ORGANISATION VON TEILGRAPHEN SEMANTISCHER NETZE ZUR ONTOLOGIEGESTÜTZTEN WISSENSVERMITTLUNG

Der Fakultät für Mathematik und Informatik
der Universität Leipzig eingereichte

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science
(B. Sc.)

im Fachgebiet Informatik vorgelegt von

Thomas Pause

geboren am 08.01.1986 in Greifswald, Deutschland

Leipzig, 19. Oktober 2020

AUTOR:
Thomas Pause

TITEL:
Organisation von Teilgraphen semantischer Netze zur ontologiegestützten Wissensvermittlung

INSTITUT/FAKULTÄT:
Institut für Medizinische Informatik, Statistik und Epidemiologie
Fakultät für Mathematik und Informatik
Universität Leipzig

BETREUER:
Prof. Dr. Alfred Winter
Konrad Höffner

ABSTRAKT

Die Lehre der medizinischen Informatik ist durch viele Begriffe und Definitionen geprägt.

Um diese besser (be-)greifbar zu machen, gibt es eine an der Universität Leipzig entwickelte Visualisierungsanwendung, die aus Lehrbüchern extrahierte semantische Daten als Graph aufbereitet zur Verfügung stellt. Das System eignet sich gut als E-Learning-Werkzeug sowohl für Lehrende als auch für Studierende.

In dieser Arbeit wird der Weg, Teilgraphen zu erstellen und zu organisieren, verbessert. Wichtige Ziele sind die möglichst intuitive Arbeit an mehreren Teilgraphen sowie die Unterstützung der Lehrenden bei der Nutzung des Systems.

Da die Arbeit eher praktisch ausgerichtet ist, richtet sich die Methodik nach dem Ablauf eines (Softwareentwicklungs-) Projektes.

Zu Beginn erfolgt eine ausführliche Einführung in die Thematik und die wichtigen Technologien. Die Recherche zum Stand der Forschung ergibt, dass zur Zielerreichung eine individuelle Lösung benötigt wird. Nach der Analyse und Bewertung der vorhandenen Visualisierung folgt die Systemauswahl nach dafür erhobenen Anforderungen. Die Systemeinführung geht mit dem Erstellen einer Anleitung und der Modellierung des Basisprozesses einher. Zusätzlich werden Beispielszenarien entwickelt und vorgestellt, die direkt in einer Veranstaltung im Bachelorstudium der medizinischen Informatik an der Universität Leipzig eingesetzt werden können.

The structure is everything.
There are billions of neurons in our brains,
but what are neurons? Just cells.
The brain has no knowledge until connections
are made between neurons.
All that we know, all that we are,
comes from the way our neurons are connected.
— *Tim Berners-Lee* —

DANKSAGUNG

Auch für mich ist Struktur alles. Ohne Struktur wäre ich wohl niemals zu der Gelegenheit gekommen, diese Arbeit zu schreiben. Ich möchte diese Seite nutzen, mich bei den Leuten zu bedanken, die großen Anteil an der Entstehung dieser Arbeit und dem erfolgreichen Abschluss meines Studiums haben.

Ich danke Sarah Haase, der Liebe meines Lebens, die mir die Struktur in meinem Leben gezeigt und mit mir gemeinsam erkämpft hat. Für die vielen Jahre und die stetige Unterstützung bin ich unendlich dankbar. Du gibst mir die Kraft, das Selbstvertrauen und eben die Struktur, die so unglaublich wichtig ist, um erfolgreich zu sein.

Während des Studiums durfte ich viele nette Menschen kennen lernen, die ich hier unmöglich alle erwähnen kann. Dennoch möchte ich Bärbel Hanle hervorheben, die es geschafft hat, mir die Welt der Mathematik zu öffnen. Ohne dich wäre ich bereits im Grundstudium kläglich gescheitert.

Desweiteren Lukas Hempel, den ich erst spät kennengelernt, aber dafür immer sehr gern mit ihm zusammengearbeitet habe. Selbst wenn ich mal ein wenig down war, konntest du mich immer wieder mit deiner unfassbaren Motivation und guten Laune zurück auf den fokussierten Weg bringen.

Ein großes Dankeschön geht an Immanuel Bekaan, meinen treuen Freund, Sportkamerad und Wegbegleiter. Mit dir war es immer toll, auch nachdem wir keine Kommilitonen mehr waren. Ich genieße jede freie Minute und jeden Kilometer auf dem Fahrrad sehr mit dir und freue mich, dass wir nun Arbeitskollegen sind und uns oft sehen können.

Weiterhin bedanke ich mich bei Konrad Höffner, dem Betreuer dieser Arbeit, meinem Mentor in den Jahren am IMISE, Freund und Berater in allen Belangen. Du bist ein wichtiger Teil für den Erfolg dieser Arbeit und meines Studiums geworden und hast mir deinen eigenen Anspruch an Wissen und den Umgang damit implementiert. Ich vermisse jetzt schon die Zeiten des unnachgiebigen Optimierens und Probierens und hoffe, dass ich das nicht auf ewig muss.

Ein großer Dank und meine Bewunderung geht an Prof. Dr. Alfred Winter, der mir als Dozent und Chef in guter Erinnerung bleibt. Ich freue mich auf die Veranstaltungen im Masterstudium und auf viele anregende Gespräche auch außerhalb des Hörsaals.

Ein großer Dank geht auch an die lieben Menschen, die diese Arbeit Korrektur gelesen haben. Das weiß ich sehr zu schätzen und es hat mir viel gebracht.

Den Rahmen ziehe ich zur Struktur in meinem privaten Umfeld. Ich mag unsere neue Dorf-Wohngemeinschaft gemeinsam mit meiner tollen Tochter Lana und ihrer starken Mama Sandra Müller sehr. Ich denke, sie wird uns allen helfen, strukturiert nach vorn gehen zu können. Vielen Dank an alle meine Mädels, Ihr seid super!

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

AKRONYME

3LGM²	3-Layer Graph-based Meta-Model
BGP	Basic Graph Pattern
BPMN	Business Process Model and Notation
CIO	Chief Information Officer
CSS	Cascading Style Sheets
CSSE	Center for Systems Science and Engineering
CSV	Comma-Separated Values
DCMI	Dublin Core Metadata Initiative
DDL	Data Definition Language
DFG	Deutschen Forschungsgemeinschaft e.V.
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
ES6	EcmaScript2015
GMDS	Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V.
HITO	Health IT Ontology
HTML	Hypertext Markup Language
HTTP	Hypertext Protocol
IMISE	Institut für medizinische Informatik, Statistik und Epidemiologie
IRI	Internationalized Ressource Identifier
ISB	Institute for Systems Biology
JS	JavaScript
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KI	Künstliche Intelligenz
KIS	Krankenhausinformationssystem
LOD	Linked Open Data
MIG	Management von Informationssystemen im Gesundheitswesen
npm	Node Package Manager
OWA	Open World Assumption
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema

SKOS	Simple Knowledge Organization System
SNIK	Semantisches Netz des Informationsmanagements im Krankenhaus
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SVN	Subversion
TDD	Test Driven Development
Turtle	Terse RDF Triple Language
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XLS	Excel Spreadsheet
XML	Extensible Markup Language
XSD	XML Schema

1 EINLEITUNG

Diese Arbeit befasst sich mit der Selektion und parallelen Anzeige sowie Bearbeitung von Teilgraphen semantischer Netze zur ontologiegestützten Wissensvermittlung.

1.1 GEGENSTAND UND MOTIVATION

1.1.1 *Gegenstand*

„Wissen (lat. videre: sehen) ist die Kenntnis über den in einem Fachgebiet zu gegebener Zeit gegebenen Konsens, vor allem bezüglich einer gültigen Terminologie, erlaubter Interpretationen, bestehender Zusammenhänge und Gesetzmäßigkeiten, empfehlenswerter Methoden und Handlungen."

— Ammenwerth u. a., 2014 —

Das in dieser Arbeit betrachtete Fachgebiet ist die medizinische Informatik. Diese fand erstmals namentliche Erwähnung im Jahre 1969, im Titel der Zeitschrift „Revue informatique medical" in Frankreich (Köhler, 1973).

Im für diese Arbeit relevanten Teilgebiet der Krankenhausinformationssysteme (KIS) und deren Management werden die teils recht verschiedenen Ansätze und komplexen Zusammenhänge deutlich.

Ziel des DFG-Projektes *Semantisches Netz des Informationsmanagements im Krankenhaus (SNIK)*[1] ist es, diese für Experten und in der Lehre besser vermitteln, systematischer weiterentwickeln, branchenspezifisch anpassen und besser integrieren zu können (vgl. Winter und Paech, 2020).

Dazu wurde das zu großen Teilen praxisbasierte Wissen aus verschiedenen Büchern, Artikeln, Standards und mit Experten geführten Interviews extrahiert und in einer Ontologie vereinigt. Somit werden Zusammenhänge und Verbindungen zwischen Daten der verschiedenen Wissensbasen erkennbar. Die Basis der entwickelten Ontologie bildet ein spezielles eigenes Metamodell. Dieses verbindet Fachbegriffe durch eigens definierte Relationen.

Mit dem SNIK Graph als Webanwendung existiert ein visuelles Werkzeug, welches das aus verschiedenen Quellen extrahierte Wissen übersichtlich aufbereitet als Graph darstellt.

Aktuell wird diese Visualisierung einmal pro Jahr in einer Mastervorlesung an der Universität Leipzig eingesetzt. Im Spiel „SNIK-Bingo" bewegen sich die Studenten über die verschiedenen Fachbegriffe durch die Ontologie. Weiterhin wird das „SNIK-Quiz" eingesetzt, welches automatisch gebildete Fragen stellt.

