglichkeit des Versteckens einzelner Elemente im Graphen von großem Vorteil, da dadurch der Fokus individuell auf einzelne Elemente gelegt werden kann und die Zusammenhänge dieser Elemente besser verstanden werden können [DLSP18].

3.2.2. Verschieben & Zoomen

Verschieben von einzelnen Elementen und des gesamten Graphen ist eine Funktionalität, die von fast allen Implementierungen implementiert und die für eine gute Interaktion unumgänglich ist [DLSP18; KW10]. Durch diese Funktion kann der Benutzer entweder die Anordnung der Elemente auf sein Bedürfnis anpassen oder den angezeigten Bereich der Darstellung auswählen.

Eine weitere Interaktionsmöglichkeit ist das Zoomen. Hier gibt es zwei unterschiedliche Möglichkeiten. Zum einen das semantische Zoomen, wodurch der Informationsinhalt verändert werden kann [HMM00]. So werden beim semantischen Heranzoomen mehr Details angezeigt, beim Herauszoomen weniger. Daneben gibt es noch das geometrische Zoomen, welches das häufiger verwendete Verfahren ist. Hierbei wird die Größe der dargestellten Elemente verändert. Diese Art des Zoomens wird meistens mit der Verschiebefunktion kombiniert. Geometrisches Zoomen ist genauso wie das Verschieben von Elementen eine Grundfunktionalität von Visualisierungen. Bei der Studie von Knoocks [KW10] wurde als eine Stärke erwähnt, dass neben dem Zoomen mittels Mausrad, auch Zoomen mit einem Doppelklick, was vor allem für Touchscreen wichtig ist, möglich ist. Durch die verschiedenen Varianten der Zoomsteuerung sind die Möglichkeiten verschiedener Geräte optimal benutzbar.

3.2.3. Dynamische Detailinformationen

Um mehr Informationen über einzelne Elemente der Ontologie zu bekommen, hilft eine dynamische Detailinformationsdarstellung. Hierfür gibt es verschiedene Ansätze. Dies kann durch einen Detailansichtsbereich im Fenster gelöst werden oder durch einen Tooltipp [DLSP18]. Wills [Wil99] erweitert dieses Konzept in seiner Implementierung um 4 Status für Elemente: Gelöscht/versteckt, normal, markiert oder fokussiert. Im fokussierten Status werden soviel Informationen wie vorhanden über das Element dargestellt. Bei Knoocks sind die Detailinformationen in einem auf- und zuklappbaren Bereich der einzelnen Elemente hinterlegt [KW10]. Die Detailansicht kann teilweise auch durch eine Fisheye-Linse realisiert werden. Hierzu mehr in Abschnitt 3.2.5.

3.2.4. Fokus & Kontext

Bei der Fokus & Kontext Methode [SA11] wird das fokussierte, ausgewählte Element der Ontologie immer in der Mitte der Visualisierung positioniert, also eine dynamische Ausschnittswahl. Durch dieses Darstellungsverfahren ist eine effektive Übersichtsdarstellung, der Fokus auf einzelne Elemente und schnelles Suchen, gegeben. Dies ist aber auch zugleich ein Nachteil, da durch das Wählen eines anderen Elementes als neuer Fokus sich die gesamte Visualisierung verschiebt [SBN+10].

3.2.5. Linsen

Die letzte hier aufgelistete Möglichkeit der Interaktion ist die Verwendung verschiedener Linsen. Hiermit lassen sich mehr Details im ausgewählten Bereich darstellen oder der Fokus auf bestimmt Eigenschaften lenken [DCL+17]. Tominski präsentiert drei Arten von Linsen für Graphenuntersuchung [TAHS06]. In Abbildung 3.5 sind die von ihm vorgeschlagenen Varianten einer Linse zu sehen. In Abbildung 3.5a ist keine Linse verwendet, was die Basisdarstellung der Situation darstellt. Es gibt zum einen die „Lokale Kanten Linse“ Abbildung 3.5b, welche im Linsenbereich nur die direkten Kanten des ausgewählten Knoten anzeigt, alle vorbeigehenden Kanten werden ausgeblendet. Diese hat den Vorteil, dass in stark verzweigten Bereichen nur die gewünschten Kanten sichtbar sind und somit die Relationen einfacher erkannt werden können. Zum anderen gibt es die „Bring Neighbours“ Linse Abbildung 3.5c, welche die Nachbarknoten in die Nähe des Knoten bringt. Mit dieser Linse lassen sich alle Relationen erkennen, welche von allen Nachbarn des ausgewählten Knoten ausgehen. Diese beiden Linsen lassen sich auch zusammen kombinieren Abbildung 3.5d, was den Vorteil beider Linsen zusammenbringt. Neben dieser 2D Linse gibt es noch die Fisheye-Linse, welche mithilfe von 3D mehr Eigenschaften darstellt [DLSP18; TAHS06].

3.2.6. Übersicht der genannten Interaktionsmöglichkeiten

In Abschnitt 3.2 wurden mögliche Interaktionsmöglichkeiten vorgestellt. Diese werden nun kurz zusammengefasst.

Inter. 1 Eine Möglichkeit für eine passende Visualisierung ist das Suchen nach bestimmten Elementen.

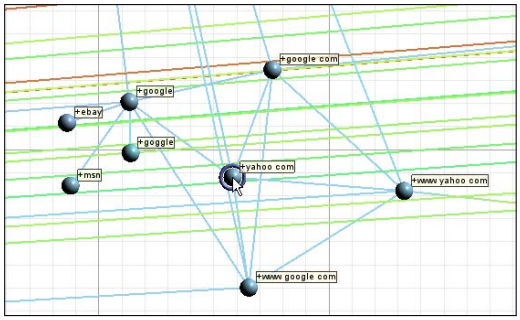
Inter. 2 Ein weiterer Punkt für den Benutzer ist die Möglichkeit des Filterns nach bestimmten Elementen.

Inter. 3 Mithilfe des Zoomens und Verkleinerns des Graphen kann der Benutzer sich im Graph orientieren.

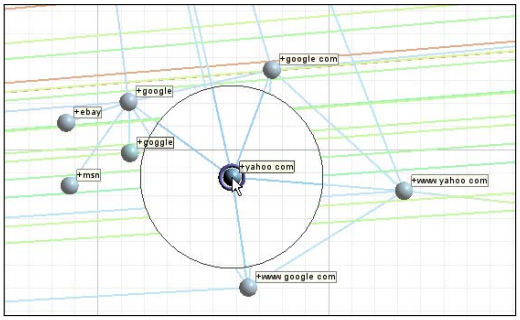
Inter. 4 Zusätzlich zum Zoomen ist das Verschieben des Graphen und einzelner Elemente für die Interaktion im Graphen vorteilhaft.

Inter. 5 Für weitere Informationen können Detailinformationen angezeigt werden.

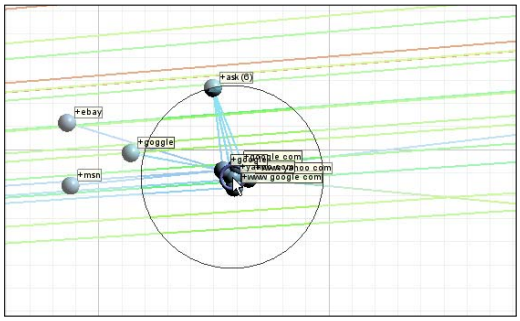
Inter. 6 Bei der Fokus & Kontext Methode wird das ausgewählte Element im Graph in der Mitte positioniert.



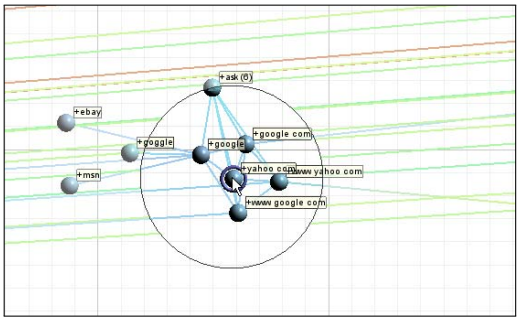
(a) Keine Linse, Standarddarstellung



(b) Lokale Kanten Linse



(c) Bring Neighbours Linse



(d) Komposition von beider Linsen

Abbildung 3.5.: Verschiedene Linsentechniken nach Tominski [TAHS06] im Vergleich. In 3.5a ist keine Linse in Benutzung. In 3.5b ist die Lokal Kanten Linse dargestellt, welche nur die Kanten des ausgewählten Knoten darstellt. Bei der Linse in 3.5c werden die Nachbarn des ausgewählten Knoten näher an den Knoten gebracht. Linse 3.5d kombiniert die Funktionen von 3.5b und 3.5c

Inter. 7 Mithilfe verschiedener Linsen können dem Benutzer unterschiedliche Informationen im ausgewählten Bereich angezeigt werden.

3.3. Features für die Visualisierung

In den vorangegangen Abschnitten sind mögliche Darstellungs- und Interaktionsvarianten vorgestellt worden. Im folgenden Abschnitt soll eine erste Sammlung an Features für die Visualisierung von Ontologien entstehen. Zuerst wird auf die Darstellung und das Layout eingegangen, anschließend geht es um die möglichen Interaktionen, die mit der Visualisierung möglich sind.

3.3.1. Darstellung des Layouts

Aus Untersuchungen mehrerer Studien [Ala03; FNS13; RV14; SF11] geht hervor, dass Aufgaben mit einer Kombination eines Node-link Tree und einer Indented List von den Teilnehmern am erfolgreichsten gelöst wurden. Der Vorteil einer Indented List liegt primär auf der Bekanntheit durch ähnliche Anwendungen, wie den Dateiexplorer [FNS16; KTH+06; TAS09]. Daneben ist diese einfach und intuitiv bedienbar [RV14]. Ebenso ist bei dieser Darstellungsvariante die Informationssuche besser [FNS13; FNS16; SBN+10]. Der Nachteil hier ist zwar der größere Platzverbrauch [FNS13; SBN+10], aber deswegen gibt es noch zusätzlich den Node-link Tree. So ein Baum hilft Nicht-Experten bei einem schnellen Einstieg [Kri09; LNHE14; Moo09].

Feature 1 Die Ontologie ist, wie in Darstellungsmöglichkeit 1 erläutert, durch eine Indented List dargestellt.

Feature 2 Die Ontologie ist, wie in Darstellungsmöglichkeit 2 erläutert, durch einen Node-link Tree dargestellt.

Zur Anordnung der Komponenten gibt es auch ein grundsätzliches Ergebnis aus den Studien. Stark verbundene Komponenten sollen enger beieinander angeordnet werden, sodass lange überschneidende Kanten verhindert werden können [LNHE16]. Neben der Entfernung der Komponenten lässt sich auch anhand der Dicke eines Links die Anzahl der Verbindungen zwischen Komponenten visualisieren [NM08]. Wie Lanum in seinem Buch schreibt, erzeugt die kraftgesteuerte Anordnung immer ein angemessenes Ergebnis. Bei den anderen Layouts kann es zu unübersichtlichen Graphen kommen [Lan17]. Ebenso steigt das Zufriedenheitsgefühl der Benutzer bei dieser Layoutvariante [AWL].

Feature 3 Die Elemente im Node-link Tree sind anhand des kraftgesteuerten Layouts angeordnet, wie schon in Darstellungsmöglichkeit 5 erklärt.

Feature 4 Darstellungsmöglichkeit 14 erläutert, dass stark verbundene Komponenten näher beieinander positioniert werden sollen.

In einer anderen Studie wurden die Benutzer nach den Bedürfnissen für die Visualisierung von Ontologien gefragt. Ein Teil der Teilnehmer wünscht sich eine Übersichtsdarstellung der Ontologie zum Einordnen der aktuell detaillierteren Ansicht [Kri09; KW10]. Ebenso lässt sich mithilfe einer Übersichtsdarstellung die gesamte Hierarchie der Ontologie darstellen [TAS09]. Da diese Darstellung hilfreich sein kann, wird dies ebenso in MUSE4Anything eingebaut.

Feature 5 Zusätzlich zur Node-link Tree Visualisierung gibt es eine Übersichtsdarstellung der Ontologie.

Die in den letzten Absätzen genannten Visualisierungen sind am besten in Kombination zu verwenden [FNS13]. Bei der Studie über OntoGraf wurde die Verknüpfung der Listenansicht und des Node-link Tree als ein Vorteil genannt [RV14]. Das gleiche Verhalten wurde auch bei TGVizTab implementiert und ebenso als hilfreich bewertet [Ala03]. Durch die Koppelung aller Visualisierungen, der Indented List, dem Node-link Tree und der Übersichtsdarstellung kann der Benutzer in den jeweiligen Momenten immer die passendste Darstellung in Betracht ziehen und somit den für sich schnellsten Weg zur Erledigung der Aufgabe finden.

Feature 6 Die einzelnen Visualisierungen sind miteinander verknüpft.

Die beschriebene Hauptdarstellung, der Node-link Tree, lässt sich noch mit einer weiteren Visualierungsmethode kombinieren, der zoombaren Visualisierung (Abschnitt 3.1.1). In der Studie von Katifori et al. [KTH+06] schneidet aufgrund dieses Features Jambalaya besser ab als TGVizTab. Durch die Anordnung der Kindelemente in den Eltern wird die gesamte Visualisierung übersichtlicher, was aber auch zu einem Nachteil werden kann, da ab einer gewissen Tiefe des Zooms der Kontext verloren gehen kann [SBN+10]. Dies kann aber durch die Verknüpfung mit der Übersichtsdarstellung verhindert werden. Vor allem für Neulinge im Bereich der Ontologievisualisierung ist die Anordnung der Kindelemente in den Elternelementen leichter verständlich [KW10; Moo09].

Feature 7 Im Node-link Tree sind, angelehnt an Darstellungsmöglichkeit 3, die Kindelemente innerhalb der Eltern dargestellt.

Neben der grundsätzlichen Art der Visualisierung lassen sich die einzelnen Elemente in der jeweiligen Methode noch zusätzlich gestalten, was zu einer besseren Unterscheidbarkeit führt [LNB14]. Hierbei können unter anderem verschiedene Farben, Formen und Symbole verwendet werden [SF11]. Farben helfen den Benutzern auf den ersten Blick ähnliche Elemente zu unterscheiden [Moo09]. Mithilfe von Formen der Elemente, lassen sich zum einen unterschiedliche Objektarten darstellen zum anderen lässt sich die Position in der Hierarchie darstellen [CSM09]. Die letzte hier genannte Gestaltungsvariante sind Symbole. Diese helfen vor allem farbenblinden Benutzern [KVWW07]. Hierbei ist nur zu beachten, dass eine 1-zu-1 Beziehung zwischen den Symbolen und deren Bedeutung besteht, da es sonst zu einer der folgenden Missverständnisse führen kann [Moo09].

- Symbol-Redundanz tritt auf, wenn mehrere Symbole das gleiche Element repräsentieren.
- Symbol-Überladung tritt auf, wenn mehrere Elemente mit dem gleichen Symbol dargestellt werden. Dies ist der schlimmste Fall, da es zu Fehlinterpretationen führen kann.
- Symbol-Überschuss tritt auf, wenn Symbole keinem Element zugeordnet sind. Dies führt zu einer erhöhten Komplexität, und beeinflusst somit die Verständlichkeit der Visualisierung
- Symbol-Mangel tritt auf, wenn ein Element kein Symbol besitzt.

Feature 8 Auf Basis der Darstellungsmöglichkeit 10 sind Elemente durch verschiedene Formen dargestellt.

Feature 9 Auf Basis der Darstellungsmöglichkeit 11 sind die Elemente durch verschiedene Farben dargestellt.

Feature 10 Auf Basis der Darstellungsmöglichkeit 12 sind den Elementen verschiedene Symbole, mit einer 1-zu-1 Beziehung, zugeordnet.

Neben den bisher genannten graphischen Visualisierungskomponenten, ist die Beschriftung dieser Komponenten mindestens genauso wichtig [LNHE14], das sogenannte „Dual Coding“ [Moo09]. Dabei sind kurze Beschreibungen sehr gut [LNHE16], da sie nicht viel Platz verbrauchen, aber es sollte darauf geachtet werden, dass die Texte nicht abgeschnitten werden [Kri09]. Bei den Beschriftungen kann noch zusätzlich darauf geachtet werden, dass technische Begriffe durch leichter verständlichere Begriffe ersetzt werden [CSM09]. Dies ist für die Neulinge in dem Bereich von Vorteil, da sie dann leichter die Beziehungen und Komponenten verstehen können.

Feature 11 Da Beschriftungen zu einem besseren Verständnis führen, hat jedes Element im Graph eine Beschriftung, wie in Darstellungsmöglichkeit 13 erläutert.

Feature 12 Damit auch Nicht-Experten die Bedeutungen leichter verstehen, sind die fachlichen Begriffe durch Verständlichere ersetzt.

Kriglstein und Wallner [KW10] beschreiben, dass Knoocks (Abschnitt 4.1.2) überwiegend positives Feedback bekommt. Unter anderem wurde die klare Trennung der Visualisierung von hierarchischen und nicht-hierarchischen Relationen genannt. Hierbei werden hierarchische Relationen in Form einer zoombaren Visualisierung und nicht-hierarchische Relationen in Form eines Node-link Tree dargestellt.

Feature 13 In der Visualisierung sind hierarchische und nicht hierarchische Relationen zu unterscheiden.

Zusätzlich lässt sich die Visualisierung um eine weitere Dimensionen bereichern. Hier gibt es verschiedene Optionen zwischen 2.5D und 3D. Der große Vorteil einer weiteren Dimension ist der neue virtuelle Platz, da der Mensch auf dreidimensionales Denken optimiert ist, der auf gleich großem Bildschirmplatz entsteht. [FNS13; HMM00; SF11]

Feature 14 Die Visualisierung ist mit mehr als zwei Dimensionen realisiert.

Eine weitere Option ist die Komponenten-Multiplikation [LNHE16] und die unsichtbaren Links [HHDK00], um mehr Übersichtlichkeit in die Visualisierung zu bringen. Eine dieser Varianten kommt für MUSE4Anything in Frage, da hierbei unter anderem Nicht-Experten von der Visualisierung profitieren.

Feature 15 Zur Vermeidung von Kantenschneidungen wird die unter Darstellungsmöglichkeit 15 zuerst genannte Variante, die mehrfache Komponentendarstellung, verwendet.

Feature 16 Zur Vermeidung von Kantenschneidungen wird die unter Darstellungsmöglichkeit 15 zuletzt genannte Variante, die unsichtbaren Links, verwendet.

3.3.2. Interaktionsmöglichkeiten

In Abschnitt 3.2 werden verschieden Möglichkeiten zur Interaktion mit der Visualisierung vorgestellt. Auf Basis dieser werden in diesem Abschnitt nun Features definiert.

Feature 17 Die Interaktionsmöglichkeit 1, die Suche nach bestimmten Elementen in der Ontologie, ist vorhanden.

Feature 18 Der Benutzer hat mit Interaktionsmöglichkeit 2 die Möglichkeit nur bestimmte Elemente der Ontologie sich darzustellen.

Feature 19 Durch die Interaktionsmöglichkeit 4 kann der Benutzer den gesamten Graphen und einzelne Elemente verschieben.

Feature 20 Mit einer dynamischen Detailansicht werden dem Benutzer mehr Informationen über ein ausgewähltes Element dargestellt.

Um die Visualisierung in einem kleinen Bereich mit mehr Informationen anzuhäufen ist die Fisheye-Linse eine mögliche Option [DLSP18]. Mit Hilfe dieser Linse lassen sich auch komplexere Aufgaben leichter lösen [TAHS06]. Eine solche Linse ist hilfreich, aber in der Implementierung aufwendig.

Feature 21 Mit der Interaktionsmöglichkeit 7 kann der Benutzer sich mit der Fisheye-Linse mehr Informationen in einem bestimmten Bereich darstellen lassen.

Als letzter Punkt für die Visualisierung ist noch das Zoomen zu nennen. Hierbei gibt es zwei Varianten. Das geometrische Zoomen, welches den Normalfall abbildet, bei dem die Visualisierung vergrößert, beziehungsweise verkleinert wird. Die weitere Option ist das semantische Zoomen, welches die Visualisierung verbessert, da leichter zwischen verschiedenen Informationsebenen gewechselt werden kann [HMM00; WLA17]. Beide Varianten haben ihre Vorteile.

Feature 22 Mit der Interaktionsmöglichkeit 3 kann der Benutzer in der Darstellung zoomen.

4. Toolvergleich

Im folgenden Kapitel werden verschiedene Anwendungen zur Visualisierung von Ontologien auf Basis der Features aus Abschnitt 3.3 verglichen. Der Vergleich wird sich, wie auch die Features, in zwei Bereiche aufteilen. Im ersten geht es um die Darstellungsmöglichkeiten. Dieser Vergleich basiert auf wissenschaftlichen Arbeiten und Screenshots der jeweiligen Anwendung. Im zweiten Bereich geht es um die Interaktionsfunktionalitäten, welche diese Anwendung beinhaltet. Dieser Schritt wird nur mit den Anwendungen vollzogen, welche von den Darstellungsmöglichkeiten her am besten zu den Features aus Abschnitt 3.3 passen. Im letzten Teil werden die bestehenden Features einer Visualisierung aus Abschnitt 3.3 mit den Erkenntnissen aus dem Toolvergleich verglichen und gegebenenfalls angepasst.