1 Projektpartner: Institut für medizinische Informatik, Statistik und Epidemiologie (IMISE) Leipzig und Universität Heidelberg, Institut für Informatik, Lehrstuhl Software Engineering. Laufzeit: 2014–2019. Förderkennzeichen 1605/7-1 und 1387/8-1.

Neben der korrekten werden auch falsche Antworten generiert, die durch ihre semantische Nähe ebenfalls sehr wahrscheinlich sind.

1.1.2 *Problematik*

Ein Resultat aus dem SNIK-Projekt ist, dass der Graph durchaus nützlich für die Lehre sein kann. So ist es aktuell möglich, mit dem SNIK Graph verschiedene graph-basierte Filter- und Darstellungsoperationen durchzuführen. Damit wird das gespeicherte Wissen eingegrenzt und bestimmte Beziehungen gesondert aufgezeigt.

Es existieren bereits konkrete Anwendungsszenarien für die Lehre, jedoch ist die Umsetzung mittels der Visualisierung mit den momentanen Funktionen nur schwierig umsetzbar. Es fehlt die Möglichkeit, die für eine Unterrichtseinheit relevanten Begriffe als zusammenhängenden Teilgraph darzustellen und zu extrahieren.

Anonym, 2019 liefert bereits eine Definition und Vorschläge zur Erstellung und Extraktion von „sinnvollen Teilgraphen“. Dabei wurde auch der Austausch mit den Lehrenden gesucht, um die Resultate zu validieren. Die Ergebnisse dieser Arbeit sind noch nicht implementiert.

Der momentane Entwicklungsstand von SNIK Graph und der Ontologie ist für den Lehrbetrieb somit nur bedingt nützlich.

Neben den teils technisch noch nicht gegebenen Möglichkeiten sind die Vorbereitungszeiten für die Lehrenden zu hoch. Es mangelt außerdem an der Möglichkeit, geschaffene Zwischenergebnisse speichern und zu einem späteren Zeitpunkt wieder aufrufen zu können.

1.1.3 *Motivation*

Da das SNIK-Projekt bereits beendet ist, hingegen das ausgeschriebene Ziel der Verbesserung der Lehre nur bedingt erfüllt ist, soll diese Arbeit diesen abschließenden Schritt möglich machen.

Für die Lehrenden ist die Umsetzung der Ziele dieser Arbeit eine Erweiterung ihrer didaktischen Möglichkeiten. Der Lehrbetrieb am IMISE kann damit positiv beeinflusst werden.

Neben den Lehrenden der medizinischen Informatik können auch die Studierenden ihren Nutzen daraus ziehen. Die Vorlesungsnach- und Prüfungsvorbereitung mit einem interaktiven, übersichtlichen und logisch aufgebautem Tool, welches den vermittelten Stoff darstellt, ist auch aus didaktischer Sicht eine Form, die Ausbildung zu verbessern.

Um den SNIK Graph in der Lehre einsetzen zu können, soll es möglich sein, in mehreren Fenstern unabhängige Ansichten zu bearbeiten.

Nicht zuletzt ist der SNIK Graph jetzt schon auf andere Wissensbasen transferierbar und könnte diese auf die angedachte Art repräsentieren.

1.2 PROBLEMSTELLUNG

Um sich der Lösung der Problematik anzunähern, werden zunächst die folgenden beiden Probleme formuliert.

- Problem P1: Beim Explorieren von Teilgraphen ist es nicht möglich, an mehr als einem Teilgraphen gleichzeitig zu arbeiten.
- Problem P2: Die schrittweise Erweiterung von Teilgraphen in SNIK Graph zur Abbildung eines behandelten Wissensausschnitts in der Lehre des Managements von Informationssystemen im Gesundheitswesen (MIG) ist zeitaufwändig.

1.3 ZIELSETZUNG

Um eine Lösung der Probleme herbeizuführen, soll mit mehreren Fenstern gearbeitet werden, die es den Lehrenden gestatten, in Vorbereitung Ihrer Lehrveranstaltungen Grafiken zu erstellen, die den behandelten Stoff mit all seinen Beziehungen darstellen. Diese sollen nahezu beliebig verändert und erweitert werden können.
Dafür werden die folgenden Ziele definiert.

- Ziele zu Problem P1:
  - Ziel Z1.1: Aufteilung des SNIK Graph-Fensters in mehrere Unterfenster mit unabhängigen Ansichten.
  - Ziel Z1.2: Kopieren, Verschieben und Einfügen von Teilgraphen in andere Fenster.
  - Ziel Z1.3: Postions- und Sichtbarkeitsänderungen von Elementen in einem Fenster haben keinen Einfluss auf die anderen Fenster.
- Ziel zu Problem P2:
  - Ziel Z2.1: Die existierende Suchfunktion soll dahingehend modifiziert werden, dass gefundene Elemente mit bereits Angezeigten in Beziehung gesetzt werden.
  - Ziel Z2.2: Unterstützung der Lehrkräfte für die kommende Lehrveranstaltung im Wintersemester 2020/21.

1.4 AUFGABENSTELLUNG

Zur erfolgreichen Umsetzung der oben definierten Ziele werden die folgenden Aufgaben festgelegt.

- Aufgaben zu Ziel 1.1:
  - Aufgabe A1.1.1: Finden einer geeigneten Layout-Manager-Bibliothek.
  - Aufgabe A1.1.2: Integrieren dieser Bibliothek in SNIK Graph.

- Aufgabe zu Ziel 1.2:
  - Aufgabe A1.2.1: Implementierung der Austausch-Funktionen von Teilgraphen zwischen den Fenstern.
- Aufgabe zu Ziel 1.3:
  - Aufgabe A1.3.1: Isolieren der Positions- und Sichtbarkeitseigenschaften innerhalb der Fenster.
- Aufgaben zu Ziel 2.1:
  - Aufgabe A2.1.1: Beschränkung der Suche auf das aktive Fenster.
  - Aufgabe A2.1.2: Sichtbarmachen von gesuchten Elementen innerhalb des aktiven Fensters.
- Aufgaben zu Ziel 2.2:
  - Aufgabe A2.2.1: Erstellen von Beispielszenarien für den praktischen Einsatz.
  - Aufgabe A2.2.2: Entwickeln einer Anleitung, die das neue Verfahren in kurzer Zeit erlern- und umsetzbar macht.

1.5 AUFBAU DER ARBEIT

Nach der Einführung der Grundlagen und formalen Definitionen, die zum Verständnis dieser Arbeit notwendig sind, soll auf den aktuellen Stand der Forschung eingegangen werden. Es wird analysiert welche Lösungen für die aktuell vorliegende Situation bereits auf dem Markt zu finden sind. Der daraus entwickelte Lösungsansatz soll dann im fünften Kapitel ausgeführt werden. Anschließend folgt die Präsentation und Diskussion der Ergebnisse. Die Arbeit endet mit einer Zusammenfassung.

2 GRUNDLAGEN

2.1 MEDIZINISCHE INFORMATIK

Die Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V. (GMDS)[1] definiert Medizinische Informatik wie folgt:
„Die Medizinische Informatik ist die Wissenschaft der systematischen Erschließung, Verwaltung, Aufbewahrung, Verarbeitung und Bereitstellung von Daten, Informationen und Wissen in der Medizin und im Gesundheitswesen. Sie ist von dem Streben geleitet, damit zur Gestaltung der bestmöglichen Gesundheitsversorgung beizutragen."[2]

Ein wichtiges Ziel ist die punktgenaue Zurverfügungstellung von Daten zur richtigen Zeit, am richtigen Ort und im richtigen Kontext, um damit die Medizin in der Praxis unterstützen zu können. Wichtige Herausforderungen der medizinischen Informatik sind zum Beispiel die elektronische Patientenakte, die Unterstützung von Therapie und Diagnostik, sowie der Epidemiologie mit computergestützten Verfahren und die Qualitätssicherung der Prozesse im Gesundheitswesen.

2.1.1 *Daten, Information und Wissen*

Zwischen den Begriffen *Daten, Information* und *Wissen* besteht ein hierarchischer Zusammenhang. So bilden *Daten* (lat. datum: gegeben) als Gebilde aus Zeichen mithilfe einer Syntax die Grundlage für *Information* (Ammenwerth u. a., 2014). Diese wird durch die Kenntnis über bestimmte Sachverhalte aus den Daten gewonnen und versieht diese mit einer Form. Im Vergleich zur Information bezieht sich *Wissen* nicht nur auf einzelne Objekte sondern beschreibt die Zusammenhänge und Gesetzmäßigkeiten zwischen mehrelementigen Mengen an Objekten (Ammenwerth u. a., 2014). Wissen ist daher ein intelligentes Netzwerk aus Informationen (vgl. Abb. 2.1).

Als Stellvertreter für ein sozio-technisches, also ein vom Menschen gestaltetes System ist ein beliebiges Unternehmen denkbar, zum Beispiel eine Arztpraxis. Die Objekte sind unter anderem der Arzt, die Schwestern und Patienten sowie Computer, der Patientenaktenschrank und das Chipkartenlesegerät. Das Zusammenwirken dieser Objekte nach klaren Festlegungen ermöglicht die Patientenversorgung und ein wirtschaftliches Auskommen der Angestellten.

Da Systeme schnell sehr komplex werden können, ist es oft sinnvoll, zur Analyse, Optimierung und zum Management bestimmte *Teilsysteme* zu betrachten. Auch werden häufig *Modelle* von Systemen erstellt, die ein tieferes Verständnis ermöglichen.