4.1. Beschreibung existierender Anwendungen

Die Suche nach existierenden Anwendungen zur Visualisierung von Ontologien wurde mit der Suchmaschine Startpage[1] realisiert. Startpage leitet die Suchanfragen anonymisiert an Google weiter. Da die Suche über einen Startpage-Server geleitet wird, ist weder IP-abhängige Filterung, noch personalisiertes Benutzer-Tracking und Verknüpfung mit vorangegangenen Suchen möglich [Sta].

Für die Suche, am 07.07.2021, wurde folgender Suchbegriff verwendet „ontology visualization tools“. Die Auswahl der Anwendungen basiert auf den ersten 20 Ergebnissen. Die vollständige Ergebnisliste ist im Anhang A zu finden. Im Folgenden werden die gefundenen Anwendungen kurz beschrieben.

4.1.1. BioOntoVis

BioOntoVis ist durch eine Web-Anwendung aufrufbar und wurde für das BioPortal, welches ein paar Hundert Ontologien beinhaltet, entwickelt. In Abbildung 4.1 sind zwei Ausschnitte zu sehen. BioOntoVis besitzt neben geometrischem Zoomen auch semantisches Zoomen. Daneben kann nach Elementen gesucht werden und die Ontologie bearbeitet werden. Damit möglichst viele Ontologien mit BioOntoVis visualisiert und bearbeitet werden können, gibt es drei Visualisierungsarten: Baum-, Netzwerk- und Fischaugen-Ansicht. In Abschnitt 4.1.1 ist die Baumansicht zu sehen. Hier kann der Benutzer einzelne Elemente auf- und zuklappen. In der Netzwerkansicht sind diese als Rechtecke visualisiert und mit Kanten verbunden. Zur Unterscheidung der Elemente werden Farben verwendet.

[1] https://www.startpage.com/ (20.07.2021)

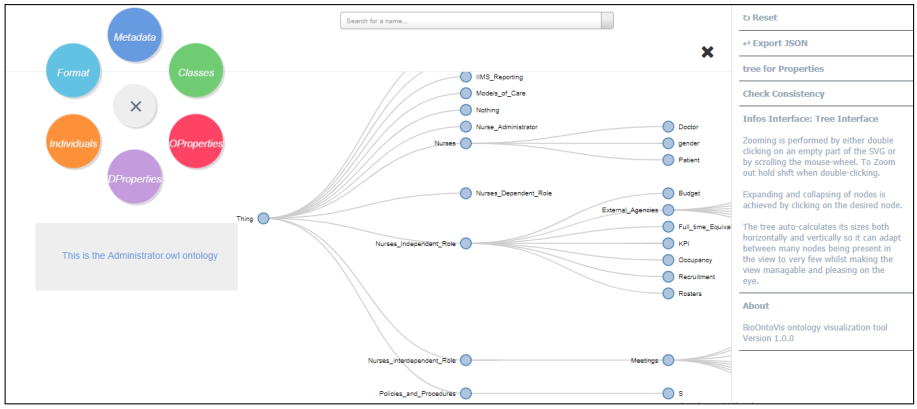


Abbildung 4.1.: BioOntoVis Screenshot - Baumansicht [AAB]

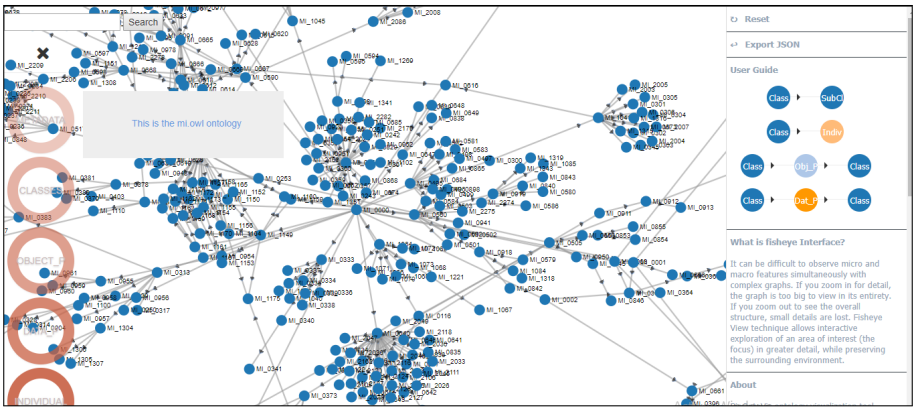


Abbildung 4.2.: BioOntoVis Screenshot - Fischaugen-Ansicht [AAB]

Daneben gibt es noch ein Detailinformationsbox. Bei großen Graphen kann es hier zu Verwirrung kommen. Deswegen gibt es noch die Fischaugen-Ansicht, welche in Abbildung 4.2 zu sehen ist. [AAB]

4.1.2. Knoocks

Die Visualisierung von Knoocks basiert aus einer Mischung eines Node-link Tree und einer Tree Map. In Abbildung 4.3 ist ein Screenshot von Knoocks zu sehen. Die Ontologie wird hier aus zwei verschiedenen Blickwinkeln betrachtet [BDM15] . Der eine (B) betrachtet die obersten hierarchischen Schichten der Ontologie. Die andere Sicht (A) stellt die einzelnen Blöcke dieser oberen Schichten, mit deren Eigenschaften, genauer dar. In der Visualisierung von Knoocks gibt es noch einen weiteren Teil, den Bereich der Werkzeuge (C). Hier kann nach Elementen unter anderem gesucht und gefiltert werden.

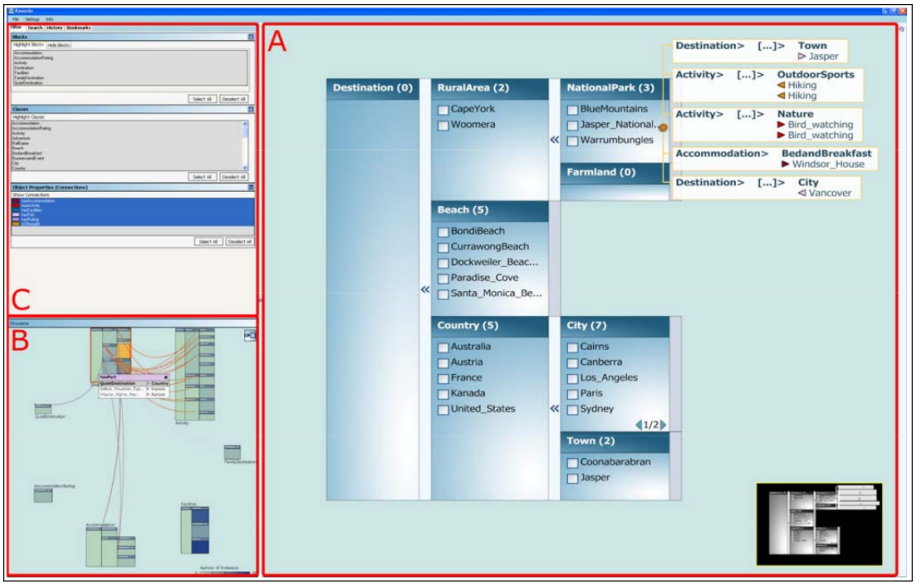


Abbildung 4.3.: Knoocks Screenshot - Die drei Hauptkomponenten von Knoocks: Hauptfenster (A), Vorschaufenster (B), Werkzeuge (C) [KW10]

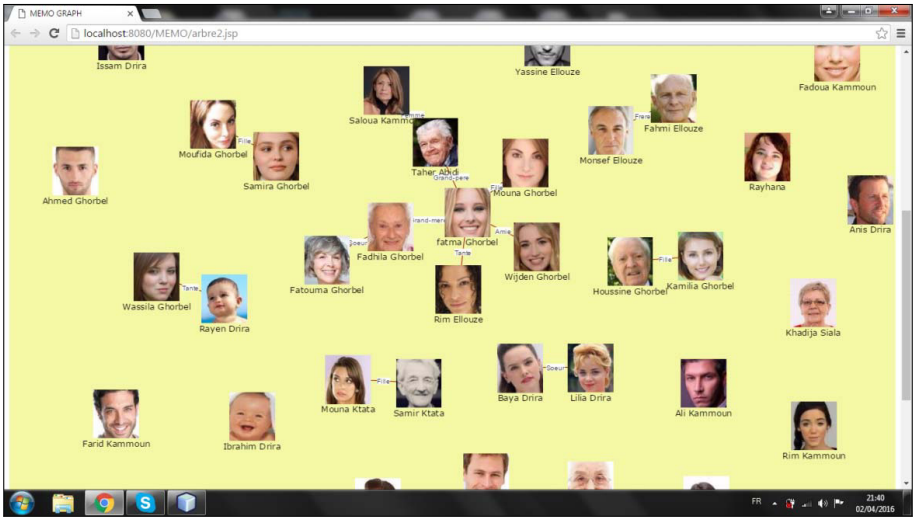


Abbildung 4.4.: MEMO GRAPH Screenshot [GEM+16]

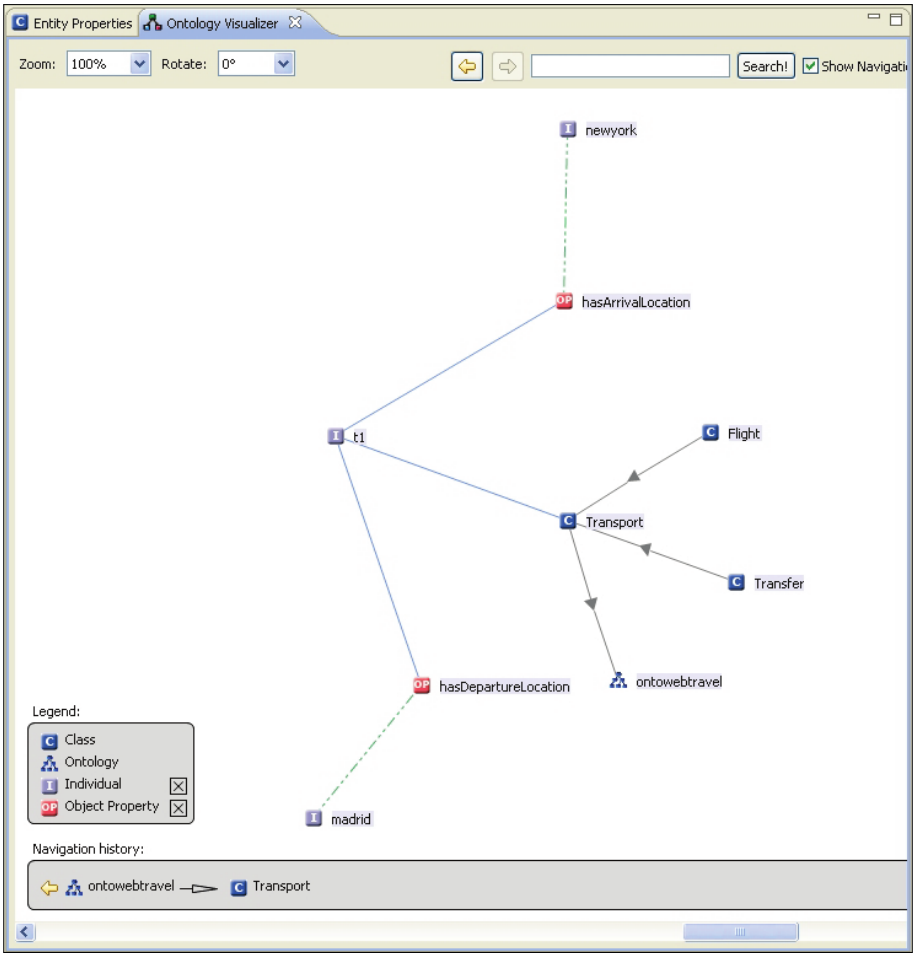


Abbildung 4.5.: Neon Toolkit Visualizer Screenshot [Sch]

4.1.3. MEMO Graph

Beim VIVA[2] Projekt geht es darum mit Hilfe einer Web Anwendung psychische Probleme zu lindern. Diese Anwendung ist speziell für Menschen mit Alzheimer im Anfangsstadium geeignet. Alzheimer Patienten haben besondere Beeinträchtigen, welche sie bei der Bedienung eines normalen User Interface (UI) beeinträchtigen. Um dieser Hürde entgegenzuwirken wurde MEMO GRAPH entwickelt. In Abbildung 4.4 ist MEMO Graph zu sehen, eine Ontologie Visualisierung, welche für jedermann entwickelt ist, für Experten genauso wie für Nicht-Experten. Die UI folgt der „design-for-all“ Philosophie, welche speziell Alzheimer Patienten bei der Bedienung hilft [GEM+16, S. 266].

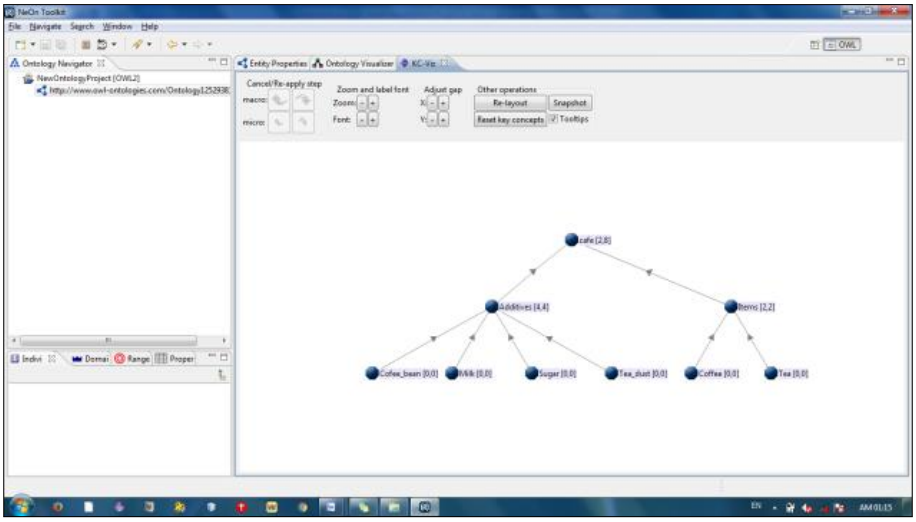


Abbildung 4.6.: KC-Viz Screenshot [FL20]

4.1.4. Neon Toolkit

Neon Toolkit ist ähnlich wie Protégé (Abschnitt 4.1.8) eine Anwendung für die Entwicklung von Ontologien. Es wurde als ein Projekt von der Europäische Union (EU) gefördert. In der Basisversion gibt es eine Listenansicht der Ontologie. Diese ist in Abbildung 4.5 zu sehen. Daneben gibt es noch eine einfache Visualisierung, welche die Ontologie in einem Node-link Tree mit kraftgesteuerter Anordnung darstellt. Zoomen ist möglich, beeinflusst aber nur die Länge der Links zwischen den Knoten. [DLSP18]

KC-Viz

KC-Viz ist eine Erweiterung für Neon Toolkit, welche in Abbildung 4.6 dargestellt ist. Es werden nur Klassen und deren hierarchische Beziehungen in einem Node-link Tree dargestellt. Die Besonderheit dieser Erweiterung ist die Visualisierung von „key concepts“. Dies bedeutet, dass mithilfe eines Algorithmuses die relevantesten Klassen der Ontologie ermittelt werden. Der Algorithmus arbeitet auf Basis von sieben Kriterien aus den Bereichen Psychologie, Linguistik und Wissensrepräsentation. Unter anderem wird die Häufigkeit der Verwendung der Klasse analysiert. Die sieben Kriterien werden einzeln bewertet. Anschließend wird die Gesamtbewertung aus einer gewichteten Summe der einzelnen Ergebnisse erstellt. Mit einem Rechtsklick auf ein Element werden Detailinformationen über dieses angezeigt. [DLSP18; MMP+11]

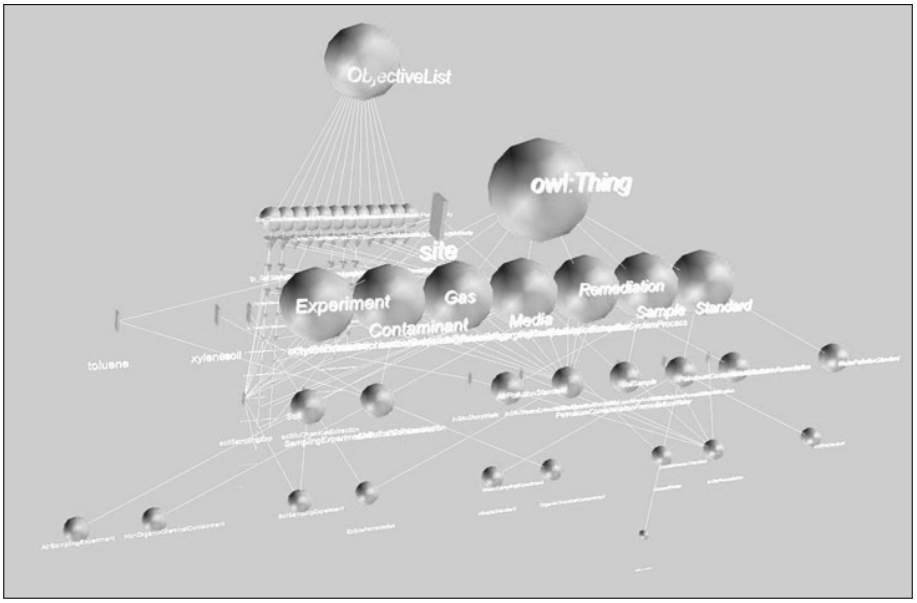


Abbildung 4.7.: Onto3DViz Screenshot [GC10]

4.1.5. Onto3DViz

Onto3DViz ist eine eigenständige Anwendung, welche für die 3D Visualisierung von Ontologien entwickelt wurde. Abbildung 4.7 zeigt einen Screenshot von Onto3DViz. Die Visualisierung basiert auf der Inferential Modeling Technique (IMT), einer Visualisierungsmethode für statische und dynamische Modelle. Statische Modelle bestehen aus Konzepten, dynamische aus Objekten. Jede Elementart der Ontologie hat eine andere Darstellungsvariante. Neben den Standardfunktionen wie Zoomen und Verschieben ist es ebenso möglich den Fokus auf einzelne Elemente zu lenken. Laut Dudas et al. sind Änderungen an der Darstellung nicht sehr flüssig und führen zu einer stockenden Visualisierung. [DLSP18; GC10]

4.1.6. Ontodia

Ontodia ist eine Web-basierte Anwendung zur Visualisierung von Ontologien. In Abbildung 4.8 ist ein Screenshot der Anwendung zu sehen. Die Visualisierung basiert auf einem Node-link Tree und ist an UML angelehnt. Hierarchische Beziehungen von Elementen werden in einer Baumansicht dargestellt. Durch Verschieben der Elemente, kann die Visualisierung frei angepasst werden. Ebenso können einzelne Elemente ein und ausgeblendet werden. Damit die passenden Elemente leichter gefunden werden, ist eine vollständige Suche integriert. [DLSP18; MPE+15]

[2]Vivre à Paris avec Alzheimer en 2030 grâce aux nouvelles technologies

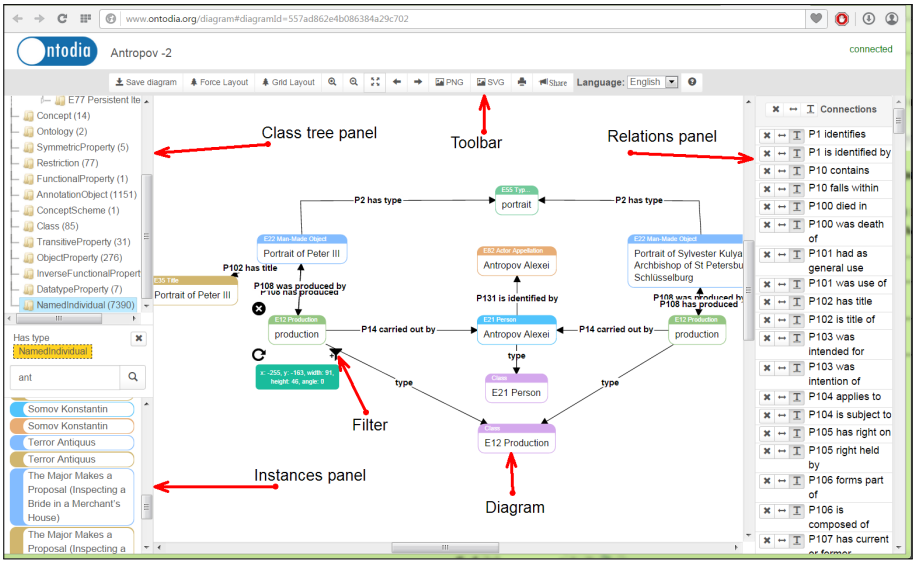


Abbildung 4.8.: Ontodia Screenshot [MPE+15]

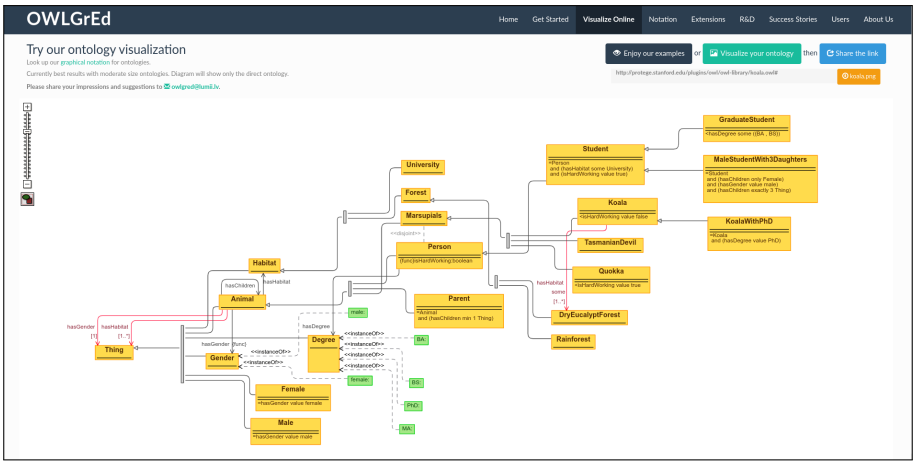


Abbildung 4.9.: OWLGrEd Web Screenshot [OWL21]

4.1.7. OWLGrED

Von OWLGrED (graphical OWL editor) gibt es einen Desktop Editor und eine Web-Anwendung zur Visualisierung von Ontologien. Beide Varianten sind in Abbildung 4.9 und Abbildung 4.10 zu sehen. Die Visualisierung von OWLGrED basiert auf UML Klassendiagrammen. Neben verschiedenen Algorithmen zur Visualisierung ist eine Suchfunktion, zoomen, verschieben von Elementen und Zusammenarbeit mit Protégé (Abschnitt 4.1.8 implementiert. Zusätzlich bietet OWLGrED einen farbigen Hintergrundrahmen für Teilbereiche der Ontologie, zur leichteren Unterscheidung der restlichen Bereiche. [BBČ+10]

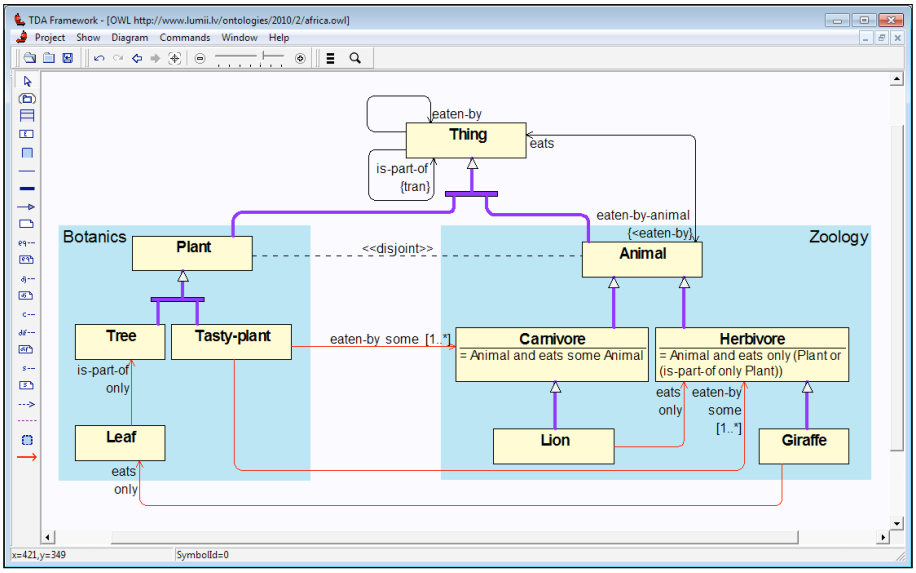


Abbildung 4.10.: OWLGrEd Desktop Screenshot [BBČ+10]

4.1.8. Protégé

Das Projekt Protégé bietet zwei Varianten des Open-Source Ontologie-Editors an. Zum einen Web-Protégé und zum anderen Protégé Desktop. Für letzteres gibt es mehrere Visualisierungsverfahren, welche teilweise in den folgenden Abschnitten dargestellt werden. [Pro] In der Grundversion liefert Protégé schon eine Indented List Ansicht mit. [DLSP18]

Jambalaya

Jambalaya, in Abbildung 4.11 abgebildet, ist ein komplexes Visualisierungsplugin für Protégé 3. Es kombiniert einen Node-link Tree mit einer Indented List. Jambalaya besitzt eine der flexibelsten Visualisierungsmöglichkeiten. Es gibt verschiedenste vordefinierte Ansichten, welche dem Benutzer helfen sollen möglichst schnell an das Ziel zu kommen. Alle Elemente und Instanzen der Ontologie sind als Knoten und die Relationen als Kanten dargestellt. Elemente können auf- und zugeklappt werden [DLSP18]. Jambalaya hat die Besonderheit, dass bei einer der sechs Visualisierungen [BDM15; FL20] Kindelemente in den Eltern positioniert werden, was in Abbildung 4.11 zu sehen ist. Dies ist eine Kombination eines Node-link Tree und der zoombaren Visualisierung.

Ontograf

Ontograf ist für Protégé 4 entwickelt. Hier ist die Visualisierung als ein Node-link Tree dargestellt, wie in Abbildung 4.12 zu sehen. Elemente sind als Rechtecke visualisiert. Es sind verschiedene Anordnungsalgorithmen implementiert, welche frei gewählt werden können [FL20]. Ebenso ist unendliches geometrisches Zoomen vorhanden. Leider ist die Navigation nur in inkrementellen Schritten möglich, was bedeutet, dass es teilweise aufwendig ist alles zu sehen. Neben dem Graphen gibt es zusätzlich eine Listenansicht, in welcher das darzustellende Element für den Graphen

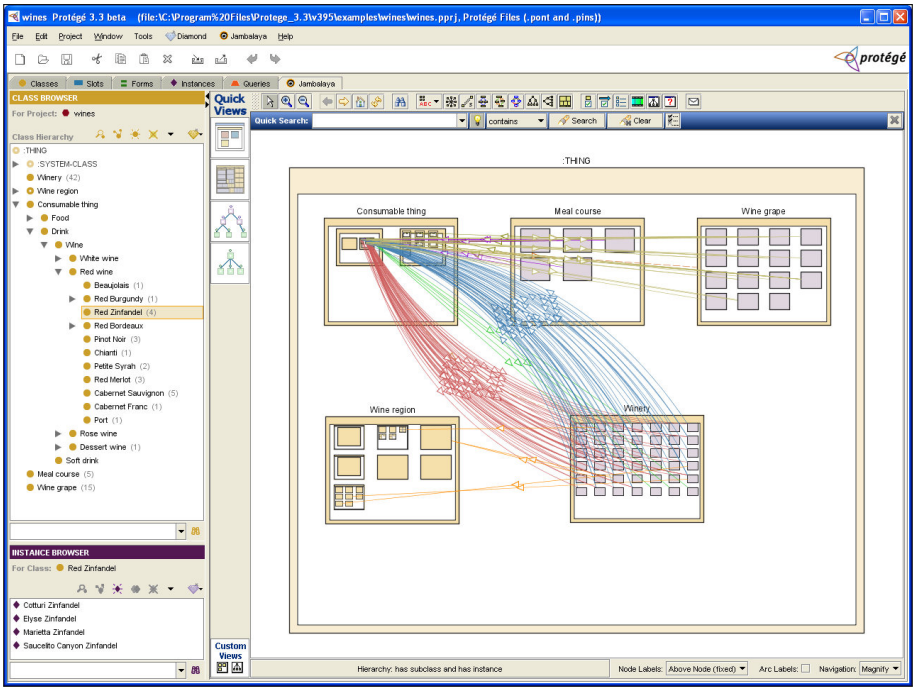


Abbildung 4.11.: Jambalaya Screenshot [CHI21]

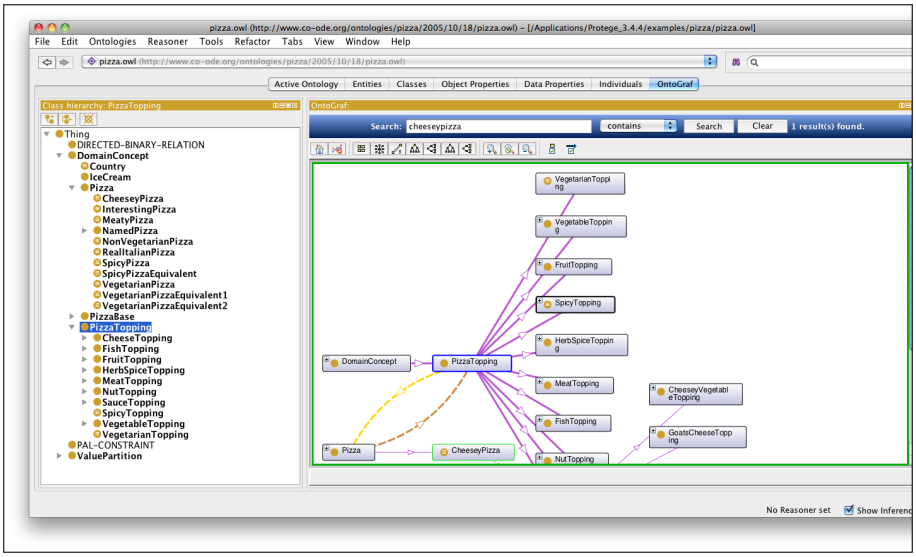


Abbildung 4.12.: Ontograf Screenshot [Fal10]

auswählbar ist. Beschriftungen an Elementen werden, sobald diese zu klein sind, automatisch ausgeblendet. Detailinformationen gibt es in einem Tooltipp zu den einzelnen Elementen. Es gibt auch eine Suche, der Fokus wird hierbei automatisch auf das gefundene Element geändert, was zu Veränderungen der Visualisierung führt. [DLSP18]

OntoSphere3D

In Abbildung 4.13 ist Ontosphere3D eine 3D Ontologievisualisierung für Protégé und Eclipse zu sehen. Die dreidimensionale Visualisierung basiert auf einem Node-link Tree. Es werden immer nur drei Hierarchieebenen parallel angezeigt, durch Navigieren kann zwischen den einzelnen Ebenen der Ontologie gewechselt werden. [DLSP18; Ontb] Es gibt verschiedene Perspektiven um die Ontologie zu untersuchen. Die Hauptansicht, die Taxonomieansicht, eine Detailansicht eines Konzeptes und die Instanzenansicht. In der Hauptansicht werden alle Elemente und deren Beziehungen dargestellt. In der Taxonomieansicht werden die Konzepte und Relationen dargestellt. In der Detailansicht eines Konzeptes wird auf die Konzepte der Taxonomien eingegangen und in der Instanzansicht werden alle Instanzen eines Konzeptes dargestellt. [BDM15]

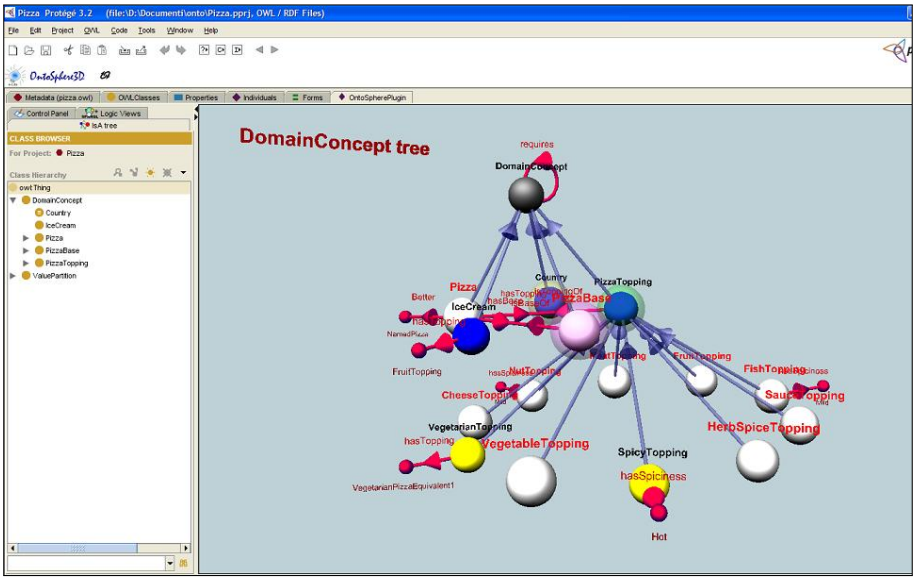


Abbildung 4.13.: Ontosphere3D Screenshot [Onta]

OWL2UML

OWL2UML visualisiert die Ontologie auf Basis von UML, welche in Abbildung 4.14 zu sehen ist. Dieses Plugin wird aber nicht mehr weiterentwickelt. Die Entwicklung wird aber in OWLGrED (Abschnitt 4.1.7) weitergeführt. [Pro]

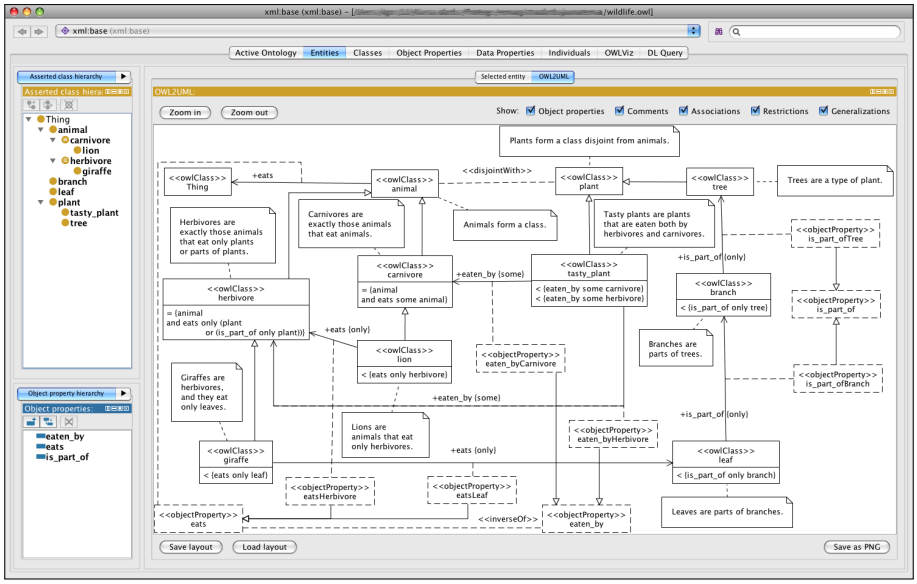


Abbildung 4.14.: OWL2UML Screenshot [Ist10]

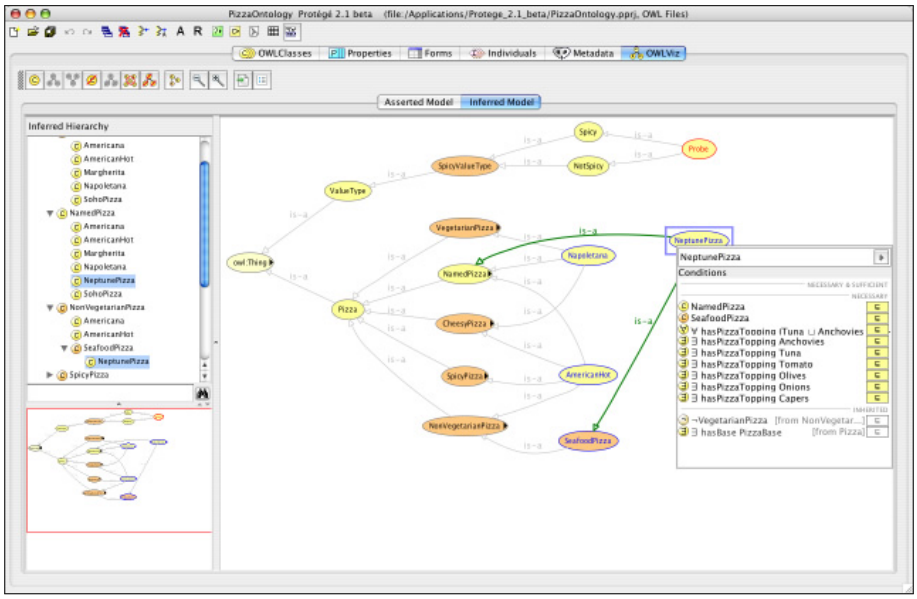


Abbildung 4.15.: OWLViz Screenshot [Hor10]

OWLViz

OWLViz ist ein weiteres Plugin für Protégé, zu sehen in Abbildung 4.15, welches auch direkt mitgeliefert wird. Bei dieser Erweiterung werden nur Klassen, keine Eigenschaften in einem Node-link Tree visualisiert. Das Plugin ist mit der Listenansicht von Protégé verknüpft. Die Anzahl der anzuzeigenden Elemente ist einstellbar. Es kann aber auch die gesamte Ontologie visualisiert werden. [DLSP18]

ProtégéVOWL

ProtegeVOWL ist wie WebVOWL eine Implementierung von Visual Notation for OWL Ontologies (VOWL)[VOW]. ProtégéVOWL ist aber nicht so vollständig wie WebVOWL (Abschnitt 4.1.9) implementiert. In einem späteren Absatz wird WebVOWL vorgestellt, daher wird auf eine genauere Untersuchung von ProtégéVOWL verzichtet.

SOVA

Simple Ontology Visualization API (SOVA) ist in Abbildung 4.16 abgebildet und ein Visualisierungsplugin für Protégé 4. Die Visualisierung der Ontologie in SOVA wird durch einen Node-link Tree realisiert. Jedes Element aus der Ontologie wird als ein einzelnes Element im Graphen dargestellt. Eigenschaften sind auch ein Element. Es kann entweder die komplette Ontologie visualisiert werden, oder immer drei Ebenen. Geometrisches Zoomen ist vorhanden. Filtern ist nach Elementtypen möglich, aber nicht nach speziellen Elementen. Dateneigenschaften werden nicht dargestellt. [BDM15; DLSP18]

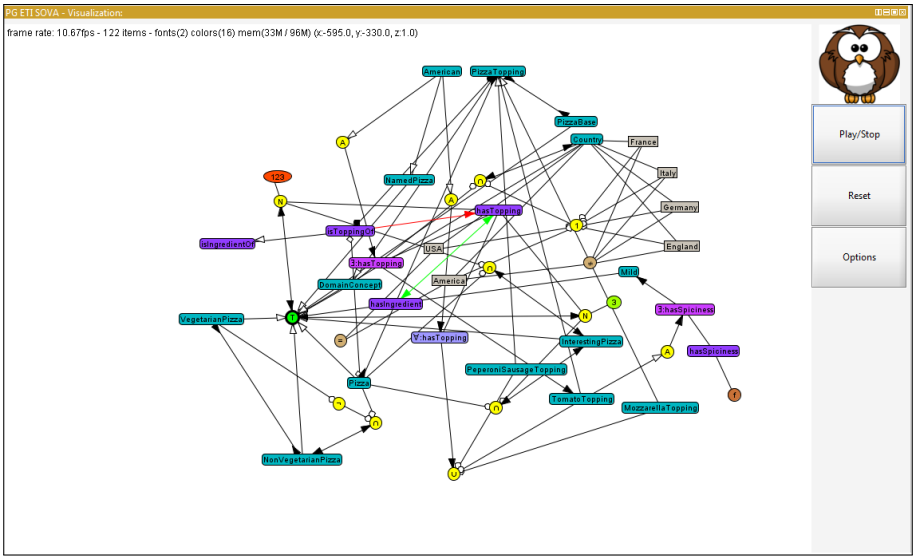


Abbildung 4.16.: SOVA Screenshot [Kun12]

TGViz

TGViz ist ein Plugin für Protégé 3. In Abbildung 4.17 ist ein Screenshot von TGViz zu sehen. TGViz stellt die Ontologie als einen Node-link Tree mit kraftgesteuerter Anordnung dar. Daneben wird die Ontologie noch als eine Indented List repräsentiert. Diese zwei Darstellungen sind miteinander gekoppelt. Es werden keine Formen für die Visualisierung benutzt, ausschließlich Beschriftungslabels. Die Ontologie kann inkrementell angezeigt werden. Eine Besonderheit in TGViz ist die Fisheye-Ansicht, welche aber versteckt ist. Mithilfe eines Rechtsklicks auf ein Menü werden verschiedene Aktionen für das gewählte Element dargestellt. Doppelklick wird nicht als Zoomen interpretiert sondern positioniert das gewählte Element in der Mitte und sorgt für

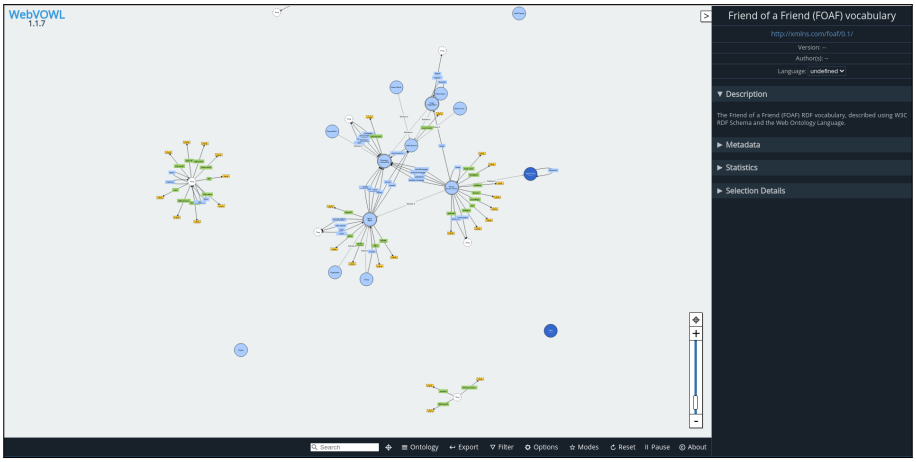


Abbildung 4.18.: WebVOWL Screenshot [Web21a]