1 Aktueller Präsident: Prof. Dr. Alfred Winter, Leipzig, vgl. https://www.gmds.de/ueber-uns/organisation/#c309

2 vgl. https://www.gmds.de/aktivitaeten/medizinische-informatik/

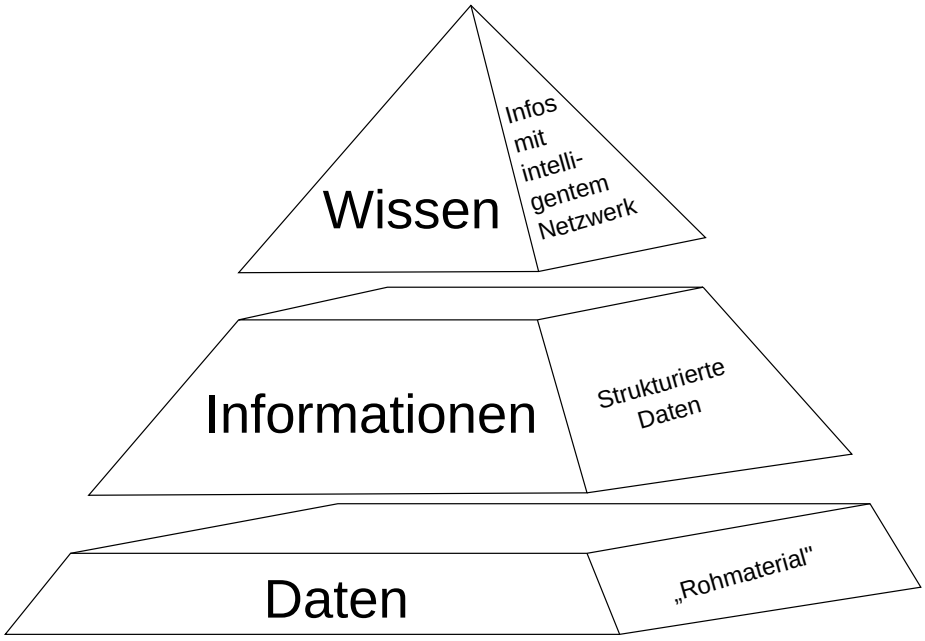

Abbildung 2.1: Die Wissenspyramide. Quelle:
https://de.wikipedia.org/wiki/Datei:Wissenspyramide.svg

Definition 1 (Informationssystem) *Das sozio-technische Teilsystem einer Einrichtung, welches die informationsverarbeitenden Aktivitäten und die daran beteiligten Akteure beschreibt, wird* Informationssystem *genannt (Ammenwerth u. a., 2014). Informationssysteme beschreiben den systematischen Informationsverarbeitungs- und Speicherungsprozess.*

2.1.2 *Krankenhausinformationssystem*

Definition 2 (Krankenhausinformationssystem) *„Ein Krankenhausinformationssystem (KIS) ist das Informationssystem der Einrichtung ‚Krankenhaus'"(Ammenwerth u. a., 2014).*

Es lassen sich je nach der Art der eingesetzten Handlungsträger *rechnerunterstützte* von *konventionellen* Informationssystemen unterscheiden. Rechnerunterstützt ist ein Teilsystem des KIS, welches sich ausschließlich auf Rechnersysteme zur Informationsverarbeitung stützt, zum Beispiel Computer, Drucker und Datenbanken. Objekte konventioneller Informationssysteme sind beispielsweise Papierakten, Telefone oder Rohrpostanlagen.

Das KIS-*Management* ist eine wichtige Teildisziplin der medizinischen Informatik und bedarf umfangreicher Kenntnis der Teilsysteme, Modelle und Prozessabläufe.

2.1.3 *Das SNIK-Projekt*

Das Projekt Semantisches Netz des Informationsmanagements im Krankenhaus (SNIK)[3], welches von der Deutschen Forschungsgemeinschaft e.V. (DFG) gefördert wird, befasst sich mit dem Wissen aus dem Bereich Management von

Informationssystemen im Gesundheitswesen (MIG). Nach Höffner u. a., 2019 wird im Rahmen von SNIK eine Ontologie auf der Basis eines Metamodells beschrieben. Dieses Modell definiert ein Vokabular zur Beschreibung der Domäne der Krankenhausinformationssysteme (KIS). Drei disjunkte Basisklassen beschreiben die Sachverhalte nach folgendem Schema: „Wer (*Rolle*) macht was (*Funktion*) und welche Information wird dafür benötigt (*Objekttyp*).“

Das Wissen für SNIK stammt aus drei Lehrbüchern, einem IT-Standard sowie einem Interview mit einem Chief Information Officer (CIO). Nach der Extraktion werden die Daten nach den Prinzipien von Linked Open Data (LOD) aufbereitet und unter Verwendung von weiteren Technologien des Semantic Web gespeichert und veröffentlicht.

Wichtige Resultate des SNIK-Projektes sind, neben der Ontologie, die Darstellungen der aus verschiedenen Quellen extrahierten Daten. Es gibt einen *RDF Dump* mit allen Triples in Turtle-Syntax, die über einen *SPARQL Protocol and RDF Query Language (SPARQL)-Endpunkt*[4] abgefragt werden können, den *LodLive RDF Browser*[5], und die Visualisierung *SNIK-Graph*[6]. Der SNIK Graph findet Einsatz in der Lehre und wird zum Beispiel für *SNIK-Bingo* oder im Rahmen des *SNIK-Quiz* verwendet.

2.2 GRAPHENTHEORIE

Die folgenden Definitionen zur Graphentheorie sind Diestel, 2012 entlehnt. Das verwendete Beispiel wurde selbst gewählt.

2.2.1 *Graph*

Definition 3 (Graph) *Ein* Graph *ist ein Paar* $G = (V, E)$ *disjunkter Mengen mit* $E \subseteq V^2$*; die Elemente von* E *sind also Paare mit Einträgen aus der Menge* V*. Die Elemente von* V *nennt man* Ecken *oder* Knoten *des Graphen* G*, die Elemente von* E *seine* Kanten*.*

Bildlich kann G dargestellt werden, indem man die Knoten als Punkte zeichnet und zwei dieser Punkte immer dann mit einer Linie verbunden werden, wenn die entsprechenden Punkte eine Kante ergeben. Die Art, wie die Knoten und Kanten gezeichnet werden (geradlinig, geschwungen, disjunkt oder überkreuzt), ist meist eine Frage der Zweckmäßigkeit und beeinflusst die formale Definition des Graphen nicht. Kanten können dabei *gerichtet* ($A \to B$) oder *ungerichtet* ($A - B$) sein. Im Folgenden werden nur gerichtete Graphen betrachtet, da auch jede Kante in einem ungerichteten Graphen als bidirektionale Kante ($A \leftrightarrow B$) aufgefasst werden kann.

Abb. 2.2 zeigt eine bildliche Darstellung des gerichteten Graphen

$$G_{Bsp.} := (V, E)$$

3 https://www.snik.eu/
4 http://www.snik.eu/sparql
5 http://www.snik.eu/ontology
6 https://www.snik.eu/graph/

mit

$$E := \{a, b, c, d, e, f, g, h\}$$

und

$$V := \{0, 1, 2\}.$$

Dabei sind die Paare in E (also die Kanten):

- $a := (0;0)$,
- $b := (1;0)$,
- $c := (0;1)$,
- $d := (0;2)$,
- $e := (2;1)$,
- $f := (1;2)$,
- $g := (2;1)$,
- $h := (2;2)$.

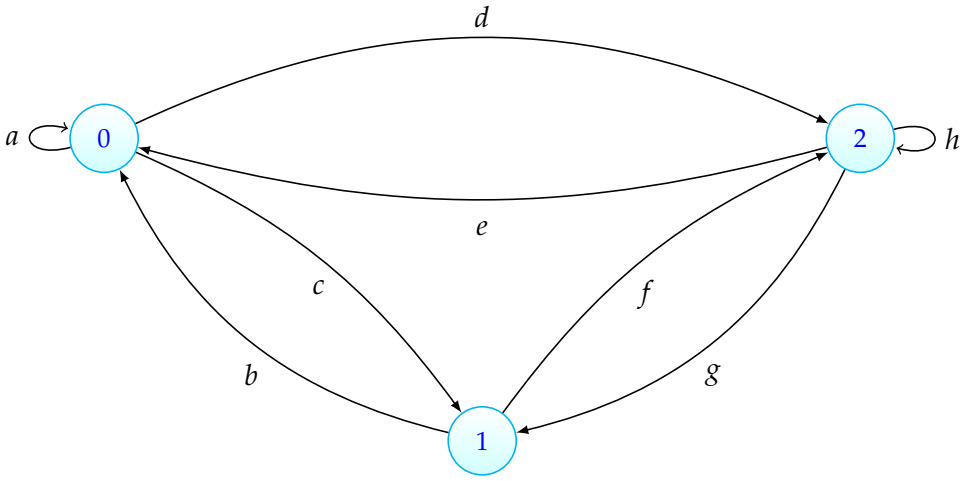

Abbildung 2.2: Der gerichtete Graph $G_{Bsp.}$

2.2.2 *Teilgraph*

Definition 4 (Teilgraph) *Wir setzen* $G \cup G' := (V \cup V', E \cup E')$ *und außerdem* $G \cap G' := (V \cap V', E \cap E')$. *Gilt* $V' \subseteq V$ *und* $E' \subseteq E$, *so ist* G' *ein* Teilgraph *von* G. G *wird dann auch* Obergraph *von* G' *genannt, geschrieben* $G' \subseteq G$.

Somit enthält G den Graphen G'.