.

eine Neuanordnung des restlichen Graphen. Damit der Graph übersichtlich bliebt, gibt es die Einstellmöglichkeit der maximal anzuzeigenden Elemente und der maximal aufzuklappenden Elemente. In TGViz ist geometrisches Zoomen und das Verschieben des Graphen implementiert. [Ala03; DLSP18]

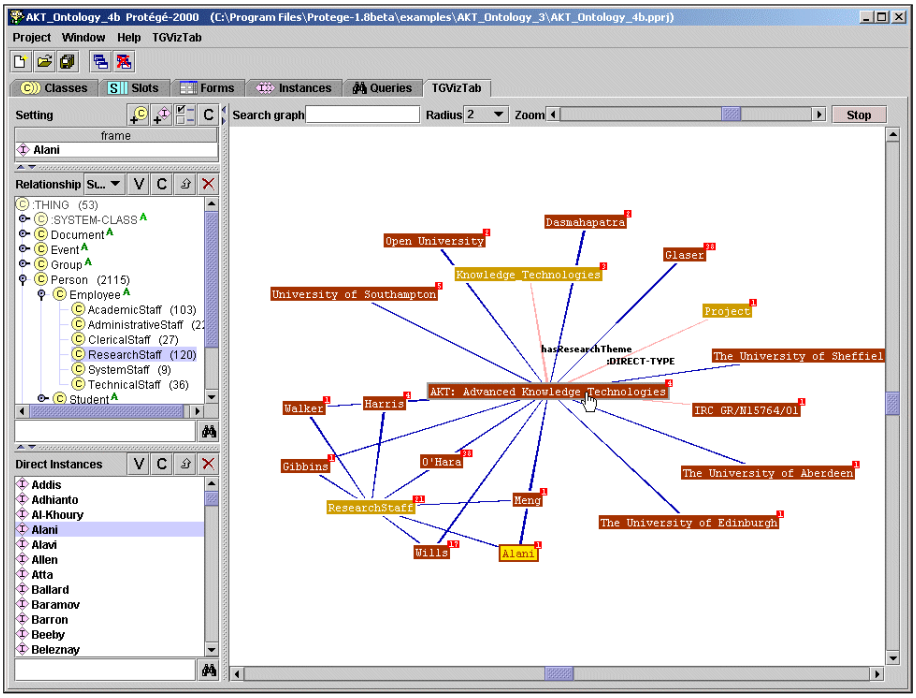


Abbildung 4.17.: TGViz Screenshot[Ala03]

4.1.9. WebVOWL

WebVOWL ist eine Implementierung von VOWL. In Abbildung 4.18 ist ein Screenshot von WebVOWL zu sehen. WebVOWL visualisiert die Ontologie als Node-link Tree mit kraftgesteuerter Anordnung, welcher zusätzlich mit Abstoßung- und Anziehungsanimationen bei der Interaktion erweitert ist. Die Anwendung ist als Web-Anwendung implementiert und basiert auf D3. Sie ist in drei Bereiche aufgeteilt. Die Graphenvisualisierung, den Kontrollbereich und eine Seitenleiste mit erweiterten Informationen. Dem Benutzer werden verschiedenste Interaktionsmöglichkeiten geboten. Es gibt geometrisches Zoomen mit dem Mausrad, mittels Doppelklick oder Steuerungselement. Für Detailanalysen kann der Graph vergrößert werden. Genauso kann er für eine Übersichtsdarstellung verkleinert werden. Der Graph sowie einzelne Elemente können verschoben werden. Es besteht die Möglichkeit nach einzelnen Elementen zu suchen, welche auch hervorgehoben werden können. Zu diesen Elementen werden weitere Informationen angezeigt. Daneben gibt es vier Möglichkeiten zu filtern: „Dateneigenschaft, einzelne Unterklasse, Disjunktheitsinformation und Mengenoperatoren“ [FL20]. Die automatische Anordnung des Graphen kann angepasst werden. Datentyp-Objekte können gesondert behandelt werden, sodass diese näher an den dazugehörigen Elementen angeordnet werden. [BDM15; LNHE14] Ab 01.Oktober 2021 wird WebVOWL nicht mehr online als Service angeboten [web21], aber es gibt die Möglichkeit es selber zu betreiben. Der Quellcode ist auf github veröffentlicht [Web21b].

4.1.10. yWorks - Ontology Visualizer

yFiles [yFi21] ist eine kommerzielle Programmierbibliothek welche explizit für die Diagrammvisualisierung entwicklet wurde und somit perfekt für Ontologievisualisierungen geeignet ist. yFiles bietet verschiedene Algorithmen für eine klare Darstellung. In Abbildung 4.19 ist ein Screenshot der Ontologievisualisierungs Demo Anwendung von yWorks zu sehen. In der Demo Anwendung lässt sich nach Elementen suchen, welche anschließend fokussiert werden. In einem kleinen Tooltipp werden Detailinformationen angezeigt. Durch die Selektion eines Elementes werden dessen Eltern und Kindelemente hervorgehoben. Für die Anordnung der Elemente gibt es drei Varianten: hierarchisch, kreisförmig und organisch. [yWo]

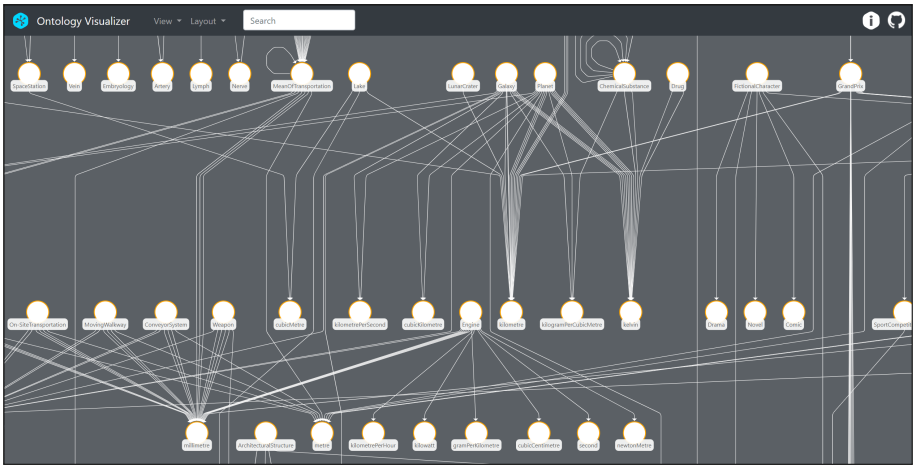


Abbildung 4.19.: yWorks Screenshot [yWo21]

4.2. Vergleich der Darstellungen

Im folgenden Abschnitt werden die vorherigen aufgeführten Tools anhand der Visualisierungskriterien aus Abschnitt 3.3 tabellarisch verglichen. Der Vergleich hat zwei Ziele.

Zum einen werden mit diesem Vergleich die Visualisierungsfeatures mit den Anwendungen verglichen, ob die erarbeiteten Features in der Praxis bis jetzt Anwendung gefunden haben oder nicht.

Auf der anderen Seite werden durch diesen Vergleich die passendsten Anwendungen auf die erarbeiteten Features ermittelt, sodass diese in einem weiteren Schritt genauer analysiert werden können.

Der Vergleich basiert auf der in Abschnitt 4.1 beschriebenen Untersuchung, basierend auf Screenshots, Papern und Anleitungen der Anwendungen.

Für den Vergleich gibt es vierstufiges Punktesystem:

- „+“ Das Feature wird voll erfüllt. Gibt in dem Vergleich zwei Punkte.
- „0“ Das Feature wird teilweise erfüllt. Gibt in dem Vergleich einen Punkt.
- „-“ Das Feature wird nicht erfüllt. Gibt in dem Vergleich einen Punkte Abzug.
- „? “ Keine Informationen vorhanden. Gibt in dem Vergleich keine Punkte.

Durch dieses Vergleichssystem werden Anwendungen, welche Features vollständig erfüllen zwar deutlich besser bewertet, was aber im Endeffekt zu einer besseren Grundlage der weiteren Anwendungsuntersuchung führt. Weniger als null Punkte kann das Punktesystem einer Anwendung nicht zuordnen.

Aus diesem Vergleich erfüllen die folgenden fünf Anwendungen am meisten der Features aus Abschnitt 3.3

- Knoocks
- Jambalaya
- OWLGred
- Ontodia
- WebVOWL

Diese Anwendungen werden nun anschließend auf die Interaktionsfunktionalitäten (Abschnitt 3.2) überprüft. Da es für Knoocks keine verfügbare Implementierung gibt, rückt Ontosphere3D, als nächster im Vergleich, in die Liste der zu untersuchenden Anwendungen. Aber auch Ontosphere3D konnte leider nicht untersucht werden, da wie schon Dudas et al. [DLSP18] in ihrem Paper geschrieben haben, gibt es zwar Ontosphere3D noch zum Herunterladen, aber es ist nicht lauffähig, weil es Fehler im Quellcode beinhaltet. Eine mögliche Alternative dafür ist Onto3DViz, da auch hier die Ontologie mit einer 3D Visualisierung dargestellt wird. Für Onto3DViz gibt es, genauso wie für Knoocks keine verfügbare Version. Daher werden nur vier Anwendungen weiter untersucht. Diese Sammlung von Tools bildet auch fast die gesamten Kriterien für die Darstellung ab.

		BioOntoVis	Knoocks	MEMO GRAPH	Neon Toolkit	KC-Viz	Onto3DViz	Ontodia	OWLGrEd	Protégé	Jambalaya	Ontograf	OntosphereJD	OWLViz	SOVA	TGViz	WebVOWL
	Fensterbereiche																
1	Node-link Diagramm	+	+	+	+	+	+	+	+	-	+	+	+	+	+	+	+
	Indented List	+	-	-	+	-	-	+	-	+	+	+	+	+	-	+	-
	Detailansicht	+	+	+	-	+	-	+	+	+	+	+	-	+	-	-	+
	Übersichtsdarstellung	-	+	-	-	-	-	-	0	-	-	-	0	+	-	-	0
2	**Verknüpfung der Visualisierungen**	-	+	-	+	-	-	+	-	+	+	+	+	+	-	+	-
3	**Zoomable (Kindelemente in Eltern)**	-	-	-	-	-	-	-	+	-	+	-	-	-	-	-	-
	Elementdarstellung																
4	Farben	+	+	-	0	-	+	+	+	+	+	+	+	-	+	+	+
	Formen	-	-	-	-	-	-	+	-	+	-	-	-	-	+	-	+
	Symbole (1-zu-1 Beziehung)	-	0	+	+	-	-	-	-	-	-	0	-	-	-	-	-
5	**Stark verbundene Komponenten**	?	+	+	-	-	-	-	+	-	-	-	-	-	-	-	-
6	**Beschreibungslabels**	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
7	**Technische Begriffe ersetzen**	-	?	+	-	?	-	-	-	+	0	-	-	-	?	-	-
8	**Relationsunterscheidung**	-	+	-	-	-	-	+	+	-	+	+	-	-	+	-	+
9	**2.5/3D für weit entfernte Kanten**	-	-	-	-	-	+	-	-	-	-	-	+	-	-	+	-
10	**Fisheye-Linse**	+	-	-	-	-	-	-	+	-	+	-	-	-	-	+	-
11	**Komponenten multiplizieren**	-	0	-	-	-	-	-	0	-	-	-	?	-	-	-	0
12	**Unsichtbare Links**	-	+	-	-	-	-	-	-	-	-	-	?	-	-	-	-
	Zoomen																
13	geometrisch	+	+	+	0	+	+	+	+	-	+	-	+	+	+	+	+
	semantisch	+	-	-	-		-			-		-				-	0
	Punktebewertung	6	16	2	0	0	0	9	13	2	14	4	7	3	1	6	8

Tabelle 4.1.: Darstellungsvergleich der Anwendungen

		Jambalaya	OWLGred	Ontodia	WebVOWL
1	**Suchen**	+	-	+	+
	Gefundene Ergebnisse werden markiert	0	-	-	+
2	**Filtern**				
	Nach Elementnamen filtern	+	-	0	-
	Nach Elementtyp filtern	+	-	0	+
	Nur bestimmte Elemente darstellen	0	-	-	0
3	**Focus & Kontext**	+	-	0	0
4	**Detailansicht zu ausgewählten Elementen**				
	Popover	-	-	-	-
	Neues Fenster	+	+	+	-
	Ansicht in Fensterbereich	-	-	-	+
5	**Elemente verschieben**	+	+	+	+
6	**Gesamten Graph verschieben**	0	+	+	+
7	**Zoomen (Mausrad, Doppelklick)**	0	0	0	+
8	**Abstände von Elementen sind anpassbar**	-	-	-	+
9	**Dargestellte Elemente sind limitierbar**	0	-	-	+
	Bewertung	14	0	6	17

Tabelle 4.2.: Interaktionsvergleich der Anwendungen

4.3. Vergleich der Interaktionsmöglichkeiten

Im folgenden Abschnitt werden die Interaktionsfunktionalitäten ausgewählter Anwendungen untersucht. Die Auswahl der Tools basiert auf der Abschlussbewertung der Anwendungen aus dem vorherigen Abschnitt und der Einzigartigkeit von Anwendungen. Hierfür wurden die einzelnen Funktionen bei den Anwendungen in der Praxis untersucht. Das Vergleichssystem ist dasselbe wie in Abschnitt 4.2.

Wie in der Tabelle 4.2 zu sehen ist, ist die Diskrepanz zwischen den Anwendungen, bezogen auf die erarbeiteten Interaktionsfunktionalitäten aus Abschnitt 3.2, relativ groß. WebVOWL und Jambalaya schneiden gut ab, im Gegensatz dazu haben OWLGred und Ontodia mit den erarbeiteten Funktionen aus Abschnitt 3.3 sehr wenig gemeinsam. Der einzige Nachteil den WebVOWL bezogen auf die Bedienung hat, ist die fehlende Filterung von einzelnen Elementen. Nach diesen kann gesucht werden und sie werden in Form von markierten Umrandungen markiert, aber die Ontologie kann nicht auf eine bestimmte Menge an Elementen reduziert werden. Bei Jambalaya ist die Abdeckung der Funktionen teilweise nicht mit dem komplett gewünschten Umfang realisiert. Daher kann für die Interaktionsgrundlage WebVOWL als Referenzanwendung für MUSEAnything genutzt werden. Wie schon in Abschnitt 4.1.9 beschrieben, ist der Quellcode von WebVOWL auf GitHub verfügbar.

4.4. Finale Anforderungen an die Visualisierung

Im letzten Abschnitt dieses Kapitels werden die Erkenntnisse des bisherigen Kapitels mit der ersten Version einer Liste an Features für die Implementierung in MUSE4Anything aus Abschnitt 3.3 kombiniert. Aus dieser Kombination werden dann die Finale Anforderung (FA) für die Ontologievisualisierung in MUSE4Anything definiert.

4.4.1. Visualisierung

Im folgenden Abschnitt geht es um die finalen Anforderungen der Ontologievisualisierung für MUSE4Anyhting.

Dreiteilige Darstellung

Die vorherige benannte Bekanntheit des Node-link Tree (Feature 2) spiegelt sich auch in der Häufigkeit der Verwendung bei den einzelnen Anwendungen wieder. Von den untersuchten sechzehn Anwendungen, haben fünfzehn einen Node-link Tree zur Visualisierung der Ontologie. Die Indented List (Feature 1) ist im Gegensatz dazu nicht ganz so häufig zu finden. Diese wird bei neun Anwendungen verwendet. Was aber wiederum in mehr Anwendungen Platz gefunden hat, ist die Detailansicht. Diese wurde in elf Anwendungen eingesetzt, was deren Notwendigkeit unterschreibt. Eine zusätzliche Übersichtsdarstellung (Feature 5) kommt nur in seltenen Fällen zum Einsatz. Diese gibt es nur in zwei Anwendungen, was aber aufgrund der Studienergebnisse in Abschnitt 3.3 bei MUSE4Anyhting trotzdem implementiert wird. Bei etwas mehr als der Hälfte der getesteten Anwendungen sind die einzelnen Visualisierungen miteinander verknüpft, was zeigt, dass dies hilfreich ist (Feature 6).

Die zoombare Visualisierung (Feature 7) kommt nur in wenigen Anwendungen vor (zwei Stück), da es ein spezieller Ansatz ist, dessen Vorteile und Nachteile in Abschnitt 3.1.1 erläutert wurden. Aufgrund der teilweise komplexen Ontologien von MUSE4Anything (Abschnitt 2.3) ist dieser Ansatz leider nicht sinnvoll verwendbar.

Finale Anforderungen: dreiteilige Darstellung

FA-1 Die Hauptvisualisierung der Ontologie soll durch einen Node-link Tree realisiert werden.

FA-2 Zusätzlich soll es eine Indented List geben.

FA-3 Die dritte Darstellung soll eine Übersichtsdarstellung des Node-link Tree sein.

FA-4 Alle drei Darstellungen sollen miteinander verknüpft sein, sodass Interaktionen in einer Darstellung Einfluss auf die anderen nehmen.

Layout des Node-link Tree

Die Positionierung (Feature 4) von stark verbundenen Komponenten eng beieinander ist bei den getesteten Anwendungen selten zu finden. Dieser Anordnungsmechanismus kann trotzdem implementiert werden, da er, wie im vorherigen Abschnitt beschrieben, Vorteile beim Verständnis bringen kann. Durch die Verwendung des kraftgesteuerten Layouts (Feature 3) im Node-link Tree, wird diese Eigenschaft schon mitgenutzt.

Die zwei vorgestellten Möglichkeiten zu Kantenschneidungsverminderung (Feature 15) werden in fast keiner Anwendung eingesetzt. Unsichtbare Links werden in Knoocks verwendet. In MUSE4Anything können die unsichtbaren Links zur Verweisung auf Kindelemente verwendet werden. Für die Relation zwischen Typen und Taxonomien sollen unsichtbare Links verwendet werden. Dies hat den Vorteil, dass Eigenschaften eines Typen, welche eine Referenz auf einen anderen Typ oder eine Taxonomie sind, kompakt gehalten werden können und nicht komplett dargestellt werden müssen.

Finale Anforderungen: Layout des Node-link Tree

FA-5 Die Elemente im Node-link Tree sollen mit Hilfe eines kraftgesteuerten Anordnungsalgorithmus positioniert sein.

FA-6 Die Positionierung soll durch den Benutzer veränderbar sein.

FA-7 Einzelne Elemente sollen an ihrer Position fixierbar sein.

FA-8 Der Benutzer soll den Anordnungsalgorithmus jederzeit neu berechnen können.

FA-9 Elemente, die eine Referenz auf einen Typ oder eine Taxonomie sind, sollen als unsichtbarer Link eingefügt werden.

Elementvisualisierung

Bei der Elementdarstellung verwenden die Anwendungen verschiedene Möglichkeiten. Am häufigsten werden unterschiedliche Farben (Feature 9) verwendet. Formen (Feature 8) sind häufiger zu finden als Symbole (Feature 10), was auf die Implementierungshürde zurückzuführen ist. Farben sind leichter zu verwenden als verschiedene Formen. Am größten ist der Aufwand bei der Verwendung von verschiedenen Symbolen. Die Kombination aus allen dreien macht aber die Visualisierung am effektivsten für den Benutzer [LNB14; Moo09].

Finale Anforderungen: Elementgestaltung

FA-10 Im Node-link Tree sollen verschiedene Elementtypen, Typen und Taxonomien durch unterschiedliche Formen dargestellt werden.

FA-11 Im Node-link Tree sollen verschiedene Elementtypen, Typen und Taxonomien durch unterschiedliche Farben dargestellt werden.

FA-12 Im Node-link Tree sollen Eigenschaften von Typen zusätzlich durch ein passendes Symbol gekennzeichnet werden.

FA-13 Elementen in der Indented List sollen die gleichen Symbole wie im Node-link Tree zugeordnet werden.

Beschriftung von Elementen

Die Notwendigkeit von Beschreibungslabels (Feature 11) der Komponenten wird durch den Vergleich in Tabelle 4.1 eindeutig klar. Alle untersuchten Anwendungen fügen den Elementen eine Beschriftung hinzu. Die Ersetzung (Feature 12) von technischen Begriffen wird eher selten verwendet, was aber im Fall von MUSE4Anything ein starkes Mittel zu besseren Verständlichkeit sein kann, da es hier um die Visualisierung der Ontologien auch für Nicht-Experten geht. Diese Ersetzung kann in einer zukünftigen Version der Ontologievisualisierung in MUSE4Anything implementiert werden, da dies keine grundlegende Einschränkung in der Benutzung betrifft.

Finale Anforderungen: Beschriftung

FA-14 Der Benutzer soll den vollständigen Namen eines Elementes im Node-link Tree und in der Indented List einsehen können.

Hierarchie

Die Trennung der hierarchischen und nicht hierarchischen Beziehungen von Elementen (Feature 13) wird bei ungefähr der Hälfte der untersuchten Anwendungen vorgenommen. Da in der vorher erwähnten Studie [KW10], dies auch von den Benutzern als positiv gewertet wurde, soll dieser Aspekt bei der Visualisierung für MUSE4Anything auch beachtet werden.

Finale Anforderungen: Hierarchiedarstellung

FA-15 Im Node-link Tree sollen Kindelemente, angelehnt an die zoombare Visualisierung, innerhalb der Elternelemente positioniert sein.

FA-16 In der Indented List sollen Kindelemente aufklappbar und eingerückt unter den Eltern aufgelistet sein.

Weitere Dimensionen

Die 2.5D und 3D Visualisierung ist nur in ein paar wenigen Anwendungen implementiert [DLSP18]. Der Einsatz von weiteren Dimensionen (Feature 14) ist in nur drei der untersuchten Anwendungen zu finden. Diese Erweiterung kann große Vorteile bringen, ist aber aufwendig in der Implementierung. Die Fisheye-Linse ist bei vier Anwendungen zu finden, aber auch noch selten, da es ebenso ein aufwendiger Schritt ist. Aufgrund des großen Implementierungsaufwandes ist diese Möglichkeit nicht in der ersten Version der Ontologievisualisierung geplant.

4.4.2. Interaktionen

Dieser Teil der finalen Anforderungsdefinition konzentriert sich auf die Interaktionsfeatures, welche in Abschnitt 3.3 erarbeitet und in Tabelle 4.2 verglichen wurden.

Suche

Um bei der Arbeit mit den Ontologien schnell an das gewünschte Element zu gelangen, ist eine Suche das effektivste Mittel (Feature 17), dies spiegelt sich auch in der Häufigkeit beim Interaktionsvergleich wider. Drei der getesteten Anwendungen beinhalten eine Suche. Das Markieren von gefundenen Elementen im Diagramm unterstützt nur eine Anwendung vollständig, ist aber eine sehr zeitsparende und hilfreiche Funktion.

Finale Anforderungen: Suchfunktion

FA-17 Die Suche soll die Namen aller Elemente durchsuchen.

FA-18 Gefundene Elemente sollen hervorgehoben werden. Zur besseren Unterscheidung sollen nicht zutreffende Elemente ausgeblendet sein.

FA-19 In der Indented List sollen Elternelemente, falls ein Kindelement zu den Suchergebnissen gehört, aufgeklappt sein.

Filtern

Beim Filtern der Elemente (Feature 18) ist die Filterung nach dem Elementtyp am häufigsten zu finden, was aus der Implementierungssicht die einfachste Möglichkeit ist. Die Filterung nach dem Namen ist seltener zu finden, aber dennoch eine mächtige Funktion. In der ersten Version der Ontologievisualisierung soll die Möglichkeit der Elementtypenfilterung und die Darstellung ausgewählter Elemente implementiert werden. Die Filterung nach Elementnamen wird in einer weiteren Version folgen.

Finale Anforderungen: Filterfunktion

FA-20 Der Benutzer soll die Möglichkeit haben im Node-link Tree nur bestimmte Elemente darzustellen.

FA-21 Der Benutzer soll die Möglichkeit haben im Node-link Tree nur Elemente einer bestimmten Art, also Typen und Taxonomien, darzustellen.

FA-22 Der Benutzer soll die Möglichkeit haben ausschließlich selektierte Elemente der Ontologie darzustellen.

Fokus & Kontext

Die Funktion des Fokus & Kontext ist eine Variante der Ergebnismarkierung, wird aber eher selten in vollem Umfang angeboten. Wie in Abschnitt 3.2.4 erläutert, ist diese Methode auf der einen Seite sehr hilfreich, kann aber auch verwirrend sein. Daher wäre es eine Option, dass diese Funktion individuell aktiviert und deaktiviert werden kann. Aufgrund der Vor- und Nachteile dieser Funktion, wird diese Funktion als ein rein optionales Feature implementiert, mit der Option der Aktivierung und Deaktivierung.

FA-23 Der Benutzer soll die Funktion Fokus & Kontext optional aktivieren können, damit Suchergebnisse in der Mitte des Graphen positioniert werden.

Detailinformationen

Bei der Detailansicht (Feature 20) zu ausgewählten Elementen ist auch eine starke Tendenz zu erkennen. Drei der vier Anwendungen öffnen ein neues Fenster. Nur WebVOWL hat einen Bereich im aktuellen Fenster für die Detailansicht definiert. MUSE4Anything wird sich an der Variante von WebVOWL orientieren, da beides Web-Anwendungen sind und hier das effektive Arbeiten mit einem weiteren Fenstern nicht sinnvoll möglich ist. Im Web gibt es Popup Blocker, welche das Öffnen eines neuen Fensters verhindern. Es besteht die Möglichkeit einen neuen Tab zu öffnen, dies sollte nur bei einem Kontextwechsel getan werden. Daher wird die Darstellung im gleichen Fenster in einem extra Bereich realisiert.

Finale Anforderungen: Detailinformationsansicht

FA-24 Für jedes Element soll es eine zusätzliche Detailinformationsansicht mit allen relevanten Information über dieses Element geben.

FA-25 In dieser Ansicht soll es auch einen Link zur Einzelansicht dieses Elements von MUSE4Anything geben.

Verschieben & Zoomen

Das Verschieben von einzelnen Elementen (Feature 19) und des ganzen Graphen ist bei so gut wie allen Anwendungen vollständig implementiert. Ebenso wie verschiedene Arten des Zoomens. Bei den Möglichkeiten des Zoomens (Feature 22) ergibt sich ein eindeutiges Bild der untersuchten Anwendungen. Geometrisches Zoomen ist in allen untersuchten Anwendungen vorhanden. Hier sollte neben dem Zoomen durch das Mausrad der Doppelklick implementiert werden, da dieser auf Touchscreens die Standardaktion zum Zoomen ist. Zusätzlich kann noch ein Interaktionsfeld den Benutzern helfen. Im Gegensatz dazu ist semantisches Zoomen sehr selten zu finden.

Finale Anforderungen: Verschieben & Zoomen

FA-26 Die Darstellung des Node-link Tree soll sich auf mehrere Arten zoomen lassen: Doppelklick, Mausrad und Interaktionstaster.

FA-27 Der Node-link Tree und darin enthaltene Elemente soll sich auf zwei Möglichkeiten verschieben lassen: mit der Maus und durch Interaktionstaster.

Parametereinstellungen

Die Möglichkeit verschiedene Einstellungen für die Visualisierung vorzunehmen, ermöglicht dem Benutzer ein passenderes Ergebnis der Visualisierung zu erzeugen. So ist die Auswahl verschiedener Algorithmen für das Layout im Node-link Tree von Vorteil. Die Einstellung von Abständen einzelner Elemente ist nur bei WebVOWL zu finden. Ebenso bietet nur diese Anwendung die Möglichkeit der Limitierung der dargestellten Elemente.

Finale Anforderungen: Parametereinstellungen

FA-28 Der Benutzer soll die maximale Anzahl standardmäßig dargestellter Eigenschaften eines Typen definieren können.

FA-29 Der Benutzer soll den Abstand der Elemente beeinflussen können.

5. Prototyp der Visualisierung

In dem vorletzten Kapitel wird auf die Datenaufbereitung eingegangen, Entwürfe von Mockups auf Basis der Visualisierungskriterien werden erstellt und anschließend implementiert. Abschließend wird die Implementierung mit den finalen Anforderungen aus Abschnitt 4.4 verglichen.

5.1. Mockup

Im folgenden Abschnitt werden aus den finalen Anforderungen für die Visualisierung aus Abschnitt 4.4 Mockups erstellt, welche anschließend in MUSE4Anything implementiert werden.

In Abbildung 5.1 ist der erste Entwurf der kompletten Visualisierung abgebildet. Das Graphical User Interface (GUI) teilt sich in vier Bereiche auf, welche in den vorangegangenen Kapiteln erarbeitet wurden. Die primäre Visualisierung wird mit einem Node-link Tree (2) dargestellt. Im vorliegenden

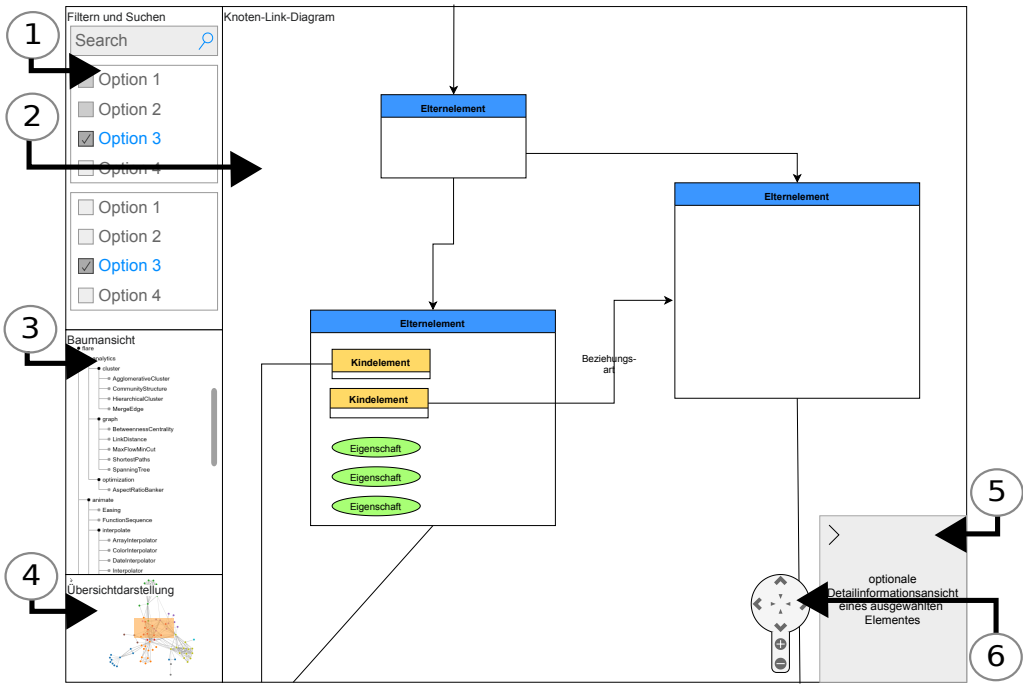


Abbildung 5.1.: Mockup: Gesamtansicht mit Ausschnitt des Node-link Tree. (1) ist der Filter und Suchbereich; (2) ist die Hauptvisualisierung, der Node-link Tree; (3) ist die zweite Visualisierung, die Indented List; (4) ist die Übersichtsdarstellung mit Positionsausschnitt des Node-link Tree; (5) ist die Detailansicht des ausgewählten Elementes; (6) ist das Interaktionsfeld

Mockup ist ein Ausschnitt einer Ontologie zu sehen, welche vergrößert ist. Diese Darstellung wird durch eine Übersichtsdarstellung (4) unterstützt. Daneben gibt es noch einen Indented Tree (3) und einen Filter- und Suchbereich (1). Die dynamische Detailansicht (5) eines Elementes wird im rechten Bereich positioniert. Für die Interaktion mit dem Graphen gibt es ein Interaktionsfeld (6).

Filtern und Suchen Es soll verschiedene Funktionalitäten des Filterns und Suchens geben. Zum einen soll es die Möglichkeit geben nach den Namen der Elemente in der Ontologie zu suchen. Suchergebnisse sollen anschließend in beiden Visualisierungen markiert werden. Daneben soll es die Möglichkeit geben nur bestimmte Elementtypen (Taxonomien oder Typen) selektiv darzustellen. Zusätzlich lassen sich auch nur ausgewählten Elementen darstellen. In diesem Bereich sollen ebenso Parameter zur Visualisierung anpassbar sein. Dazu zählt die Anzahl der anzuzeigenden Eigenschaften eines Typen. Der Abstand der Elemente in der Visualisierung soll auch einstellbar sein.

Node-link Tree Die Hauptdarstellung der Ontologie wird durch ein Node-link Tree realisiert. Wie in Abschnitt 3.1.1 aufgelistet gibt es hierfür mehrere Anordnungsmöglichkeiten. Implementiert wird in diesem Fall die kraftgesteuerte Anordnung der Elemente, da dies, wie in Abschnitt 3.1.1 beschrieben, die hilfreichste Anordnung erzeugt. Eigenschaften eines Elementes werden innerhalb des Elternelementes angeordnet. In Abbildung 5.1 ist dies durch die gelben rechteckigen und grünen ovalen Elemente dargestellt. Elemente im Graph werden sich durch verschiedene Formen und Farben unterscheiden lassen.

Indented List Die Indented List stellt die hierarchische Struktur der Ontologie dar. Kindelemente sind deren Elternelementen untergeordnet und entsprechend eingerückt (Abbildung 3.1a). Elemente mit Kindern lassen sich auf- und zuklappen, um die Kindelemente temporär auszublenden. Die Baumansicht ist mit dem Node-link Tree verknüpft. Dies bedeutet, ausgewählte Elemente in einer der zwei Visualisierungen werden ebenso in der anderen Darstellung markiert. Zur Unterscheidbarkeit verschiedener Elemente in dieser Ansicht, hat jeder Eintrag ein zusätzliches Symbol. Die selben Symbole werden auch im Node-link Tree verwendet, beziehungsweise stellen die Elementform in klein dar.

Übersichtsdarstellung Die Übersichtsdarstellung ist eine Kleindarstellung des Node-link Tree. Zusätzlich wird durch ein Rechteck der aktuelle sichtbare Ausschnitt des Node-link Tree dargestellt. Dies ist in Abbildung 5.1 durch das orange Rechteck zu sehen. Interaktionen gibt es in der Übersichtsdarstellung keine. Diese dient lediglich der Orientierung. In dieser Darstellung werden alle Texte sowie Kindelemente von Taxonomien und Typen nicht dargestellt, da diese bei der Größe ohnehin nicht lesbar sind. Taxonomien und Typen werden aber in gleichen Größenverhältnissen dargestellt, was zur besseren Orientierung beider Graphen führt.

Detailansicht In der Detailansicht werden dynamisch zum ausgewählten Element mehr Informationen angezeigt. Dieser Bereich wird über dem Node-link Tree positioniert, ist aber nur sichtbar, sobald ein Element ausgewählt ist.

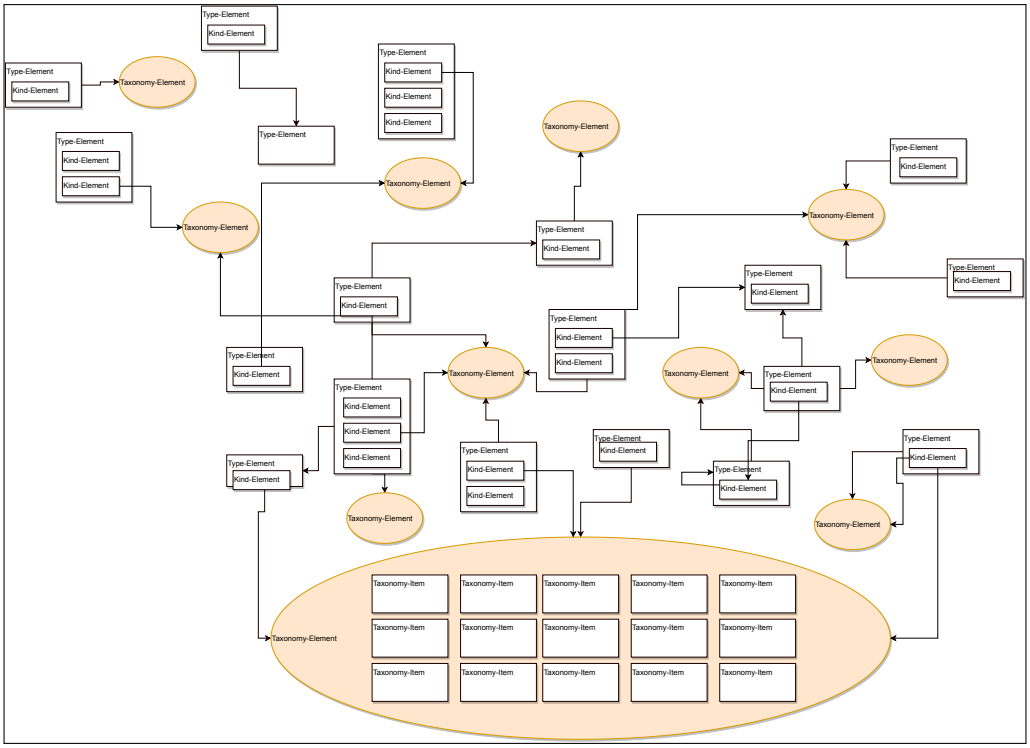


Abbildung 5.2.: Mockup: Startdarstellung des Node-link Tree. Typ-Elemente sind durch weiße Rechtecke und Taxonomie-Elemente durch orangene Ovale dargestellt. Relationen zwischen den Elementen werden durch Pfeile dargestellt. Taxonomien lassen sich aufklappen.

Interaktionsfeld Das Interaktionsfeld bietet neben den Grundfunktionen wie Zoomen und Verschieben des gesamten Graphen noch weiter Funktionen. Hierzu gehört die Maximierung beziehungsweise die Minimierung des Graphen. Ebenso kann die Visualisierung zurückgesetzt werden, sodass alle Elemente wieder sichtbar sind.

5.2. Implementierung

Im folgenden Abschnitt geht es um die Implementierung der erarbeiteten Visualisierung für MUSE4Anything.

In Abbildung 5.3 ist die erste Version der Implementierung zu sehen. In dieser Version waren die Elemente noch rein zufällig positioniert. Ebenso sind die Elemente nicht unterscheidbar. Es sind nur die Elternelemente dargestellt, Kindelemente sind noch nicht visualisiert. In der Übersichtsdarstellung wurde der dargestellte Ausschnitt des Node-link Tree nicht markiert. Daneben gab es noch keine Filter- und Suchfunktionen, das Interaktionsfeld und die Detailansicht fehlten auch noch.

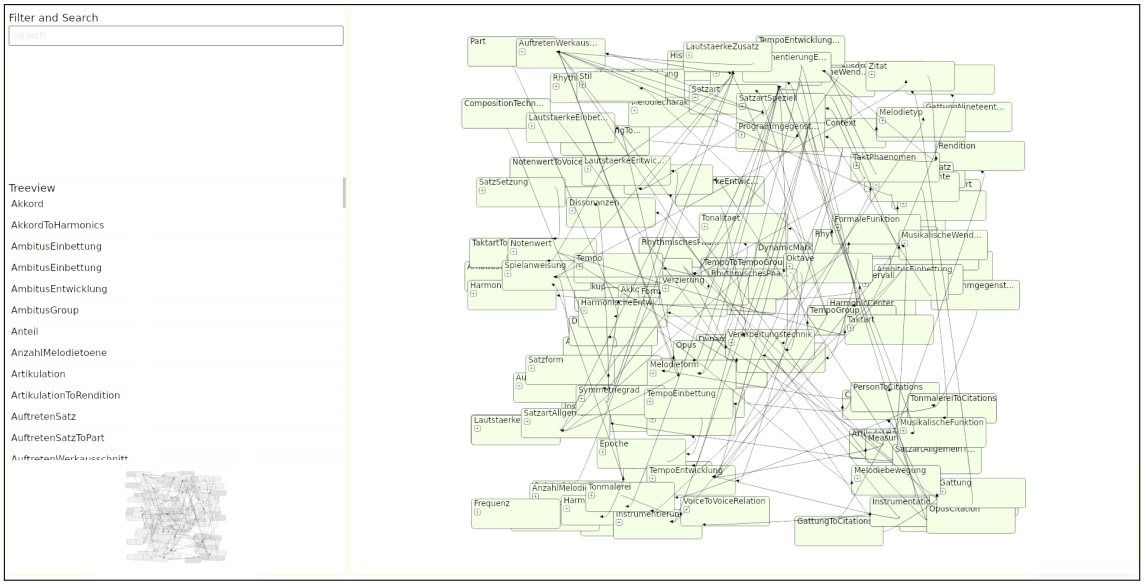


Abbildung 5.3.: Erste Version der Implementierung

5.2.1. MUSE4Anything - Architektur

MUSE4Anything ist nach dem Prinzip einer dreistufigen Cloud-Anwendung aufgebaut. Diese besteht aus drei Komponenten. Zum einen ist das das Frontend, welches mit einer HTTP API mit dem Backend kommuniziert. An das Backend ist noch eine Datenbank angeschlossen. Die Visualisierung der Ontologie wird im Frontend implementiert. Die Datenabfrage läuft über die API des Clients an das Backend.

Datenabfrage Bevor die Ontologie visualisiert werden kann, müssen erst die dafür notwendigen Daten über eine HTTP API abgerufen werden. Der Client interagiert mit dem Server mittels Hypermedia as the Engine of Application State (HATEOAS). Dies bedeutet, der Client hat kein Wissen über die Interaktionsmöglichkeiten mit dem Server, sondern der Server übermittelt bei jeder Abfrage dem Client die möglichen verfügbaren Funktionen. Somit werden die Navigationslinks zu den Ressourcen mit der Ressource mitgeschickt.