Definition 5 (Induzierter Teilgraph) *Ein Teilgraph* G' *heißt* induziert *oder* aufgespannt *(von* V' *in* G*), wenn er* alle *Kanten* $x, y \in E$ *mit* $x, y \in V'$ *enthält. Einen solchen induzierten Teilgraphen nennen wir einen* Untergraphen.

V	Eingangsgrad $d^-_{G_{Bsp.}}$	Ausgangsgrad $d^+_{G_{Bsp.}}$
0	$\lvert\{a,b,e\}\rvert = 3$	$\lvert\{a,c,d\}\rvert = 3$
1	$\lvert\{c,g\}\rvert = 2$	$\lvert\{b,f\}\rvert = 2$
2	$\lvert\{d,f,h\}\rvert = 3$	$\lvert\{e,g,h\}\rvert = 3$

Tabelle 2.1: Knotengrade von $G_{Bsp.}$

2.2.3 *Der Grad eines Knotens*

Definition 6 (Grad) *Sei G ein gerichteter Graph. Der* Grad*, auch* Knotengrad *oder* Valenz *genannt, bezeichnet die Anzahl der an einen Knoten $v \in V$ angrenzenden Kanten. Man unterscheidet dabei zwischen:*

- *dem* Eingangsgrad $d^-_G(v)$ *als Kardinalität der Menge der eingehenden Kanten und*
- *dem* Ausgangsgrad $d^+_G(v)$ *als Kardinalität der Menge der ausgehenden Kanten.*

Ein Knoten mit Eingangsgrad $d^-_G(v) = 0$ heißt Quelle*, ein Knoten mit Ausgangsgrad $d^+_G(v) = 0$ heißt* Senke*.*

Tabelle 2.1 fasst die Ein- und Ausgangsgrade des Beispielgraphen $G_{Bsp.}$ (siehe Abb. 2.2 zusammen.

2.2.4 *Pfade*

Definition 7 (Pfad) *Ein* Pfad *(oder* Weg, Kantenzug, Kantenfolge*) ist in der Graphentheorie eine Abfolge von Knoten, in der jeweils die zwei aufeinanderfolgenden Knoten durch eine Kante verbunden sind.*

Sind bei einem Pfad Start- und Endknoten identisch, wird von einem *Zyklus* gesprochen. Ist dieser Start- und Endpunkt der einzige sich wiederholende Knoten in der Folge, so heißt dieser Zyklus *Kreis*.

2.3 SEMANTIC WEB

Im Rahmen dieser Arbeit wird sich verschiedener Technologien des Semantic Web bedient, daher soll im Folgenden eine Einführung in die wichtigsten Begriffe zu diesem Thema gegeben werden.

Zur Frage, was ein *Web*, ein Netz in diesem Kontext darstellt, führen Allemang und Hendler, 2008 aus, dass es sich um eine technische Idee handelt, die es ermöglicht, Informationen über die ganze Welt hinweg teilen zu können. Dafür ist eine offene Gemeinschaft essentiell, so dass jeder seine Ideen, sichtbar für die Gesamtheit, zur Verfügung stellen kann. Diese Offenheit führt zu tieferem Verständnis von verschiedensten Themen. Der Grundgedanke hinter dem Semantic Web ist, ein Netz aus konsistenten Informationen aufzubauen. Dabei müssen die Daten an sich nicht diese Eigenschaften haben, es geht darum, „die richtigen

Daten zum richtigen Platz“ (Allemang und Hendler, 2008) zu bekommen, sodass eine Infrastruktur der Information durch Verknüpfungen aufgebaut werden kann. Das Semantic Web, oder Netz der Daten, ist eine Erweiterung des allgemein bekannten World Wide Web (WWW), welches Daten miteinander verknüpft und somit maschinenlesbar macht. Es wurde im Jahr 2001 von Sir Timothy John Berners-Lee, dem „Erfinder des Internets“ in Berners-Lee, Hendler und Lassila, 2001 erstmals als solches bezeichnet. Durch die semantischen Verknüpfungen können automatische Antworten auf Fragestellungen wie zum Beispiel „Welche Sehenswürdigkeit in New York City wird im Monat März von den meisten Touristen besucht?“ gefunden werden. Das Semantic Web soll auch Computern ermöglichen, Ambiguitäten und Kontextabhängigkeiten der natürlichen Sprache zu verstehen. Die wichtigsten Technologien sind im *Semantic Web Technology Stack* zusammengefasst.

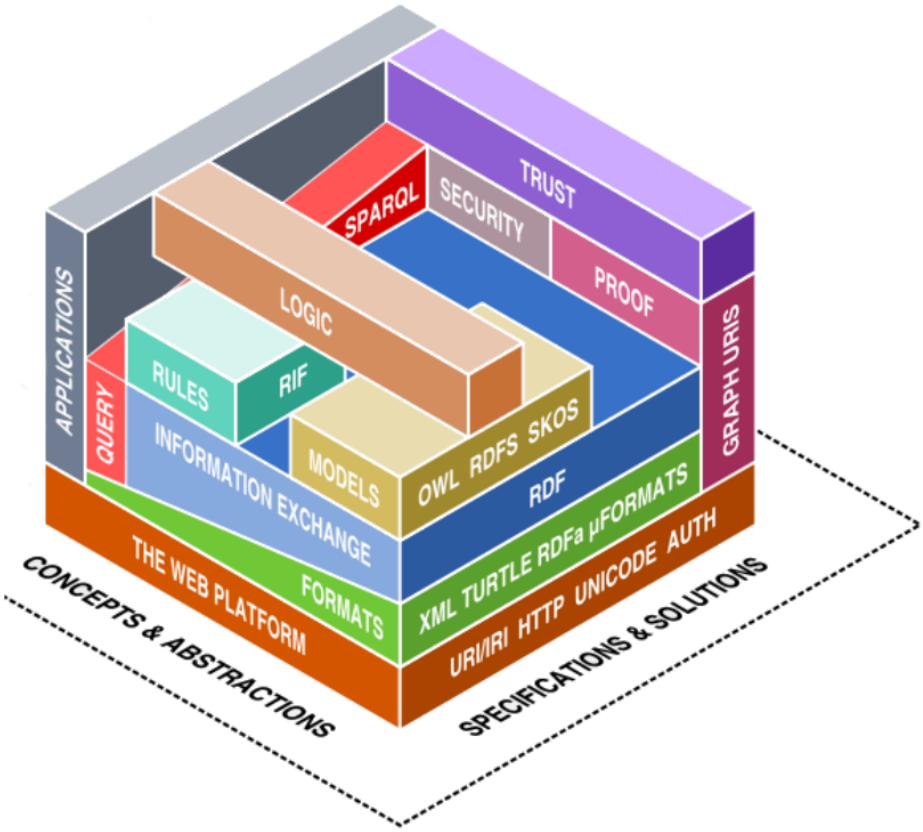

Abbildung 2.3: Der Semantic Web Technology Stack.
Quelle (bearbeitet): `http://www.zfdg.de/sites/default/files/medien/semantic_web_2017_002.png`

2.3.1 *URIs und URLs*

Eine Uniform Resource Identifier (URI) ist eine Zeichenkette, die eindeutig eine abstrakte oder physische Ressource identifiziert (Berners-Lee, Fielding und Masinter, 2005). Dabei können URIs oft durch vorher definierte Präfixe

verkürzt und damit vereinfacht werden. Ein Beispiel bietet die Ressource `http://dbpedia.org/resource/Leipzig`, die zu `dbr:Leipzig` gekürzt werden kann.

Eine Uniform Resource Locator (URL) ist eine URI, die eine Adresse (engl. locator) enthält, über die die beschriebene Ressource erreichbar ist (Berners-Lee, Masinter und McCahill, 1994).

2.3.2 *World Wide Web*

Das World Wide Web (WWW) ist laut Berners-Lee u. a., 1994 eine Informationswelt ohne Beschränkungen, in der alle Objekte eine Referenz besitzen, über die sie erreicht werden können. Die Objekte sind untereinander durch Verknüpfungen verbunden und bilden somit eine Netzstruktur.

Dazu wird neben URIs und URLs ein allgemeingültiges Netzwerkprotokoll (das Hypertext Protocol (HTTP)) benötigt. Weiterhin wird eine Auszeichnungssprache (englisch *markup language*) definiert, die Hypertext Markup Language (HTML). Sie muss von jedem Client verstanden werden und dient zum Beispiel zur Übersetzung von Texten und Menüs.

Seit Oktober 1994 ist das World Wide Web Consortium (W3C) für die Standardisierung der Technologien des WWWs zuständig. Begründer sowie Vorsitzender der Mitgliederorganisation ist Sir Tim Berners-Lee (vgl. Berners-Lee, 2006).

2.3.3 *Linked (Open) Data*

Die Idee von Linked Data ist laut Berners-Lee, 2010, Daten nicht nur in das WWW zu stellen, sondern sie so zu verlinken, dass Menschen und Maschinen sie lesen und damit das Netz der Daten entdecken können. So können ausgehend von einem Startpunkt andere, verwandte Daten gefunden werden. Weiterhin sollen die Daten frei verfügbar, also mit einer offenen Lizenz versehen sein (Linked Open Data (LOD)). Um diesen Prozess zu unterstützen, schuf Tim Berners-Lee das 5-Sterne-Modell für offene Daten (siehe Abb. 2.4).