Um herauszufinden welche Ressourcen aktuell abrufbar sind, gibt es in MUSE4Anything einen „NavigationLinksService “. Mithilfe der verfügbaren Links in diesem Service kann überprüft werden, welche Ressourcen abgerufen werden können. Wenn der Benutzer nicht angemeldet ist, hat er keinen Zugriff auf die notwendigen Ressourcen, die für die Visualisierung der Ontologie notwendig sind. Somit kann er die Ontologie nicht sehen. Wenn er die Visualisierung sehen möchte, muss er angemeldet sein. In MUSE4Anything wird es in Zukunft ein Benutzermanagement, welches jedem Benutzer individuell Zugriff auf einzelne Ressourcen gewähren kann geben. Falls der angemeldete Benutzer Zugriff auf die entsprechenden Ressourcen hat, ist er auch berechtigt sich die Ontologie visualisieren zu lassen.

Der Zugriff auf diese Daten erfolgt mittels zweier Collection Ressources (deut. Ressourcen Sammlung). Eine solche Sammlung ist eine Sammlung von homogenen Ressourcen. Die Sammlungen sind zusätzlich noch aus Performancegründen mit Pagination aufgeteilt, sodass immer nur eine bestimmte

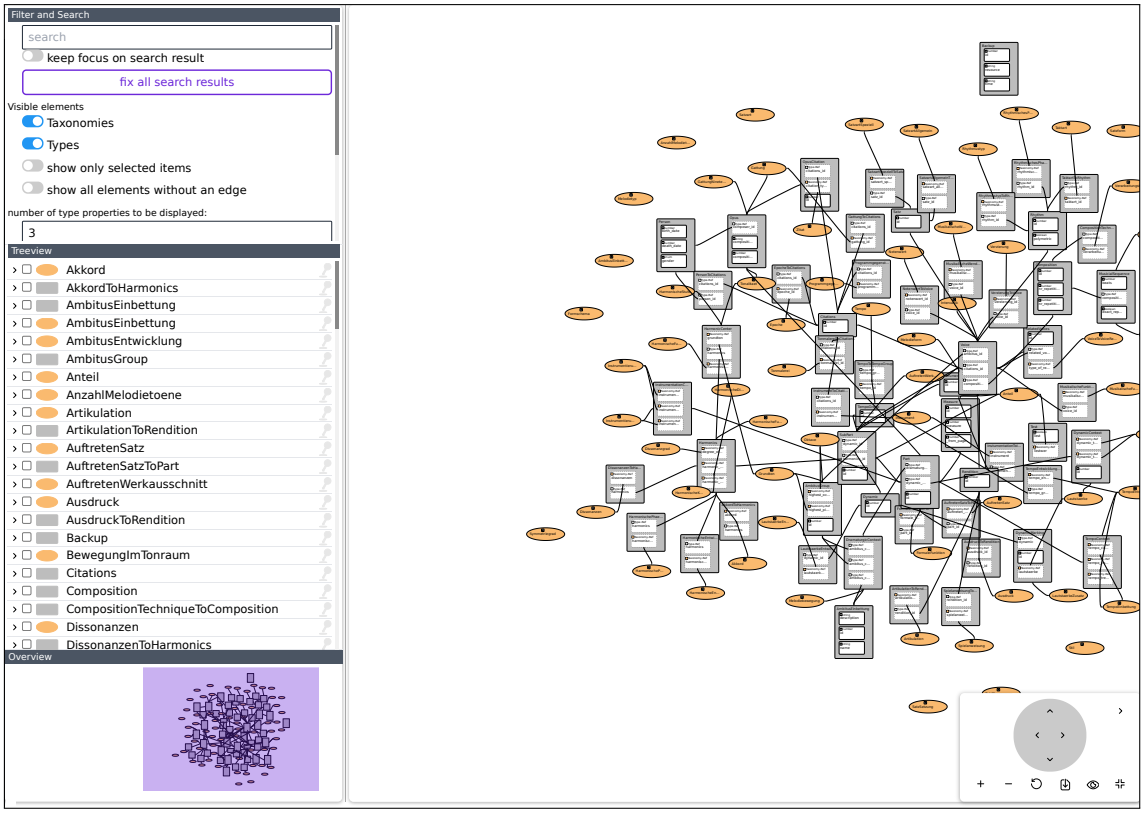


Abbildung 5.4.: Fullscreen Screenshot der Implementierung. Die Anordnung der einzelnen Bereiche entspricht denen des Mockups in Abbildung 5.1.

Anzahl an Elementen mit einer Abfrage abgerufen werden können. Die abgerufenen Ressourcen enthalten dabei immer auch weiterführende Links. Dies bedeutet zum Beispiel, dass eine Taxonomie Ressource die Links zu den Taxonomie Elementen enthält und eine Collection Ressource enthält, sofern vorhanden, den Link zu der nächsten Collection. Da mit jeder Abfrage die Information über mögliche weitere Collections mitgeliefert wird, kann das Frontend herausfinden, ob es noch einen weiteren Teil der Collection Ressource abrufen muss, oder ob alle Daten abgerufen sind. Zur Beschleunigung des Datenabrufs besitzt das Frontend einen lokalen Cache um die einzelnen Ressourcen zu speichern. Dadurch muss nicht mit jeder Abfrage die Schnittstelle des Backends angesprochen werden.

5.2.2. Dreiteilige Darstellung

Die Anforderungen an die dreiteilige Darstellung sind, wie in Abbildung 5.4 zu sehen, vollständig erfüllt. Im Hauptbereich ist der Node-link Tree, links daneben die Indented List und die Übersichtsdarstellung ist links unten. Die drei Visualisierungen sind miteinander verbunden. Die Übersichtsdarstellung und der Node-link Tree sind miteinander verknüpft, indem in der Übersichtdarstellung immer der aktuelle Ausschnitt des Node-link Tree ersichtlich ist. Die Selektion von ausgewählten Elementen in der Indented List wird auch auf den Node-link Tree angewendet. In der Indented List sind immer alle Kindelemente sichtbar. Im Node-link Tree werden immer nur die

maximale Anzahl an konfigurierten Elementen angezeigt. Die einzelnen Bereiche der Visualisierung sind in der Größe verstellbar. Zwischen dem linken Bereich und dem Node-link Tree ist eine horizontal verschiebbare Kante. Ebenso ist die Kante zwischen dem Filter/Suchbereich und der Indented List und zwischen der Indented List und der Übersichtsdarstellung verschiebbar.

Evaluation der finalen Anforderungen: dreiteilige Darstellung

FA-1 Die Hauptvisualisierung der Ontologie soll durch einen Node-link Tree realisiert werden.

Diese Anforderung ist vollständig erfüllt, da der Node-link Tree die primäre Visualisierung der Ontologie ist.

FA-2 Zusätzlich soll es eine Indented List geben.

Diese Anforderung ist vollständig erfüllt. Die gesamte Ontologie ist auch in einer Indented List dargestellt.

FA-3 Die dritte Darstellung soll eine Übersichtsdarstellung des Node-link Tree sein.

Diese Anforderung ist vollständig erfüllt. Die Übersichtsdarstellung stellt den Node-link Tree in klein dar.

FA-4 Alle drei Darstellungen sollen miteinander verknüpft sein, sodass Interaktionen in einer Darstellung Einfluss auf die anderen nehmen.

Diese Anforderung ist vollständig erfüllt. In der Übersichtsdarstellung ist der aktuelle Sichtbereich des Node-link Tree dargestellt. Ebenso sind die Aktionen der Elemente zwischen dem Node-link Tree und der Indented List verknüpft.

5.2.3. Node-link Tree

Die Ontologie ist unter anderem mit einem Node-link Tree visualisiert. Die Elemente im Graphen werden mit Hilfe eines Algorithmus für die Berechnung der kraftgesteuerten Anordnung angeordnet. Als erstes wurde für das Layout die „ngraph.forcelayout“ [Kas21b] Bibliothek verwendet, da diese sehr einfach zu implementieren ist. Diese Bibliothek bietet einen Algorithmus für die kraftgesteuerte Anordnung der Elemente für 2,3 oder mehr Dimensionen. Im aktuellen Fall wird dieser nur für die zweidimensionale Darstellung verwendet, die Bibliothek ist daher aber auch für die Zukunft geeignet, wenn der Visualisierung eine weitere Dimension hinzugefügt werden soll. Da die Bibliothek Teil von „ngrah.graph“ [Kas21a] ist, ist die Verwendung anderer graphenbezogener Algotithmen aus dieser Bibliothek sehr leicht implementierbar, weil die Datenbasis bei jedem Algorithmus die gleiche sein muss. Als zweites wurde noch die „d3-force“ [d3-21] Bibliothek implementiert. Der Benutzer kann sich den gewünschten Algorithmus auswählen. Die „d3-force“ Bibliothek ist mächtiger als die „ngraph.forcelayout“, aber auch aufwendiger in der Implementierung.

Der Benutzer hat die Möglichkeit jedes Element im Graphen individuell zu platzieren. Daneben kann er jedes Element fixieren, sodass dieses bei einer Neuberechnung des Algorithmus an der gleichen Position bleibt. Dies kann zum einen in der Detailansicht eines Elementes und in der Indented List mithilfe der Pinnnadel am Ende der Zeile erledigt werden. Nachdem der Benutzer einzelne Elemente fixiert hat, kann er die Anordnung der restlichen Elemente neu berechnen lassen.

Kindelemente, welche eine Referenz auf einen Typ oder eine Taxonomie darstellen, werden als unsichtbare Links eingefügt. Diese Elemente unterscheiden sich zudem in der Umrandung von den anderen Elementen, sie ist bei diesen gestrichelt. Kanten werden immer zwischen den Elementen direkt gezogen. Falls ein Kindelement, aufgrund der Beschränkung nicht angezeigt wird, geht die Kante vom Elternelement aus. Die Darstellung aller Kanten ist wichtig, um die Relationen zwischen den Elementen zu erkennen.

Evaluation der finalen Anforderungen: Layout des Node-link Tree

FA-5 Die Elemente im Node-link Tree sollen mit Hilfe eines kraftgesteuerten Anordnungsalgorithmus positioniert sein.

Diese Anforderung ist vollständig erfüllt. Es gibt zwei Algorithmen, mit welchen die Anordnung der Elemente berechnet werden kann.

FA-6 Die Positionierung soll durch den Benutzer veränderbar sein.

Diese Anforderung ist vollständig erfüllt. Der Benutzer kann jedes Element im Graph frei verschieben.

FA-7 Einzelne Elemente sollen an ihrer Position fixierbar sein.

Diese Anforderung ist vollständig erfüllt. Der Benutzer kann jedes Element im Graph an seiner Position fixieren, sodass es bei einer Neuberechnung an der selben Position bleibt.

FA-8 Der Benutzer soll den Anordnungsalgorithmus jederzeit neu berechnen können.

Diese Anforderung ist vollständig erfüllt. Der Benutzer kann manuell eine Neuberechnung der Positionen starten.

FA-9 Elemente, die eine Referenz auf einen Typ oder eine Taxonomie sind, sollen als unsichtbarer Link eingefügt werden.

Diese Anforderung ist vollständig erfüllt. Eigenschaften die eine Referenz auf einen Typ oder eine Taxonomie darstellen, sind als unsichtbarer Link vorhanden. Diese unterscheiden sich von den anderen Eigenschaften durch die gestrichelte Umrandung.

5.2.4. Indented List

In Abbildung 5.5 ist zu sehen, dass die Indented List vollständig implementiert ist. Wie zu sehen ist jeder Eintrag in der Liste mit einem Symbol markiert. Das Symbol stellt entweder die Form des Elementes im Node-link Tree dar oder ist ein Symbol für den Datentyp der Eigenschaft. Links des Symbols lassen sich mit der Auswahlbox Elemente dauerhaft hervorheben. Am „Opus“ Element ist zu sehen, dass die Elemente einzeln auf- und zuklappbar sind. Am Ende eines Listeneintrages ist die Pinnnadel. Mit dieser lässt sich das jeweilige Element im Graph fixieren oder lösen.

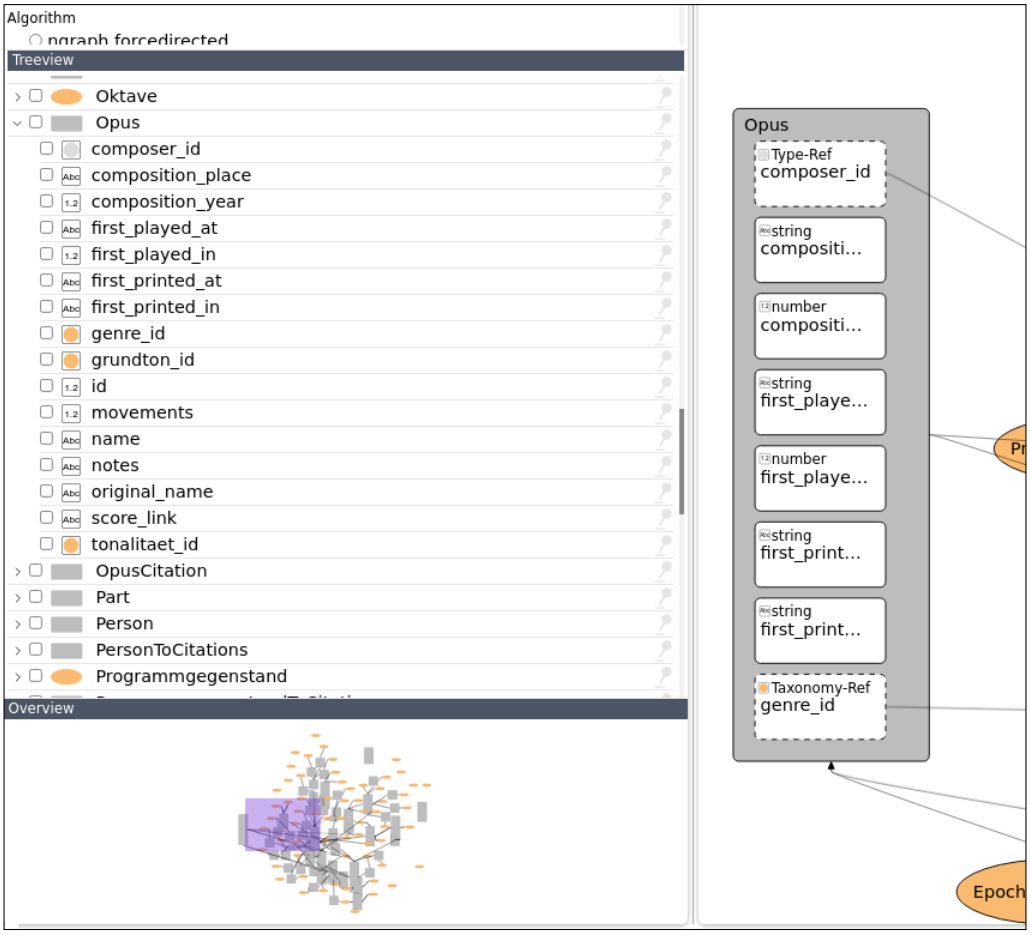


Abbildung 5.5.: Screenshot der Indented List. Zu sehen ist das Typ-Element „Opus“ sowohl in der Indented List, als auch im Node-link Tree. Ebenso ist zu sehen, wie den Eigenschaften in beiden Visualisierungen dieselben Symbole zugewiesen sind.

5.2.5. Elementgestaltung

In Abbildung 5.6 ist die Gestaltung eines Taxonomie-Elementes und eines Typ-Elements zu sehen. Es ist zu sehen wie die Elemente durch unterschiedliche Formen und Farben dargestellt sind. Ebenso ist jeder Eigenschaft eins Typen dasselbe wie in der Indented List zugeordnet. Es ist zu sehen, wie Kindelemente eines Typen, welche eine Referenz eines Typen oder einer Taxonomie sind nur als Referenzelement dargestellt sind und mit einer Kante auf das Zielelement zeigen. Zusätzlich haben Eigenschaften, welche eine Referenz auf einen anderen Typ oder eine Taxonomie darstellen, eine andere Umrandung als die anderen Eigenschaften.

Bei der Implementierung hat sich als sinnvoll erwiesen, dass im Node-link Tree Kanten zwischen Elementen leicht geschwächt angezeigt werden. Zur deutlichen Erkennung der Kanten eines Elementes werden die Kanten des ausgewählten Elementes hervorgehoben. Dies ist in Abbildung 5.6 dargestellt. Hier ist das „InstrumentationContext“ Element ausgewählt. Ebenso hat sich bei der

Implementierung herausgestellt, dass es sinnvoll ist die Elemente einer Taxonomie standardmäßig auszublenden. Diese sind durch den Erweiterungs- oder Minimier-Button in der oberen Mitte einer Taxonomie darstellbar oder versteckbar.

Evaluation der finalen Anforderungen: Elementgestaltung

FA-10 Im Node-link Tree sollen verschiedene Elementtypen, Typen und Taxonomien durch unterschiedliche Formen dargestellt werden.

Diese Anforderung ist vollständig erfüllt. Die Elemente im Node-link Tree sind durch verschiedene Formen dargestellt.

FA-11 Im Node-link Tree sollen verschiedene Elementtypen, Typen und Taxonomien durch unterschiedliche Farben dargestellt werden.

Diese Anforderung ist vollständig erfüllt, da Elemente im Node-link Tree durch verschiedene Farben dargestellt sind.

FA-12 Im Node-link Tree sollen Eigenschaften von Typen zusätzlich durch ein passendes Symbol gekennzeichnet werden.

Diese Anforderung ist vollständig erfüllt. Jedes Element im Node-link Tree hat ein eindeutiges Symbol zugeordnet.

FA-13 Elementen in der Indented List sollen die gleichen Symbole wie im Node-link Tree zugeordnet werden.

Diese Anforderung ist vollständig erfüllt. Den Elementen in der Indented List sind die selben Symbole wie im Node-link Tree zugeordnet.

5.2.6. Beschriftung

Wie in Abbildung 5.6 zu sehen, hat jedes Element einen Beschriftungstext. Dieser zeigt den Namen des Elementes an. Falls dieser zu lang sein sollte, wird dieser im Node-link Tree mit der mitgelieferten Funktion angepasst oder in den restlichen Darstellungen mittels CSS Eigenschaften abgeschnitten. Die Anzeige des vollständigen Namens durch einen Tooltipp ist sowohl im Node-link Tree als auch in der Indented List realisiert.

Evaluation der finalen Anforderungen: Beschriftung

FA-14 Der Benutzer soll den vollständigen Namen eines Elementes im Node-link Tree und in der Indented List einsehen können.

Diese Anforderung ist vollständig erfüllt. Der Benutzer hat überall in der Visualisierung die Möglichkeit mittels Tooltipps den vollständigen Namen einzusehen.

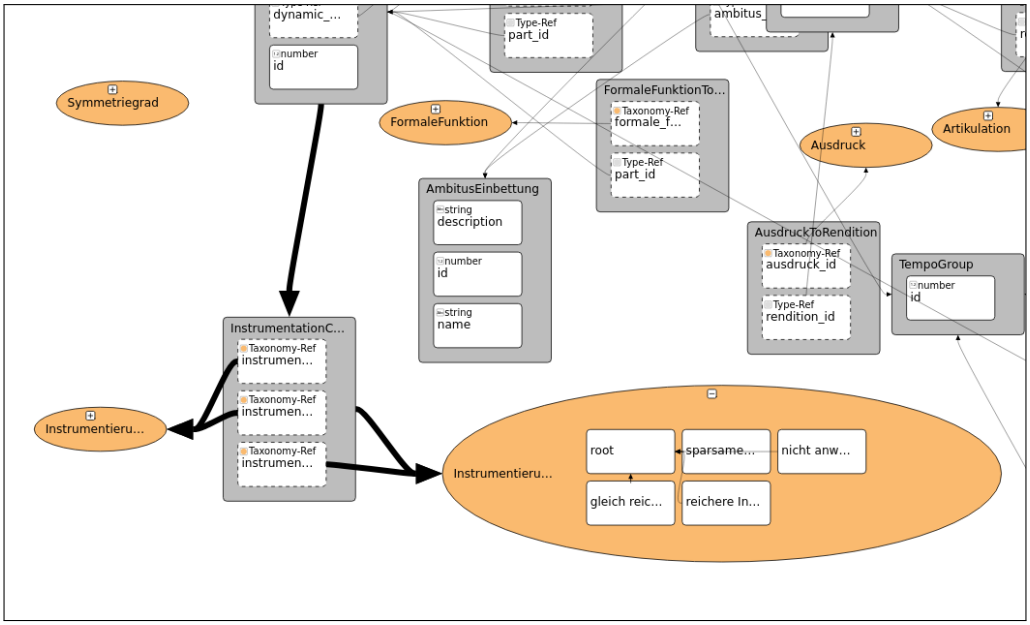


Abbildung 5.6.: Screenshot eines Typen mit einer Taxonomie. Typen sind als graue Rechtecke dargestellt. Taxonomien sind durch orange Ovale dargestellt. Eigenschaften eines Typen und Elemente der Taxonomie sind jeweils durch weiße Rechtecke gekennzeichnet.

5.2.7. Hierarchie

Die Hierarchie wird im Node-link Tree wie folgt dargestellt. Wie in Abbildung 5.6 zu sehen, werden hierarchische Relationen als Kindelement innerhalb der Elternelemente positioniert. Kindelemente welche eine Referenz auf einen anderen Typ oder eine Taxonomie darstellen, werden als unsichtbarer Link eingefügt, sodass die Rekursivität an dieser Stelle direkt unterbrochen wird. Diese verweisen mit einer Kante auf das Zielelement, welches auch als Elternelement dargestellt ist. Alle nicht hierarchischen Relationen werden ebenso als Kante eingefügt.

In der Indented List funktioniert die Darstellung genau nach dem gleichen Schema, nur dass hier ausschließlich hierarchische Kanten dargestellt werden. Kanten von Kindelementen, welche einen Verweis auf einen andern Typ oder eine Taxonomie darstellen, werden nicht abgebildet. Kindelemente werden in der Indented List unterhalb des Elternelementes als eingerücktes Element dargestellt. Diese sind auf- und zuklappbar.

Evaluation der finalen Anforderungen: Hierarchiedarstellung

FA-15 Im Node-link Tree sollen Kindelemente, angelehnt an die zoombare Visualisierung, innerhalb der Elternelemente positioniert sein.

Diese Anforderung ist vollständig erfüllt. Kindelemente sind innerhalb der Eltern angeordnet.

FA-16 In der Indented List sollen Kindelemente aufklappbar und eingerückt unter den Eltern aufgelistet sein.

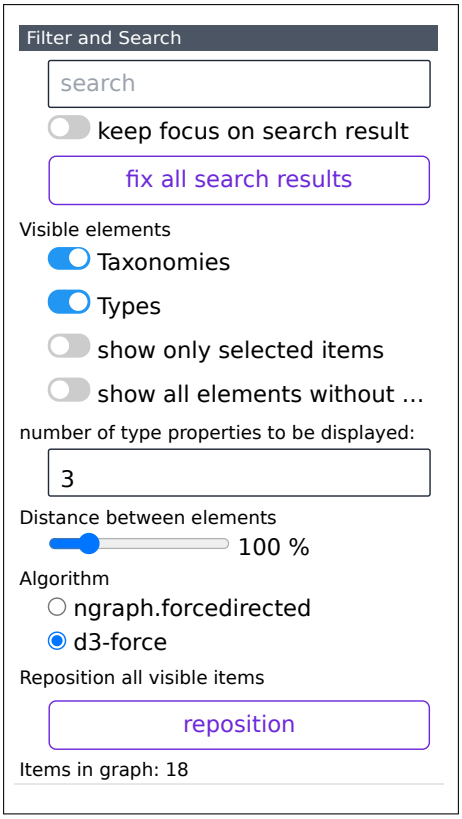


Abbildung 5.7.: Screenshot des Filter- und Suchbereiches

Diese Anforderung ist vollständig erfüllt. Elemente die Kindelemente enthalten sind auf- und zuklappbar.

5.2.8. Suche

Die Suche ist in Abbildung 5.7 ganz oben abgebildet. Diese Suche durchsucht die Namen der Elemente und deren Eigenschaften. In Abbildung 5.8 ist das Ergebnis der Suche nach dem Wort „Instrument“ zu sehen. Wie zu sehen sind die Ergebnisse im Node-link Tree hervorgehoben und alle anderen Elemente geschwächt dargestellt. Diese Hervorhebung verschwindet, sobald der Fokus nicht mehr auf dem Suchfeld liegt. Während der Implementierung hat sich als sinnvoll erwiesen, dass es auch gut ist, wenn der Fokus auf den gefundenen Elementen bleibt. Dies ist individuell unterhalb der Suche aktivierbar. In der Indented List ist zu sehen, wie alle Suchtreffer ebenso markiert werden und falls Kindelemente zu den Treffern gehören, werden die Elternelemente aufgeklappt.

Evaluation der finalen Anforderungen: Suchfunktion

FA-17 Die Suche soll die Namen aller Elemente durchsuchen.

Diese Anforderung ist vollständig erfüllt. Die Suche durchsucht die Namen aller Elemente, dabei wird nicht auf die Groß- und Kleinschreibung geachtet.

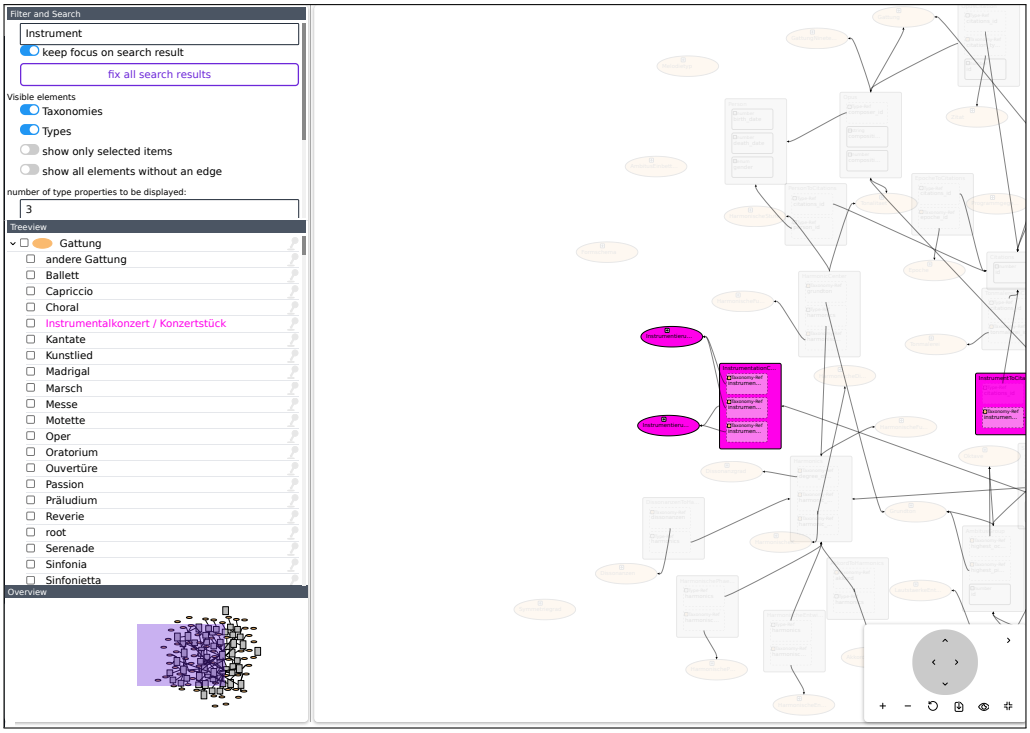


Abbildung 5.8.: Screenshot des Ergebnisses einer Suche nach dem Begriff „Instrument“. Sowohl in der Indented List, als auch im Node-link Tree sind die Ergebnisse markiert. Solange der Fokus auf dem Suchfeld liegt, sind auch im Node-link Tree die restlichen Elemente durchsichtig.

FA-18 Gefundene Elemente sollen hervorgehoben werden. Zur besseren Unterscheidung sollen nicht zutreffende Elemente ausgeblendet sein.

Diese Anforderung ist vollständig erfüllt. Elemente werden sowohl im Node-link Tree, als auch in der Indented List hervorgehoben. Alle anderen Ergebnisse werden durchsichtig dargestellt.

FA-19 In der Indented List sollen Elternelemente, falls ein Kindelement zu den Suchergebnissen gehört, aufgeklappt sein.

Diese Anforderung ist vollständig erfüllt. Falls ein Kindelement zu den Suchergebnissen gehört, wird das Elternelement aufgeklappt, sodass das Kindelement sichtbar wird.

5.2.9. Filtern

In Abbildung 5.7 ist neben der Suche auch der Filterbereich zu sehen. Hier ist es möglich die Taxonomien und die Typen im Node-link Tree sichtbar und unsichtbar zu stellen. Ebenso ist es möglich nur die ausgewählten Elemente im Node-link Tree darzustellen. Ein Element ist ausgewählt, sobald das dazugehörige Kontrollkästchen im Indented Tree angewählt ist.

Evaluation der finalen Anforderungen: Filterfunktion

FA-20 Der Benutzer soll die Möglichkeit haben im Node-link Tree nur bestimmte Elemente darzustellen.

Die Anforderung ist durch die nächsten zwei Anforderungen implementiert.

FA-21 Der Benutzer soll die Möglichkeit haben im Node-link Tree nur Elemente einer bestimmten Art, also Typen und Taxonomien, darzustellen.

Die Anforderung ist vollständig implementiert. Der Benutzer kann nur Elemente einer bestimmten Art darstellen.

FA-22 Der Benutzer soll die Möglichkeit haben ausschließlich selektierte Elemente der Ontologie darzustellen.

Die Anforderung ist vollständig implementiert. Der Benutzer kann Elemente selektieren und anschließend ausschließlich die selektierten Elemente darstellen.

5.2.10. Fokus & Kontext

Diese Funktion ist aufgrund der Nachteile nicht implementiert worden.

FA-23 Der Benutzer soll die Funktion Fokus & Kontext optional aktivieren können, damit Suchergebnisse in der Mitte des Graphen positioniert werden.

Die Anforderung ist nicht implementiert. Diese Funktion kann als eine optionale Funktion in der Zukunft implementiert werden.

5.2.11. Detailinformation

In Abbildung 5.9 ist die Detailansicht des Typ-Elementes „PersonToCitations“ dargestellt. Hier werden vorhandene Informationen dieses Elementes dargestellt. Alle weiteren Informationen eines Elementes sind über den Knopf „open item in new tab“ zu sehen. Hier gelangt der Benutzer auf die Elementseite von MUSE4Anything.

Evaluation der finalen Anforderungen: Detailinformationsansicht

FA-24 Für jedes Element soll es eine zusätzliche Detailinformationsansicht mit allen relevanten Information über dieses Element geben.

Die Anforderung ist vollständig implementiert. Es gibt eine Detailansicht, die alle relevanten Informationen über das jeweilige Element anzeigt.

FA-25 In dieser Ansicht soll es auch einen Link zur Einzelansicht dieses Elements von MUSE4Anything geben.

Die Anforderung ist vollständig implementiert. Der Benutzer kann die Einzelansicht des ausgewählten Elementes in einem neuen Tab direkt öffnen.

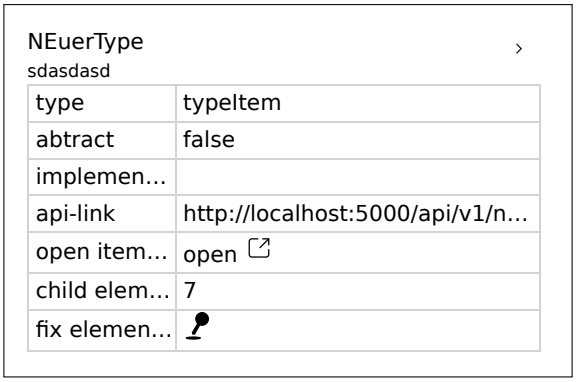


Abbildung 5.9.: Screenshot der Detailansicht des Elementes „PersonToCitations“. Hier werden alle vorhandene Informationen dargestellt. Es gibt die Möglichkeit das Element zu fixieren und direkt zur passenden Seite um das Element zu bearbeiten zu gelangen.

5.2.12. Verschieben & Zoomen

Der Node-link Tree lässt sich mittels der Maus verschieben, durch Doppelklick und das Mausrad vergrößern. Daneben gibt es noch das Interaktionsfeld im rechten unteren Bereich des Graphen. Hier lässt sich der Node-link Tree ebenso verschieben und vergrößern.

Evaluation der finalen Anforderungen: Verschieben & Zoomen

FA-26 Die Darstellung des Node-link Tree soll sich auf mehrere Arten zoomen lassen: Doppelklick, Mausrad und Interaktionstaster.

Die Anforderung ist vollständig implementiert. Der Node-link Tree lässt sich mit allen drei Methoden vergrößern und verkleinern.

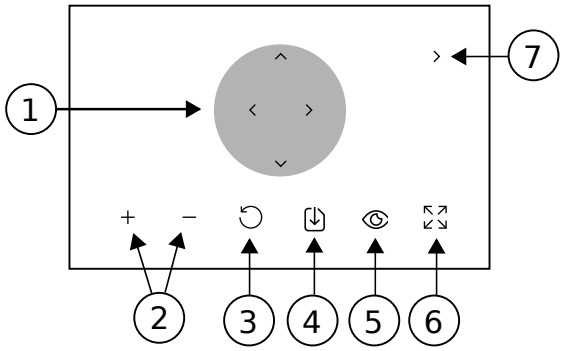


Abbildung 5.10.: Screenshot des Interaktionsfeldes. Mit (1) lässt sich der Node-link Tree verschieben. Zoomen im Node-link Tree funktioniert (2). Mit (3) lässt sich der Graph wieder auf die Ausgangsposition zurücksetzen. Mit (4) wird der Graph downloadbar sein. Mit (5) lässt sich ein temporärer Zoom-Out des Node-link Tree darstellen. Mit (6) kann der Graph maximiert oder minimiert werden. Mit (7) kann das Interaktionsfeld verborgen werden

FA-27 Der Node-link Tree und darin enthaltene Elemente soll sich auf zwei Möglichkeiten verschieben lassen: mit der Maus und durch Interaktionstaster.

Die Anforderung ist vollständig implementiert. Der Node-link Tree und die darin enthaltenen Elemente lassen sich mittels der Maus verschieben. Zusätzlich lässt sich der Node-link Tree noch mit den Interaktionstastern verschieben.

5.2.13. Parameter für das automatische Layout

In Abbildung 5.7 sind im unteren Bereich Möglichkeiten zur Parametereinstellung gegeben. Der Benutzer kann neben der Anzahl standardmäßig dargestellter Eigenschaften eines Typen den Abstand der Elemente zur Positionierung beeinflussen. Die Möglichkeit an Einstellungen kann in den weiterführenden Versionen erweitert werden, zum Beispiel könnten verschiedene Anordnungsalgorithmen eingeführt werden und der Benutzer kann den gewünschten auswählen.

Evaluation der finale Anforderungen: Parametereinstellungen

FA-28 Der Benutzer soll die maximale Anzahl standardmäßig dargestellter Eigenschaften eines Typen definieren können.

Die Anforderung ist vollständig implementiert. Der Benutzer kann die Anzahl der angezeigten Eigenschaften eines Typen definieren. Standardmäßig sind dies drei Eigenschaften.

FA-29 Der Benutzer soll den Abstand der Elemente beeinflussen können.

Die Anforderung ist vollständig implementiert. Der Benutzer kann in der Parametereinstellung den Abstand durch einen prozentualen Wert beeinflussen. Standardmäßig ist die Darstellung bei 100%.

6. Zusammenfassung und Ausblick

Der erste Teil der Arbeit gibt einen Einblick in die Hintergründe und Grundlagen der Arbeit. Insbesondere wird auf das Datenbankschema von MUSE4Anything, den Aufbau der Ontologien und Taxonomien eingegangen. Die Ontologiedefinitionen können teilweise zu sehr komplexen Datenstrukturen führen, da unter anderem die Typen sich rekursiv beinhalten können.

Im zweiten Teil geht es um die Anforderung einer solchen Visualisierung. Hierbei wurden zuerst auf Grundlage von wissenschaftlichen Arbeiten verschiedenste Varianten der Visualisierung erläutert. Eine Möglichkeit ist der Node-link Tree. Eine weitere ist die Indented List. Die Kombination dieser beiden Visualisierungen führt dazu, dass Aufgaben von Benutzern am erfolgreichsten gelöst werden. Die möglichen Gestaltungsarten der Elemente in einem Node-link Tree wurden auch untersucht und aufgelistet. Ebenso wurden verschiedenste Interaktionsfunktionalitäten erläutert und verglichen. Anhand dieser Möglichkeiten und mehreren wissenschaftlichen Untersuchungen wurden Features für eine mögliche Visualisierung aufgestellt. Anschließend wurden mehrere reale Anwendungen untersucht, ob diese die erarbeiteten Features implementiert haben. Auf Basis der Häufigkeit der verwendeten Features und den Untersuchungen wurden die finalen Anforderungen an die Visualisierung für MUSE4Anything definiert. Im letzten Schritt wurden die finalen Anforderungen in MUSE4Anything implementiert.

Die erarbeiteten finalen Anforderungen aus Abschnitt 4.4 sind erfolgreich implementiert worden. Die Ontologien in MUSE4Anything lassen sich nun mit einem Klick graphisch darstellen. Somit lässt sich damit besser arbeiten und Relationen werden auf einen Blick sichtbar.

Nächste Schritte

Eine sinnvolle Verbesserung ist die Optimierung der verwendeten Algorithmen für die kraftgesteuerte Anordnung. Der „ngraph.forcedirected“ Algorithmus ist einfach und schnell zu verwenden, aber in der Flexibilität ist dieser beschränkt. Die Einstellungsparameter können auch noch optimiert werden, was vermutlich zu einer besseren Anordnung führt. Als zweiter Algorithmus wurde der „d3-force“ Algorithmus implementiert. Dieser ist sehr mächtig und hat viele Einflussfaktoren. In der aktuellen Version liefert dieser schon ein sehr gutes Ergebnis. Dieses lässt sich aber aufgrund der vielen Faktoren und Einflüsse, die in diesem Algorithmus definierbar sind, noch verbessern. So kann unter anderem darauf geachten werden, dass sich Objekte in der Visualisierung nicht überlagern. Die dargestellten Objekte sind unterschiedlich groß, daher kann es sein, dass diese sich überlagern. Neben dem Algorithmus, kann auch die Verwendung anderer Layoutvarianten, wie beispielsweise der Baumanordnung, eine weitere Möglichkeit sein, dem Benutzer die passendste Visualisierung zu bieten.

Bei dem Thema der Visualisierung kann auch die Verwendung einer weiteren Dimension untersucht werden. Hier gibt es verschiedene Möglichkeiten, entweder wird die weitere Dimension nur auf eine Linse angewendet oder auf die gesamte Visualisierung. Die Anwendung der weiteren Dimension auf eine Linse, entspricht der Fisheye-Linse. Hier ist es dann möglich in einem ausgewählten Bereich, der Linse, aufgrund der weiteren Dimension mehr Informationen darzustellen beziehungsweise den Fokus auf gewünschte Informationen lenken.

Neben der möglichen Verwendung einer Linse mit weiterer Dimension können auch semantische Linsen einen Vorteil bringen. Diese wurden in Abschnitt 3.2.5 erklärt. Mit diesen Linsen lassen sich im Linsenbereich bestimmte Informationen genauer darstellen oder andere Informationen ausblenden.

Bei der Suche nach Elementen aus der Ontologie gibt es auch noch Optimierungsbedarf. Diese kann mithilfe von Indexierung noch beschleunigt werden. Auf der anderen Seite könnte diese dem Benutzer automatisch Vorschläge zu dem bisher eingegebenen Suchbegriff bieten.

Eine weitere hilfreiche Funktion ist der Export der Visualisierung. Diese Funktion ist schon ansatzweise implementiert. Es ist eine hilfreiche Funktion, wenn der Benutzer die Möglichkeit hat, sich die aktuelle Visualisierung in verschiedenen Formaten (PDF, PNG, SVG) herunterzuladen.

Neben dem Herunterladen der aktuellen Visualisierung ist noch das Speichern der aktuellen Visualisierung eine sinnvolle Funktion. Hiermit kann der Benutzer seine aktuelle Visualisierung abspeichern. Dies ist von Vorteil, da der Benutzer seine Visualisierung individuell anpassen kann. Somit können die Anpassungen wieder angewendet werden.

In Abschnitt 3.2.4 ist die Fokus & Kontext Methode vorgestellt. Diese Methode ist aufgrund ihrer Nachteile nicht in der ersten Version implementiert. Es ist möglich diese als eine individuell aktivierbare Funktion zu implementieren. So kann der Benutzer frei wählen, ob er die Methode aktiviert haben möchte oder nicht.

Literaturverzeichnis

[AAB] N. Achich, A. Algergawy, B. Bouaziz. *BioOntoVis: An Ontology Visualization Tool*. en. URL: https://project.inria.fr/ekaw2018/files/2018/11/ekaw-demo-16.pdf (besucht am 14. 07. 2021) (zitiert auf S. 31).

[AAI+77] C. Alexander, P. i. t. D. o. A. C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, A. Shlomo. *A Pattern Language: Towns, Buildings, Construction*. en. OUP USA, 1977. ISBN: 978-0-19-501919-3 (zitiert auf S. 9).

[Ala03] H. Alani. *TGVizTab: An Ontology Visualisation Extension for Protégé*. Jan. 2003. URL: https://eprints.soton.ac.uk/258326/ (zitiert auf S. 26, 27, 42).

[ALK+17] A. Anikin, D. Litovkin, M. Kultsova, E. Sarkisova, T. Petrova. *Ontology Visualization: Approaches and Software Tools for Visual Representation of Large Ontologies in Learning*. Pages: 149. Aug. 2017. ISBN: 978-3-319-65550-5. DOI: 10.1007/978-3-319-65551-2_10 (zitiert auf S. 23).

[AWL] M. R. A. Asmat, V. Wiens, S. Lohmann. *A Comparative User Evaluation on Visual Ontology Modeling Using Node-Link Diagrams*. en. Techn. Ber. DOI: 10.3233/978-1-61499-894-5-1 (zitiert auf S. 15, 26).

[Bar] J. Barzen. *Taxonomien kostümrelevanter Parameter: Annäherung an eine Ontologisierung der Domäne des Filmkostüms*. de. Techn. Ber., S. 60 (zitiert auf S. 7).

[Bar18] J. Barzen. „Wenn Kostüme sprechen – Musterforschung in den Digital Humanities am Beispiel vestimentärer Kommunikation im Film“. de. text.thesis.doctoral. Universität zu Köln, 2018. URL: https://kups.ub.uni-koeln.de/9134/ (besucht am 11. 05. 2021) (zitiert auf S. 7, 9).