Dabei steht der erste Stern für die Erreichbarkeit unter freier Lizenz im Netz, unabhängig vom verwendeten Datenformat. Der zweite Stern verlangt strukturierte, maschinenlesbare Daten, wie zum Beispiel Tabellen im XLS-Format (Excel Spreadsheet). Für den dritten Stern soll zusätzlich ein nicht-propreitäres Format genutzt werden, zum Beispiel Comma-Separated Values (CSV) statt Excel Spreadsheet (XLS). Der vierte Stern schreibt vor, die Daten mithilfe von W3C-Standards wie RDF und SPARQL identifizierbar zu machen. Um das komplette 5-Sterne-Datenmodell zu erfüllen, sollen die Daten zusätzlich zu Daten anderer Personen oder Institutionen verlinkt sein um Kontext bereitzustellen. (Berners-Lee, 2010)

Ein beeindruckendes Beispiel für Linked Open Data ist das gemeinschaftliche Universitätsprojekt *DBpedia*[7], welches Artikel von Wikipedia nach den Konzepten von LOD transformiert und somit die Weiterentwicklung des Semantic Web vorantreibt. Beteiligt an dem 2007 gegründeten Projekt sind neben Forschern

7 `https://wiki.dbpedia.org/`

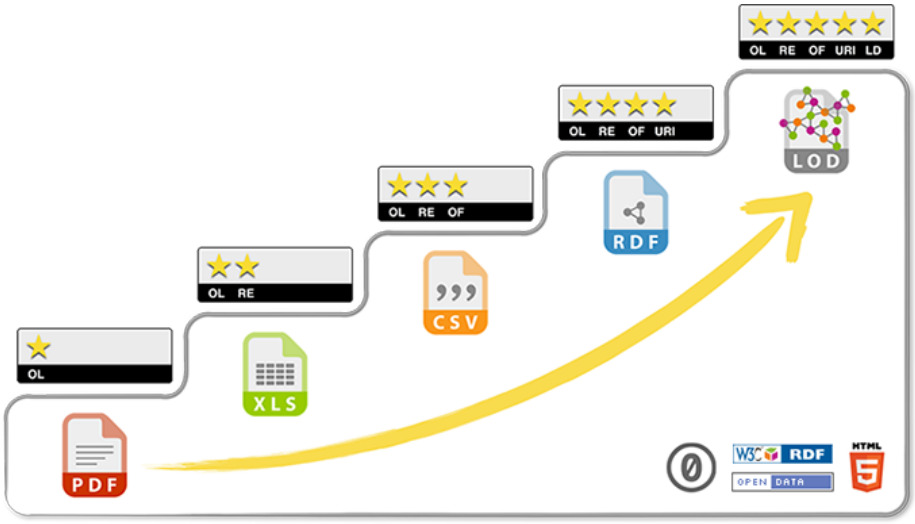

Abbildung 2.4: Das 5-Sterne-Modell für Offene Daten nach Berners-Lee.
Quelle: `https://5stardata.info/de/`

der Universität Leipzig[8] auch Abteilungen der Universität Mannheim[9] und des Hasso-Plattner-Instituts Potsdam[10].

2.3.4 *Resource Description Framework*

Im Bereich der natürlichen Sprachen bezeichnet *Semantik* den Weg von Worten hin zu ihrer Bedeutung. So kann das Wort „Mars“ in unserem Sprachraum für Verschiedenes stehen. Mars ist der römische Kriegsgott, der „rote Planet“ oder auch ein Schokoriegel. Bezogen auf Programmiersprachen meint Semantik, dass man sich „normalerweise auf die Abbildung der Sprachsyntax auf einen Formalismus, der die Bedeutung dieser Sprache ausdrückt“ (Allemang und Hendler, 2008) bezieht. Das WWW besteht aus verlinkten Dokumenten während das Semantic Web aus *Ressourcen* besteht. Eine Ressource kann dabei Alles von Interesse sein, ein Name, Honigdachse, „die ersten 100 Nachkommastellen von Pi“ oder „alle bekannten Rezepte für Smørebrød“. Dieser Begriff wurde vom W3C als Standard für solche Entitäten definiert und findet namentliche Erwähnung in der Technologie Resource Description Framework (RDF).
Der RDF-Standard[11] bildet als Beschreibungssprache für web-basierte Ressourcen die Basis für das Semantic Web.
Ressourcen (URIs[12]) werden durch *Triples* beschrieben.

8 hier speziell das AKSW: `http://aksw.org/Projects/DBpedia.html`
9 Mannheim Linked Data Catalog: `http://linkeddatacatalog.dws.informatik.uni-mannheim.de/de/dataset/dbpedia`
10 Das HPI leistet mit seiner Initiative *openHPI* einen wichtigen Beitrag zum Verständnis der Technologien des Semantic Web: `https://open.hpi.de/courses?q=semanticweb`
11 vgl. `https://www.w3.org/TR/rdf11-concepts/`
12 Außerdem können Ressourcen auch durch Internationalized Ressource Identifiers (IRIs) definiert werden, welche eine größere Menge von Unicode Characters zulassen.

Definition 8 (RDF Triple) *Ein RDF Triple besteht nach Richard Cyganiak, 2014 aus drei Komponenten:*

- *dem* Subjekt *(engl. subject), was ein URI oder ein leerer Knoten (engl. blank node) sein kann,*
- *dem* Prädikat *(engl. predicate), was ein URI ist und*
- *dem* Objekt *(engl. object), was ein URI, ein Literal oder ein leerer Knoten sein kann.*

Es wird in der Reihenfolge Subjekt, Prädikat, Objekt dargestellt und kann auch als Kante mit dem Label predicate *zwischen* subject *und* object *aufgefasst werden. Dabei können Objekte, solang sie eine URI besitzen, als neue Subjekte fungieren.*

URIs, Literale und blank nodes werden *RDF Terme* genannt. Sie sind voneinander verschieden und klar unterscheidbar. *Blank nodes*, sogennante leere Knoten, beschreiben eine Existenz ohne Identifizierbarkeit und Referenz. Sie können zum Beispiel ähnlich gelagerte Fakten mit mehreren Prädikaten zusammenfassen. Dabei sind sie nicht „von außen“ erreichbar, da sie nicht referenziert werden können. Innerhalb eines Dokumentes, also zum Beispiel einer Datei mit N-Triples oder Turtle-Triples, können Sie mit einer ID versehen und entsprechend referenziert werden. Literale bestehen aus zwei oder drei Elementen:

- einer lexikalischen Form (Unicode-Zeichenkette), die in Normalform[13] vorliegen muss,
- einem Datentyp-URI zur Identifizierung eines Datentyps, der bestimmt, wie die lexikalische Form auf einen literalen Wert abgebildet wird und
- wenn (und nur wenn) der Datentyp-URI String[14] ist, einem nicht-leeren, wohlgeformten Sprachtag (bestehend aus einem @-Zeichen und einer zweistelligen Sprachkennung, zum Beispiel @en oder @de).

Definition 9 (RDF Graph) *Eine Menge von RDF Triples wird* RDF Graph *genannt. Subjekte und Objekte werden als Knoten, ihre Prädikate als Kanten dargestellt. Die Triples können durch* Queries *abgefragt und manipuliert werden.*

Zur Serialisierung von RDF Triples werden unter anderem die Technologien *N-Triples* und die *Turtle-Syntax* verwendet. N-Triples ist dabei eine Sammlung von RDF Triples, die als Grundlage für zum Beispiel die Erstellung eines RDF Graphen dient. Turtle bietet hingegen für den Menschen besser lesbare Triple-Repräsentationen. Die resultierende Graph-Datenbank wird *Triple-Store* genannt.

13 Hier ist die kanonische Dekomposition gefolgt von der kanonischen Komposition (NFC) gemeint, vgl. `https://unicode.org/reports/tr15/#Norm_Forms`

14 Beschreibung des Datentyps *String*: `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`

2.3.5 *Die Turtle-Syntax*

Mit einer Menge an „syntaktischem Zucker“[15] und weiteren nützlichen Funktionen bildet die Terse RDF Triple Language (Turtle) Notation eine wichtige Erweiterung von RDF Triples und N-Triples. Durch die Annotation *@prefix* können Präfixe festgelegt werden, Teile von URIs, auf denen die Bestandteile der Triples aufbauen. Die Annotation *@base* steht dabei für alles was in spitzen Klammern steht und keinen Präfix hat. Das folgende Beispiel besagt, dass ein Planet ein Himmelskörper ist:

```
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
@base   dbo:<http://dbpedia.org/ontology/>.

  <Planet>  rdfs:subClassOf <CelestialBody>.
```

Turtle unterstützt weiterhin Prädikatlisten, die durch Semikolon getrennte Prädikat-Objekt-Paare dem selben Subjekt zuordnen sowie Objektlisten (gleiches Subjekt und Prädikat, durch Komma getrennte Objekte). Auch typisierte Literale können mit Präfixen beschrieben und damit abgekürzt werden. Die Syntax hierfür funktioniert nach dem Schema:

```
@prefix dbo:<http://dbpedia.org/ontology/>.
@prefix dbr:<http://dbpedia.org/resource/>.
@prefix xsd:<http://www.w3.org/2001/XMLSchema#>.

  dbr:Linus_Torvals dbo:birth_Date  "1969-12-28"^^xsd:date.
```

Leere Knoten (blank nodes) werden mit eckigen Klammern beschrieben und können verschachtelt werden. Auch für die RDF List Datenstrukturen *Container* (offene Liste) und *Collection* (geschlossene Liste) werden Vereinfachungen durch Turtle angeboten.

2.3.6 *Ein Beispiel zu RDF Graphen und Turtle*

In Abb. 2.5 ist ein einfacher RDF Graph dargestellt, der Wissen über den „roten Planeten“ *Mars* repräsentiert. So ist Mars sowohl ein Planet, als auch ein Himmelskörper (celestial body) und trägt den Namen „Mars“. Weiterhin hat er zwei Monde, Phobos und Deimos, die beide von Asaph Hall im Jahr 1877 entdeckt wurden.

Der benötigte Triple Store ist unten in Turtle Syntax angegeben. Dabei wird die zulässige Abkürzung „a“ für `rdf:type` sowie Prädikatlisten verwendet.

```
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
@prefix dbp:<http://dbpedia.org/property/>.
@prefix dbo:<http://dbpedia.org/ontology/>.
@prefix foaf:<http://http://xmlns.com/foaf/0.1/>.
@prefix xsd:<http://www.w3.org/2001/XMLSchema#>.
@base   dbr:<http://dbpedia.org/resource/>.
```

15 Unter dem Begriff „syntaktischer Zucker“ (engl. syntactic sugar) werden in Programmiersprachen Syntaxerweiterungen und -vereinfachungen zusammengefasst, die die Lesbarkeit und Schreibweise für Programmierer verbessern (vgl. `http://ftp.informatik.rwth-aachen.de/jargon300/syntacticsugar.html`).

```
<Mars>           a                dbo:Planet;
                 foaf:name        "Mars"@en.

dbo:Planet       rdfs:subClassOf  dbo:CelestialBody.

<Phobos_(moon)>  a                dbo:CelestialBody;
                 dbp:satelliteOf  <Mars>;
                 dbo:discoverer   <Asaph_Hall>;
                 dbo:discovered   "1877-08-17"^^xsd:date.

<Deimos_(moon)>  a                dbo:CelestialBody;
                 dbp:satelliteOf  <Mars>;
                 dbo:discoverer   <Asaph_Hall>;
                 dbo:discovered   "1877-08-12"^^xsd:date.
```

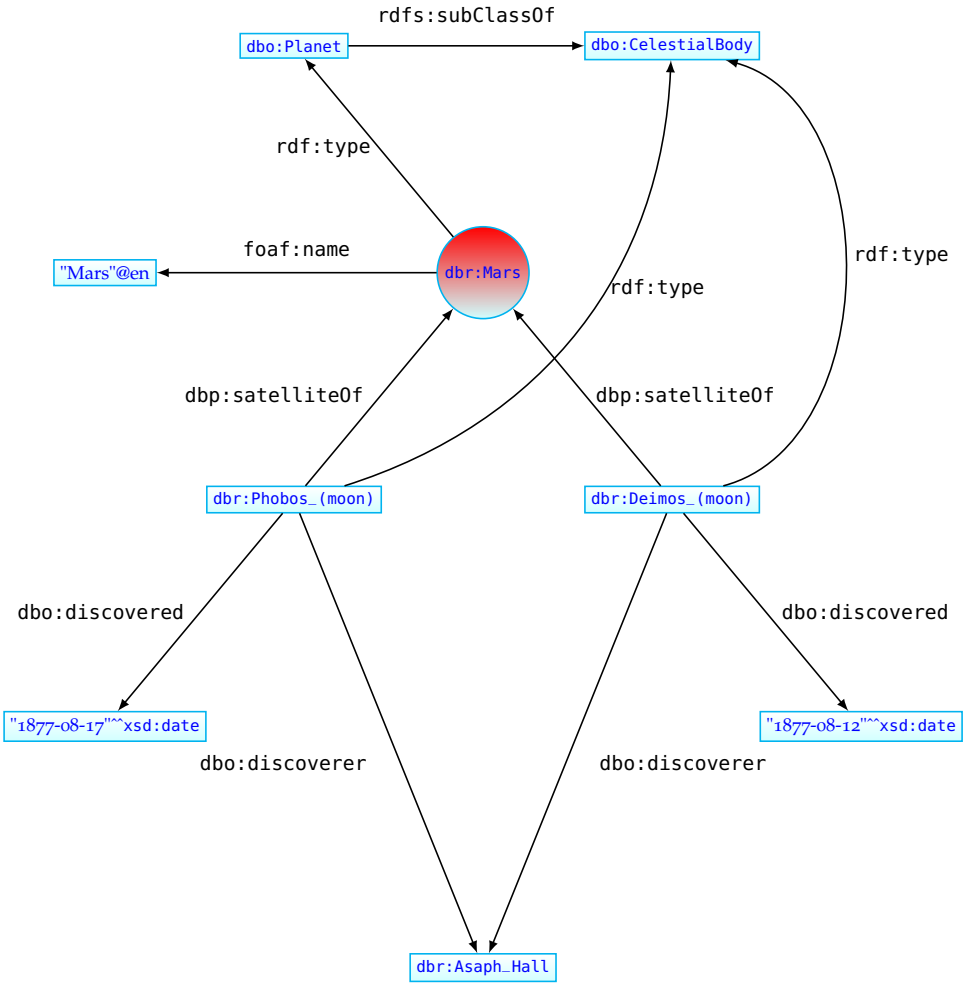

Abbildung 2.5: Ein RDF Graph mit Wissen zum Planeten Mars

Beim Betrachten der Triple und des Graphen wird eine weitere wichtige Eigenschaft der semantischen Beziehungen deutlich, die logischen Inferenzen. Am Beispiel ist `dbr:Mars` vom Typ `dbo:Planet` und `dbo:Planet` eine Unterklasse von `dbo:CelestialBody`. Das bedeutet, dass dann `dbr:Mars` eine Teilmenge

von `dbo:CelestialBody`, also eine Instanz Himmelskörper ist. Aufgrund dieser einfachen Feststellung kann eine zusätzliche Kante gezogen und somit eine neue Beziehung hergestellt werden. (Diese Beziehung ist implizit, da subClassOf eine Teilmengenbeziehung und transitiv ist.)

2.3.7 *RDF Schema*

Für die Repräsentation von Bedeutung reicht RDF allein nicht aus. Sprechende Bezeichner ermöglichen noch keine maschinenverständliche Beschreibung von semantischen Konzepten. Daher gibt es im Semantic Web Technology Stack die „Models"-Schicht. Dort werden Technologien zur Erstellung formaler und logischer Modelle genannt, unter anderem die Web Ontology Language (OWL) und RDF Schema (RDFS). Auf OWL wird im nächsten Abschnitt noch detaillierter eingegangen. RDFS, offiziell die „RDF Vocabulary Description Language" genannt, erlaubt als Erweiterung das Konzept von *Klassen* und *Instanzen*, also eine Art Vererbungshierarchie oder Taxonomie. Im Mars-Beispiel wurde dies auch bereits benutzt, als `rdfs:subClassOf`, um die hierarchische Beziehung zwischen `dbo:Planet` und `dbo:CelestialBody` zu beschreiben. Weiterhin bietet RDFS zusätzlichen semantischen Ausdruck durch die Möglichkeit Definitions- und Wertebereiche von *Properties* (*domain* und *range*) festzulegen. So erwartet zum Beispiel die Property `dbo:birthDate` als `rdfs:domain` eine Entität der Klasse `dbo:Person` und als `rdfs:range` ein Literal vom Typ `xsd:date`. Damit wird ausgesagt, dass das Geburtsdatum ein Attribut zwischen einer Person und eben ihrem Geburtsdatum ist.

RDFS versteht sich auch als Ausdruckssprache für *Beschreibungslogiken*, da sich mit den Elementen der Sprache sowohl ein terminologischer Formalismus (*TBox*) als auch assertionales Wissen (*ABox*) definieren lassen. In der *TBox* wird konzeptuelles Wissen über eine bestimmte Domäne[16] festgelegt. Das bedeutet, es wird definiert, welche Klassen von Objekten es gibt und welche Eigenschaften diese haben. In der *ABox* wird hingegen das Wissen über konkrete Instanzen einer Domäne beschrieben. Somit wird der Zustand der modellierten Welt repräsentiert. Der Zweck dieser Aufteilung ist, dass grundsätzlich das Wissen der TBox fix ist während das assertionale Wissen Veränderungen unterliegt. So wird zum Beispiel ein Zebra immer ein pferdeähnliches Säugetier sein, während die Anordnung und Ausprägung der Streifen von Individuum zu Individuum variiert.

Der Unterschied von RDFS gegenüber anderen Data Definition Languages (DDLs) wie XML Schema (XSD) ist eben diese formale logikbasierte Semantik. Sie basiert auf einem soliden mathematischen Fundament und ermöglicht so, implizites Wissen abzuleiten.

16 Ein Wissensgebiet, also ein "Gebiet menschlichen Wissens, auf dem wissenschaftliche Erkenntnisse vorliegen". Quelle: `https://www.duden.de/rechtschreibung/Wissensgebiet`

2.3.8 *Vokabulare und Ontologien*

Definition 10 (Vokabular) *Ein Vokabular ist allgemein eine Menge an Wörtern, deren Bedeutung bekannt ist. Es ermöglicht Individuen mit dem gleichen Vokabular ohne Missverständnisse zu kommunizieren.*
Im Semantic Web definieren Vokabulare die Konzepte und Beziehungen, die benutzt werden, um eine Domäne angemessen zu repräsentieren. Sie werden genutzt, um Beziehungen (oder „Terme") zu klassifizieren und zu charakterisieren. Außerdem werden mögliche Beschränkungen festgelegt[17].

Ein bekannter Vetreter für ein Vokabular ist das von der Dublin Core Metadata Initiative (DCMI) definierte „Dublin Core Metadata Element Set"[18]. Es handelt sich dabei um eine Sammlung standardisierter Konventionen, um bibliographische Dokumente und Objekte mittels Metadaten[19] zu beschreiben. Diese Datenbeschreibungen können auch mittels Turtle/RDF beschrieben werden.