[BBČ+10] J. Bārzdiņš, G. Bārzdiņš, K. Čerāns, R. Liepiņš, A. Sprogis. „UML Style Graphical Notation and Editor for OWL 2“. en. In: *Perspectives in Business Informatics Research*. Hrsg. von W. van der Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, P. Forbrig, H. Günther. Bd. 64. Series Title: Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 102–114. ISBN: 978-3-642-16100-1. DOI: 10.1007/978-3-642-16101-8_9 (zitiert auf S. 36, 37).

[BBE+17] J. Barzen, U. Breitenbücher, L. Eusterbrock, M. Falkenthal, F. Hentschel, F. Leymann. *The vision for MUSE4Music: Applying the MUSE method in musicology*. en. Techn. Ber. 3-4. Juli 2017, S. 323–328. DOI: 10.1007/s00450-016-0336-1 (zitiert auf S. 7, 9).

[BDM15] L. Balzer, M. T. Do, D. Maseluk. „Comparison and Evaluation of Ontology Visualizations“. Diss. Jan. 2015. URL: http://dx.doi.org/10.18419/opus-3499 (besucht am 05. 07. 2021) (zitiert auf S. 31, 37, 39, 41, 43).

[BJKK10] T. Boiński, A. Jaworska, R. Kleczkowski, P. Kunowski. *Ontology visualization*. Pages: 20. Juli 2010. ISBN: 978-1-4244-8182-8. (Besucht am 28. 05. 2021) (zitiert auf S. 19).

[BR] T. D. Breaux, J. W. Reed. *Hierarchical Information Clustering Using Ontology Languages*. en. Techn. Ber., S. 7. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.3645&rep=rep1&type=pdf (zitiert auf S. 22).

[Büh21] F. Bühler. *MUSE4Anything*. en. Techn. Ber. Accepted: 2021-04-20T10:24:28Z. 2021. DOI: http://dx.doi.org/10.18419/opus-11410 (zitiert auf S. 7, 9, 11, 22).

[CHI21] T. CHISEL Group. *Jambalaya*. 2021. URL: https://thechiselgroup.org/jambalaya/ (besucht am 15. 07. 2021) (zitiert auf S. 38).

[CSM09] N. Catenazzi, L. Sommaruga, R. Mazza. „User-Friendly Ontology Editing and Visualization Tools: The OWLeasyViz Approach“. In: *2009 13th International Conference Information Visualisation*. ISSN: 2375-0138. Juli 2009, S. 283–288. DOI: 10.1109/IV.2009.34 (zitiert auf S. 18, 19, 23, 27, 28).

[d3-21] d3-force. *d3-force*. original-date: 2015-11-03T20:35:53Z. Nov. 2021. URL: https://github.com/d3/d3-force (besucht am 20. 11. 2021) (zitiert auf S. 58).

[DCL+17] F. Du, N. Cao, Y.-R. Lin, P. Xu, H. Tong. „iSphere: Focus+Context Sphere Visualization for Interactive Large Graph Exploration“. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. New York, NY, USA: Association for Computing Machinery, Mai 2017, S. 2916–2927. DOI: 10.1145/3025453.3025628 (zitiert auf S. 24).

[DLSP18] M. Dudáš, S. Lohmann, V. Svátek, D. Pavlov. *Ontology visualization methods and tools: a survey of the state of the art*. en. Techn. Ber. Publisher: Cambridge University Press. 2018. DOI: 10.1017/S0269888918000073 (zitiert auf S. 14–18, 23, 24, 29, 34, 35, 37, 39–42, 44, 49).

[ERG02] P. Eklund, N. Roberts, S. Green. „OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser“. In: Feb. 2002, S. 405–411. DOI: 10.1109/CW.2002.1180907 (zitiert auf S. 19).

[Fal10] S. Falconer. *OntoGraf*. 2010. URL: https://protegewiki.stanford.edu/wiki/OntoGraf (besucht am 15. 07. 2021) (zitiert auf S. 38).

[FL20] M. Florrence, R. Lourdusamy. *Feature analysis of ontology visualization methods and tools*. Techn. Ber. Juli 2020, S. 61–77. DOI: 10.11591/csit.v1i2.p61-77 (zitiert auf S. 34, 37, 43).

[FNS13] B. Fu, N. F. Noy, M.-A. Storey. „Indented Tree or Graph? A Usability Study of Ontology Visualization Techniques in the Context of Class Mapping Evaluation“. en. In: *Advanced Information Systems Engineering*. Hrsg. von D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, C. Salinesi, M. C. Norrie, Ó. Pastor. Bd. 7908. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 117–134. ISBN: 978-3-642-38708-1. DOI: 10.1007/978-3-642-41335-3_8 (zitiert auf S. 14, 18, 26–28).

[FNS16] B. Fu, N. F. Noy, M.-A. Storey. *Eye tracking the user experience – An evaluation of ontology visualization techniques*. en. Techn. Ber. 1. Nov. 2016, S. 23–41. DOI: 10.3233/SW-140163 (zitiert auf S. 14, 26).

[Fur14] F. J. Furrer. *Eine kurze Geschichte der Ontologie: Von der Philosophie zur modernen Informatik*. de. Techn. Ber. 4. Aug. 2014, S. 308–317. DOI: 10.1007/s00287-012-0642-3 (zitiert auf S. 10).

[GC10] S. S. Guo, C. W. Chan. „A tool for ontology visualizaiton in 3D graphics: Onto3DViz“. In: *CCECE 2010*. ISSN: 0840-7789. Mai 2010, S. 1–4. DOI: 10.1109/CCECE.2010.5575219 (zitiert auf S. 35).

[GEM+16] F. Ghorbel, N. Ellouze, E. Métais, F. Hamdi, F. Gargouri, N. Herradi. *MEMO GRAPH: An Ontology Visualization Tool for Everyone*. Techn. Ber. 2016, S. 265–274. DOI: 10.1016/j.procs.2016.08.139. URL: https://doi.org/10.1016/j.procs.2016.08.139 (zitiert auf S. 32, 33).

[GJZZ13] Q. Guo, W. Ji, S. Zhong, E. Zhou. „The Analysis of the Ontology-based K-Means Clustering Algorithm“. en. In: *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. China: Atlantis Press, 2013. DOI: 10.2991/iccsee.2013.186 (zitiert auf S. 22).

[HHDK00] M. C. Hao, M. Hsu, U. Dayal, A. Krug. „Web-Based Visualization of Large Hierarchical Graphs Using Invisible Links in a Hyperbolic Space“. en. In: *Advances in Visual Information Management*. Hrsg. von H. Arisawa, T. Catarci. Boston, MA: Springer US, 2000, S. 83–94. ISBN: 978-1-4757-4457-6. DOI: 10.1007/978-0-387-35504-7_6 (zitiert auf S. 20, 28).

[HKRS08] P. Hitzler, M. Krötzsch, S. Rudolph, Y. Sure. *Semantic Web*. de. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-33993-9. DOI: 10.1007/978-3-540-33994-6 (zitiert auf S. 10).

[HMM00] I. Herman, G. Melancon, M. Marshall. *Graph visualization and navigation in information visualization: A survey*. Techn. Ber. 1. Conference Name: IEEE Transactions on Visualization and Computer Graphics. Jan. 2000, S. 24–43. DOI: 10.1109/2945.841119 (zitiert auf S. 18, 23, 28, 29).

[Hor10] M. Horridge. *OWLViz*. 2010. URL: https://protegewiki.stanford.edu/wiki/OWLViz (besucht am 15. 07. 2021) (zitiert auf S. 40).

[Ist10] I. Istochnick. *OWL2UML*. 2010. URL: https://protegewiki.stanford.edu/wiki/OWL2UML (besucht am 15. 07. 2021) (zitiert auf S. 40).

[Kas21a] A. Kashcha. *ngraph*. original-date: 2014-01-02T05:27:25Z. Nov. 2021. URL: https://github.com/anvaka/ngraph (besucht am 20. 11. 2021) (zitiert auf S. 58).

[Kas21b] A. Kashcha. *ngraph.forcelayout*. original-date: 2013-11-15T04:49:55Z. Nov. 2021. URL: https://github.com/anvaka/ngraph.forcelayout (besucht am 20. 11. 2021) (zitiert auf S. 58).

[KHL+07] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, E. Giannopoulou. *Ontology visualization methods—a survey*. en. Techn. Ber. 4. Nov. 2007, S. 10. DOI: 10.1145/1287620.1287621 (zitiert auf S. 7, 18).

[Kob12] S. Kobourov. *Spring Embedders and Force Directed Graph Drawing Algorithms*. Techn. Ber. Jan. 2012. URL: https://arxiv.org/abs/1201.3011 (besucht am 21. 11. 2021) (zitiert auf S. 15).

[Kri09] S. Kriglstein. „User Requirements Analysis on Ontology Visualization“. In: *2009 International Conference on Complex, Intelligent and Software Intensive Systems*. März 2009, S. 694–699. DOI: 10.1109/CISIS.2009.37 (zitiert auf S. 23, 26, 28).

[KTH+06] A. Katifori, E. Torou, C. Halatsis, G. Lepouras, C. Vassilakis. *A Comparative Study of Four Ontology Visualization Techniques in Protege: Experiment Setup and Preliminary Results*. Pages: 423. Aug. 2006. ISBN: 978-0-7695-2602-7. DOI: 10.1109/IV.2006.3 (zitiert auf S. 14, 19, 26, 27).

[Kun12] P. Kunowski. *SOVA*. 2012. URL: https://protegewiki.stanford.edu/wiki/SOVA (besucht am 20. 07. 2021) (zitiert auf S. 41).

[KVWW07] S. Krivov, F. Villa, R. Williams, X. Wu. „On Visualization of OWL Ontologies“. In: *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*. Journal Abbreviation: Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences. Jan. 2007, S. 205–221. ISBN: 978-0-387-48436-5. DOI: 10.1007/978-0-387-48438-9_11 (zitiert auf S. 18, 27).

[KW10] S. Kriglstein, G. Wallner. „Knoocks - A Visualization Approach for OWL Lite Ontologies“. en. In: *2010 International Conference on Complex, Intelligent and Software Intensive Systems*. Krakow, TBD, Poland: IEEE, Feb. 2010, S. 950–955. DOI: 10.1109/CISIS.2010.55 (zitiert auf S. 19, 20, 23, 26–28, 32, 49).

[Lan17] C. Lanum. *Visualizing graph data*. OCLC: ocn948562767. New York: Manning, 2017. ISBN: 978-1-61729-307-8 (zitiert auf S. 26).

[LNB14] S. Lohmann, S. Negru, D. Bold. „The ProtégéVOWL Plugin: Ontology Visualization for Everyone“. en. In: *The Semantic Web: ESWC 2014 Satellite Events*. Hrsg. von V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, A. Tordai. Bd. 8798. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, S. 395–400. ISBN: 978-3-319-11954-0. DOI: 10.1007/978-3-319-11955-7_55 (zitiert auf S. 18, 27, 48).

[LNHE14] S. Lohmann, S. Negru, F. Haag, T. Ertl. „VOWL 2: User-Oriented Visualization of Ontologies“. en. In: *Knowledge Engineering and Knowledge Management*. Hrsg. von K. Janowicz, S. Schlobach, P. Lambrix, E. Hyvönen. Bd. 8876. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, S. 266–281. ISBN: 978-3-319-13703-2. DOI: 10.1007/978-3-319-13704-9_21 (zitiert auf S. 18, 19, 26, 28, 43).

[LNHE16] S. Lohmann, S. Negru, F. Haag, T. Ertl. *Visualizing ontologies with VOWL*. en. Techn. Ber. 4. Mai 2016, S. 399–419. DOI: 10.3233/SW-150200 (zitiert auf S. 19, 23, 26, 28).

[MMP+11] E. Motta, P. Mulholland, S. Peroni, M. d'Aquin, J. M. Gomez-Perez, V. Mendez, F. Zablith. „A Novel Approach to Visualizing and Navigating Ontologies“. en. In: *The Semantic Web – ISWC 2011*. Hrsg. von L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, E. Blomqvist. Bd. 7031. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-25073-6_30 (zitiert auf S. 34).

[Moo09] D. Moody. *The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering*. en. Techn. Ber. 6. Nov. 2009, S. 756–779. DOI: 10.1109/TSE.2009.67 (zitiert auf S. 18, 19, 26–28, 48).

[MPE+15] D. Mouromtsev, D. Pavlov, Y. Emelyanov, A. Morozov, D. Razdyakonov, M. Galkin. *The simple, Web-based tool for visualization and sharing of semantic data and ontologies*. Okt. 2015. URL: https://www.researchgate.net/publication/283079862 (besucht am 21. 11. 2021) (zitiert auf S. 35, 36).

[NM08] Q. Nguyen, H. Mao-Lin. *Large Graph Visualization by Hierarchical Clustering*. Techn. Ber. Okt. 2008. DOI: 10.3724/SP.J.1001.2008.01933 (zitiert auf S. 26).

[Onta] Ontosphere. *OntoSphere 3D - Screenshots*. URL: http://ontosphere3d.sourceforge.net/screenshots.html (besucht am 14. 07. 2021) (zitiert auf S. 39).

[Ontb] Ontosphere. *OntoSphere3D, more than a 3D ontology visualization tool*. URL: http://ontosphere3d.sourceforge.net/ (besucht am 15. 07. 2021) (zitiert auf S. 39).

[OWL21] OWLGrEd. *OWLGrEd*. 2021. URL: http://owlgred.lumii.lv/online_visualization/koala.owl (besucht am 14. 07. 2021) (zitiert auf S. 36).

[Pil12] U. Pillkahn. *Innovationen zwischen Planung und Zufall: Bausteine einer Theorie der bewussten Irritation*. de. BoD – Books on Demand, März 2012. ISBN: 978-3-8448-4601-0 (zitiert auf S. 22).

[Pro] Protégé. *Protege Wiki*. URL: https://protegewiki.stanford.edu/wiki/Main_Page (besucht am 15. 07. 2021) (zitiert auf S. 37, 39).

[RV14] S. Ramakrishnan, A. Vijayan. *A study on development of cognitive support features in recent ontology visualization tools*. en. Techn. Ber. 4. Apr. 2014, S. 595–623. DOI: 10.1007/s10462-012-9326-2 (zitiert auf S. 13–15, 17, 26, 27).

[SA11] R. Sivakumar, P. V. Arivoli. *Ontology Visualization Protégé Tools – a Review*. en. SSRN Scholarly Paper ID 3429010. Rochester, NY: Social Science Research Network, Aug. 2011. URL: https://papers.ssrn.com/abstract=3429010 (besucht am 10. 06. 2021) (zitiert auf S. 13, 17, 24).

[SBN+10] C. Stab, M. Breyer, K. Nazemi, D. Burkhardt, C. Hofmann, Dieter Fellner. „SemaSun: Visualization of semantic knowledge based on an improved sunburst visualization metaphor". In: *Proceedings of edMedia + innovate learning 2010*. Hrsg. von J. Herrington, Craig Montgomerie. Toronto, Canada: Association for the Advancement of Computing in Education (AACE), Juni 2010, S. 911–919. URL: https://www.learntechlib.org/p/34743 (besucht am 21. 11. 2021) (zitiert auf S. 13–15, 17, 18, 24, 26, 27).

[Sch] C. Schmidt. *OWL Editor*. en. Techn. Ber., S. 79. URL: http://neon-toolkit.org/w/images/Doku_2.3.pdf (besucht am 20. 07. 2021) (zitiert auf S. 33).

[SF11] I. Silva, C. Freitas. „Requirements for Interactive Ontology Visualization - Using Hypertree +2.5D Visualization for Exploring Relationships between Concepts." In: Jan. 2011, S. 242–248. URL: https://www.researchgate.net/publication/221116212 (besucht am 21. 11. 2021) (zitiert auf S. 18, 26–28).

[SS08] A. Schlicht, H. Stuckenschmidt. „A Flexible Partitioning Tool for Large Ontologies“. In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Bd. 1. Dez. 2008, S. 482–488. DOI: 10.1109/WIIAT.2008.398 (zitiert auf S. 22).

[Sta] Startpage. *Startpage - Private Search Engine. No Tracking. No Search History.* en. URL: https://www.startpage.com (besucht am 14. 07. 2021) (zitiert auf S. 30).

[Stu09] H. Stuckenschmidt. *Ontologien*. de. Techn. Ber. 2009. DOI: 10.1007/978-3-540-79333-5 (zitiert auf S. 10).

[TAHS06] C. Tominski, J. Abello, F. van Ham, H. Schumann. „Fisheye Tree Views and Lenses for Graph Visualization“. In: *Tenth International Conference on Information Visualisation (IV'06)*. ISSN: 2375-0138. Juli 2006, S. 17–24. DOI: 10.1109/IV.2006.54 (zitiert auf S. 24, 25, 29).

[TAS09] C. Tominski, J. Abello, H. Schumann. *CGV—An interactive graph visualization system*. en. Techn. Ber. 6. Dez. 2009, S. 660–678. DOI: 10.1016/j.cag.2009.06.002 (zitiert auf S. 26).

[TK18] A. Tiwari, A. Kumar. „Comparative Analysis of Optimized Algorithms for Ontology Clustering“. In: *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. Nov. 2018, S. 1–7. DOI: 10.1109/UPCON.2018.8597150 (zitiert auf S. 22).

[TWC11] TWCarlson. *British Isles Euler diagram*. 2011. URL: https://commons.wikimedia.org/wiki/File:British_Isles_Euler_diagram_15.svg (besucht am 15. 11. 2021) (zitiert auf S. 17).

[VOW] VOWL. *VOWL: Visual Notation for OWL Ontologies*. URL: http://vowl.visualdataweb.org/v2/ (besucht am 14. 07. 2021) (zitiert auf S. 41).

[Web21a] WebVOWL. *WebVOWL*. 2021. URL: http://www.visualdataweb.de/webvowl/ (besucht am 15. 07. 2021) (zitiert auf S. 42).

[Web21b] WebVOWL. *WebVOWL*. 2021. URL: https://github.com/VisualDataWeb/WebVOWL (besucht am 20. 11. 2021) (zitiert auf S. 43).

[web21] webvowl. *WebVOWL*. 2021. URL: http://vowl.visualdataweb.org/webvowl.html (besucht am 20. 11. 2021) (zitiert auf S. 43).

[Wil99] G. J. Wills. *NicheWorks—Interactive Visualization of Very Large Graphs*. en. Techn. Ber. 2. Juni 1999, S. 190–212. DOI: 10.1080/10618600.1999.10474810 (zitiert auf S. 23).

[WLA17] V. Wiens, S. Lohmann, S. Auer. *Semantic Zooming for Ontology Graph Visualizations*. Pages: 8. Dez. 2017. DOI: 10.1145/3148011.3148015 (zitiert auf S. 29).

[yFi21] yFiles. *yFiles*. 2021. URL: https://www.yworks.com/yfiles-overview (besucht am 20. 11. 2021) (zitiert auf S. 43).

[yWo] yWorks. *Visualizing an Ontology*. en. URL: https://www.yworks.com (besucht am 14. 07. 2021) (zitiert auf S. 43).

[yWo21] yWorks. *Visualizing an Ontology*. 2021. URL: https://www.yworks.com/use-case/visualizing-an-ontology (besucht am 14. 07. 2021) (zitiert auf S. 43).

A. Suchergebnisse: ontology visualization tools

Die Suche wurde am 07.07.2021 mit startpage.com durchgeführt.

1. http://vowl.visualdataweb.org/webvowl.html
2. https://stackoverflow.com/questions/9409187/the-best-tool-for-visualizing-ontologies
3. https://www.researchgate.net/publication/326578306_Ontology_visualization_methods_and_tools_a_survey_of_the_state_of_the_art
4. http://ceur-ws.org/Vol-2262/ekaw-demo-16.pdf
5. http://disi.unitn.it/~p2p/RelatedWork/Matching/a10-katifori.pdf
6. https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/ontology-visualization-methods-and-tools-a-survey-of-the-state-of-the-art/5EA8C64D7DF60A84F6D2B7B9A09B6E6A
7. https://www.yworks.com/use-case/visualizing-an-ontology
8. https://protegewiki.stanford.edu/wiki/Visualization
9. https://cse.buffalo.edu/sneps/Bibliography/iwood15.pdf
10. https://www.sciencedirect.com/science/article/pii/S1877050916319408
11. http://protege-project.136.n4.nabble.com/Ontology-visualisation-tools-td4671197.html
12. https://link.springer.com/chapter/10.1007/978-3-319-65551-2_10
13. https://link.springer.com/chapter/10.1007/978-3-030-29551-6_4
14. http://scg.unibe.ch/research/vison
15. https://www.semanticscholar.org/paper/Ontology-Visualization-Tools-to-Assist-in-Creating-Schlegel-Elkin/3ffb60e33e8009761b4465dee57c7ef86c4eda22
16. https://www.fruct.org/publications/fruct11/files/Smi.pdf
17. http://geneontology.org/docs/tools-overview/
18. https://www.youtube.com/watch?v=bpjMYBc98bk
19. http://owlgred.lumii.lv/online_visualization
20. http://ontosphere3d.sourceforge.net/

Erklärung

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

Ort, Datum, Unterschrift