Das Projekt „Linked Open Vocabularies" hat frei verfügbare Vokabulare gesammelt und auf ihrer Homepage als übersichtlichen Graph (vgl. Abbildung 2.6) dargestellt. Dabei ist ein Vokabular umso größer dargestellt, je öfter es benutzt wird. Die gut durchdachte Suchfunktion hilft Interessierten beim Finden eines für den jeweiligen Zweck geeigneten Vokabulars.

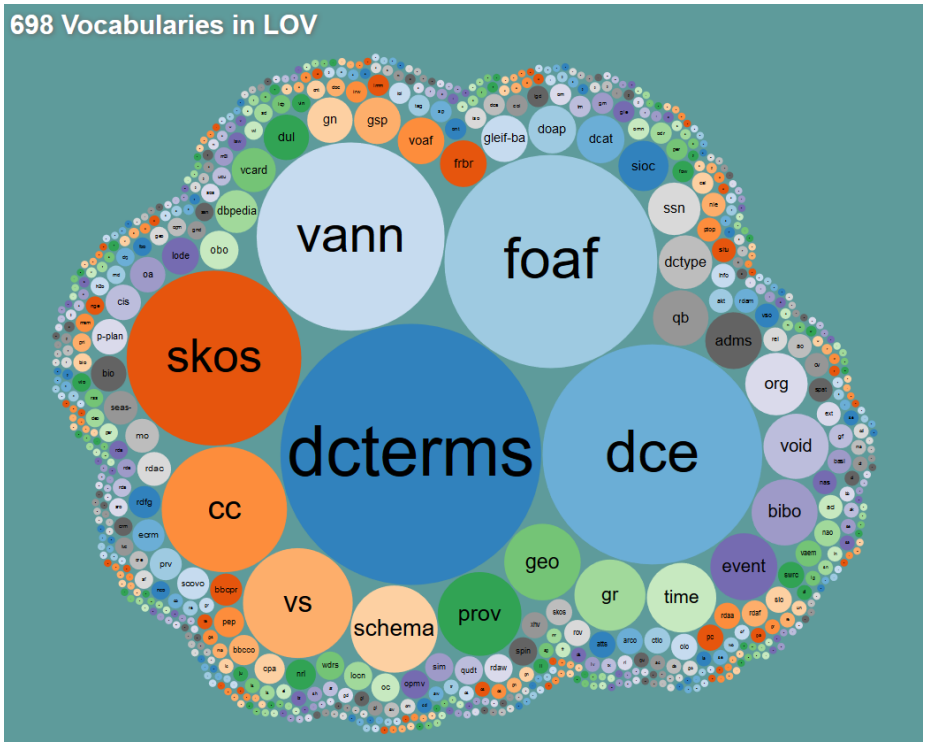

Abbildung 2.6: Vokabulare des Linked Open Vocabularies Projektes.
Quelle: `https://lov.linkeddata.es/dataset/lov/`

17 vgl. `https://www.w3.org/standards/semanticweb/ontology.html`
18 Spezifikationen siehe `https://www.dublincore.org/specifications/dublin-core/dces/`
19 „Metadaten sind Daten über Daten, oder anders ausgedrückt Informationen zu Daten" (vgl. `https://www.itwissen.info/Metadaten-meta-data.html`).

Eine wichtige Annahme im kompletten Umfeld des Semantic Web und speziell für Vokabulare und Ontologien ist die *Open World Assumption (OWA)*. Sie besagt, dass beim Fehlen von genauen Informationen eine gestellte Frage nicht mit „Nein“ beantwortet wird. Ein Beispiel hierzu: Es liegt die Information vor, dass eine Kuh ein Tier ist und vier Beine hat. Die Fragestellung: „Können Kühe fliegen?“. Die Antwort gemäß OWA und unseres Wissens wäre: „Keine Ahnung, aber eventuell ja!“ Diese Denkweise ermöglicht es, mit der naturgegebenen Unvollständigkeit solcher Systeme so umzugehen, dass Sie immer erweiterbar bleiben.

Definition 11 (Ontologie) *Eine Ontologie ist eine explizite, formale Spezifikation einer geteilten Konzeptualisierung (Begriffsbildung). Der Begriff ist der Philosophoie entlehnt, in der Ontologie eine systematische Darstellung der Existenz bzw. des Existierenden beschreibt. Damit ist es eine Art metaphysischer Ansatz, unsere Welt umfassend beschreiben zu können.*
Für Systeme aus dem Bereich Künstliche Intelligenz (KI) „existiert“ das, was repräsentiert werden kann (vgl. Gruber u. a., 1993).

Konzeptualisierung meint dabei ein abstraktes Modell einer Domäne mit identifizierten, relevanten Konzepten und Relationen. *Explizit* bedeutet, dass die Aussage aller dieser Konzepte definiert sein muss. *Formal* steht für eine *Maschinen-Verständlichkeit*, also nicht nur die automatisierbare Lesbarkeit und *geteilt* heißt, dass Konsens bezüglich der Ontologie herrschen sollte.

Ontologien bestehen aus *Klassen*, *Relationen* und *Instanzen*. *Klassen* repräsentieren die Konzepte der Ontologie und werden durch Attribute charakterisiert. Attribute sind Bezeichner-Wert-Paare (zum Beispiel ‘name “Thomas“ ‘). *Relationen* sind spezielle Attribute, deren Werte Objekte von (anderen) Klassen sind. Es können Regeln und Beschränkungen (wie $A \cap B = \emptyset$ oder die Relationen $1:1$ und $n:m$) gesetzt werden. So können Kombinationen zu komplexen Statements/Assertionen erstellt werden. Einen Spezialfall bildet die Nutzung von *formalen Axiomen*, die zusätzlich für Spezialwissen eingeführt werden können. *Instanzen* schließlich beschreiben die Individuen einer Ontologie. Sie repräsentieren in der gebildeten Beschreibungslogik das assertionale Wissen (ABox).

2.3.9 *Die Web Ontology Language OWL*

Mithilfe von RDF und RDFS können nicht alle logischen und semantischen Zusammenhänge abgebildet werden. Als Beispiel seien hier die Negation und das Konzept der Inversivität (zum Beispiel „ist Elternteil von“ und „ist Kind von“) genannt. Daher wurde die Web Ontology Language (OWL) eingeführt, die es in verschiedenen Varianten erlaubt, die Mächtigkeit einer Beschreibungslogik zu erreichen. Verschiedene Varianten können als zumeist entscheidbare Fragmente von *Prädikatenlogik erster Stufe* verstanden werden. So existieren OWLLite, OWLDL und OWLFull, die unterschiedliche Mächtigkeit besitzen. Der Namensraum von OWL ist @prefix owl: `<http://www.w3.org/2002/07/owl#>` und es existiert Turtle Syntax. Dabei bestehen OWL Axiome aus den folgenden drei Grundbausteinen:

19 Prädikatenlogik erster Stufe ist nicht immer entscheidbar, vgl. `http://kilby.stanford.edu/~rvg/154/handouts/fol.html`

- Classes (vergleichbar mit RDFS-Klassen)
- Individuals (vergleichbar mit Instanzen in RDFS)
- Properties (vergleichbar mit RDFS-Properties)

Es gibt zwei vordefinierte Klassen, `owl:Thing`, welche alle Individuen enthält und `owl:Nothing`, welche als leere Klasse fungiert, in allen Klassen enthalten ist, aber selbst keine Unterklassen hat. Außerdem gibt es zwei Arten von Properties, Objekt- und Datentyp-Properties. Objekt-Properties haben Klassen und Datentyp-Properties haben Datentypen als Wertebereich (range). Da hier nicht auf alle Einzelheiten der Syntax eingegangen werden kann, folgt ein kurzes Beispiel. Es beschreibt in der TBox ein *Buch* und einen *Schreiber* sowie, was einen *Autor* und ein *Erscheinungsjahr* ausmacht. Die angegebene ABox enthält zwei Individuen.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .

  :Buch       a   owl:Class .
  :Schreiber  a   owl:Class .

  :Autor  a   owl:ObjectProperty ;
          rdfs:domain   :Buch ;
          rdfs:range    :Schreiber .

  :erscheinungsJahr   a   owl:DatatypeProperty ;
                      rdfs:domain   :Buch ;
                      rdfs:range  xsd:integer .

  :JoanneKRowling     a   Schreiber .
  :HarryPotterUndDerOrdenDesPhoenix   a   :Buch;
                                      :autor   :JoanneKRowling;
                                      :erscheinungsJahr   2003 .
  :QuidditchImWandelDerZeit           a   :Buch;
                                      :autor   :JoanneKRowling;
                                      :erscheinungsJahr   2001 .
```

2.3.10 *SPARQL*

Das rekursive Akronym SPARQL steht für SPARQL Protocol and RDF Query Language und ist ein weiterer Standard des W3C (vgl. Aranda u. a., 2013).

Es ist sowohl eine Abfragesprache als auch ein Protokoll über der HTTP-Protokoll-Schicht zum beidseitigen Austausch von Daten zwischen Client und Server. Der Austausch erfolgt dabei oft über eine clientseitige Benutzerschnittstelle (User Interface (UI)), womit über das SPARQL Protokoll auf den Server zugegriffen wird. Dort befindet sich der SPARQL Endpunkt, in dem RDF Triple in Form einer Graph-Datenbank gespeichert sind. Als Abfragesprache dient SPARQL der Traversierung von RDF Graphen. Das Ausgabeformat ist dabei eine Tabelle im Extensible Markup Language (XML)-Format[20]. Die Syntax ist inspi-

20 das „SPARQL Query XML Results Format“

riert von der Structured Query Language (SQL). Der Aufbau erfolgt durch die drei Grundelemente *SELECT*, *FROM* und *WHERE* sowie einen Abschnitt, um die Ausgabe zu filtern, gruppieren oder anderweitig zu modifizieren. SPARQL basiert auf der Turtle-Serialisierung und dem grundlegenden Mustervergleich auf Graphen (engl. basic graph pattern matching).

Definition 12 ((Basic) Graph Pattern) *Ein Graph Pattern (oder Triple Pattern) ist ein RDF-Triple, was an beliebigen Stellen* Variablen *enthalten kann (also als Subjekt, Property und Objekt). Damit ist ein solches Muster eine Verbindung aus Turtle Termen und Variablen. Ein* Basic Graph Pattern *ist eine Menge an Triple Pattern.*

Variablen beginnen in SPARQL mit einem Fragezeichen. Das folgende vereinfachte Beispiel steht für ein Pattern, welches in einer Abfrage Länder ihren Hauptstädten in Tabellenform gegenüberstellt.

```
?country    dbo:capital     ?capital .
```

Für eine komplette Anfrage (Query) müssen noch verwendete Namespaces definiert werden. Dies funktioniert analog zu Turtle, jedoch ohne dass ein Punkt am Ende gesetzt wird. Es muss das gesamte im *WHERE* Teil definierte Basic Graph Pattern (BGP) konjunktiv erfüllt sein (mit Ausnahme von durch *OPTIONAL* gekennzeichneten Triple Patterns).

Weitere interessante Ansätze bieten die SPARQL Schlüsselwörter *ASK* und *CONSTRUCT*. Mit ASK wird nur auf die Existenz von Ergebnissen ohne Paramater geprüft. CONSTRUCT bietet die Möglichkeit, aus den Ergebnissen strukturierte RDF Triple zu erzeugen, also quasi einen Ergebnis-Graph zu extrahieren.

SPARQL ist sehr komplex und vielseitig nutzbar. Eine Darstellung aller Möglichkeiten ist in dieser Arbeit nicht möglich. Daher soll nun ein kurzes und einfaches Beispiel zum besseren Einblick beitragen und die wichtigsten Konzepte zusammenfassen. Es wird nach den Namen und URIs von Planeten gefragt, welche mehr als 5 Begleiter (Satelliten, Monde) haben. Die Suche erfolgt zunächst nach den Planeten und ihren Klarnamen, anschließend wird nach Planeten mit Begleitern gesucht. Die angezeigten Ergebnisse werden nach der Anzahl der Begleiter (mehr als 5) gefiltert. Die Anzeigen, welche Nichtzahlen als Angabe bei Begleiter ausweisen, werden nicht berücksichtigt.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp:  <http://dbpedia.org/property/>
PREFIX dbo:  <http://dbpedia.org/ontology/>
PREFIX foaf: <http://http://xmlns.com/foaf/0.1/>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>

SELECT ?name ?satellites ?planet
FROM <http://dbpedia.org/resource/>
WHERE
{
  ?planet a dbo:Planet.
  ?planet foaf:name ?name.
  ?planet dbp:satellites ?satellites.
  FILTER ( ?satellites > "5"^^xsd:integer && datatype(?satellites) = xsd:
      integer )
}
ORDER BY DESC( ?satellites )
```

Tabelle 2.2 gibt die Ergebnisse als Tabelle an.

name	satellites	planet
Jupiter	67	http://dbpedia.org/resource/Jupiter
Saturn	62	http://dbpedia.org/resource/Saturn
Uranus	27	http://dbpedia.org/resource/Uranus
Neptune	14	http://dbpedia.org/resource/Neptune

Tabelle 2.2: Die Ergebnisse der SPARQL Query

2.4 GRUNDLAGEN DER IMPLEMENTIERUNG

Zum Abschluss des Grundlagen-Kapitels wird auf einige Grundlagen der Implementierung eingegangen, die im Verlauf dieser Arbeit relevant sind.

2.4.1 *JavaScript*

JavaScript (JS) ist eine im Jahr 1995 von Netscape veröffentlichte *Skriptsprache*. Der ursprüngliche Name *LiveScript* wurde noch im selben Jahr als Folge des Zusammenschlusses mit *Sun Microsystems*[21] in *JavaScript* geändert. Ursprünglich sollte JS zur Erstellung dynamischer Webseiten dienen, um zum Beispiel Nutzerinteraktionen auszuwerten. Der Quellcode wird zur Laufzeit mittels eines *Interpreters* ausgewertet[22]. Der standardisierte Sprachkern ECMAScript[23] kann von allen gängigen Browsern interpretiert werden. Sie kann als Skriptsprache des objektorientierten Programmierparadigmas (spätestens seit ES6) bezeichnet werden. Jedoch kann man mit JS auch prozedural oder funktional programmieren. Die Typisierung erfolgt dynamisch und ist schwach ausgeprägt (Bewersdorff, 2018).

Typische Anwendungsfelder für JavaScript sind zum Beispiel die Anzeige von *Dialogen*, die dynamische Manipulation über das Document Object Model (DOM) oder das Senden und Empfangen von Daten, ohne die Notwendigkeit des Neuladens der Seite (Ajax). Ursprünglich war JS als clientseitige Sprache gedacht. Dieses Konzept wird jedoch in den letzten Jahren immer mehr aufgeweicht, unter anderem durch vorinstallierte und serverseitig kompilierte Pakete (zum Beispiel mittels *NodeJS*).

2.4.2 *ECMAScript 6*

JS ist seit 1997 standardisiert als EcmaScript 1. Es folgten die Versionen zwei bis aktuell neun, seit Version sechs werden die Releases nach dem Jahr ihres Erscheinens benannt[24]. EcmaScript2015 (ES6) oder ECMAScript 2015 ist der

21 Den Erfindern der Programmiersprache *Java*, die jedoch grundverschieden zu JS ist.
22 im Gegensatz zu einem *Compiler*, der den Code als ganzes im maschinenlesbaren Code übersetzt.
23 European Computer Manufacturers Association (ECMA)
24 vgl. `https://www.w3schools.com/js/js_versions.asp`

für SNIK Graph verwendete Standard des JavaScript-Sprachkerns. Es bietet dank *Prototypen* und einer Syntax, die klassenähnliche Konstrukte zulässt, alle Möglichkeiten einer objektorientierten Programmiersprache. Die Einführung beinhaltet viele nützliche Funktionalitäten wie die Datentypen *let* und *const*, *Arrow-Functions* und den *Spread-Operator*[25].

2.4.3 *NodeJS*

NodeJS ist eine Laufzeitumgebung für JS. Sie ermöglicht es, den eigentlich für die clientseitige Interpretation gedachten Code auch serverseitig interpretieren zu können. Dies wird durch zahlreiche sehr nützliche Pakete ermöglicht, die mithilfe des Node Package Manager (npm) installiert und verwaltet werden. Für SNIK Graph wird NodeJS neben npm auch für die Tests und das Linting genutzt. Die frühere Nutzung für Babel und Webpack zur Unterstützung älterer Browser wurde inzwischen aus Performancegründen eingestellt.

2.4.4 *Module*

Alle verbreiteten Programmiersprachen haben ein Modulsystem. Prominente Beispiele dafür sind Java Packages oder die über Header eingebundenen Bibliotheken in C. Solche Systeme dienen dem besseren Überblick und verhindern Namenskonflikte zwischen einzelnen Paketen, Bibliotheken und Variablen. Bei JavaScript gab es solch ein System ursprünglich nicht, da ursprünglich nur kleine Skripte geschrieben wurden. Über die Zeit wurde JS aber immer wichtiger und die Skripte immer umfangreicher. So führten verschiedene Bibliotheken und Frameworks eigene Modulsysteme ein, wie z.B. die Node-Modules oder auch RequireJS[26]. Seit der Einführung von ES6 im Jahr 2005 gibt es Module auch im standardisierten Sprachkern, sie gehören daher nativ zu JS und können inzwischen mit allen gängigen Browsern genutzt werden.

2.4.5 *Cytoscape.js*

Cytoscape[27] ist eine vom Institute for Systems Biology (ISB)[28] entwickelte quelloffene Software, die zur Visualisierung von molekularen Interaktionsnetzwerken und anderen bioinformatischen Zusammenhängen genutzt wird. Die Codebasis ist Java, jedoch gibt es ein Schwesterprojekt (*Cytoscape.js*[29]), welches in einer JS-Umgebung angesiedelt ist. So ist Cytoscape plattformunabhängig (benötigt nur die Java Virtual Machine (JVM)) und kann als Cytoscape.js sogar im Browser verwendet werden. Mit Cytoscape lassen sich Graphen visualisieren und analysieren, es ist gut skalierbar und kann durch Plugins erweitert werden. Das eigene Datenformat *.cys* ermöglicht es, Sitzungen („sessions“) zu speichern und zu laden, was im Verlauf dieser Arbeit wichtig ist. Weitere nützliche Funktionen

25 ES6 Sprachspezifikation: `http://www.ecma-international.org/ecma-262/6.0/`
26 vgl. `https://developer.mozilla.org/de/docs/Web/JavaScript/Guide/Modules`
27 `http://www.cytoscape.org/`
28 `http://www.systemsbiology.org/`
29 `http://js.cytoscape.org/`

sind unter anderem das Mapping der Knotengröße zum Grad oder die Möglichkeit, Sichtbarkeiten der Knoten und Kanten individuell anzupassen. So lassen sich mithilfe dieser Software auch RDF Graphen visualisieren, wie zum Beispiel der SNIK Graph.

2.4.6 *GitHub*

GitHub[30] ist ein Online-Dienst für die Verwaltung und Speicherung von Softwareentwicklungsprojekten. Es basiert auf dem Versionsverwaltungswerkzeug *git*, welches im Jahr 2005 von Linus Torvalds[31] entwickelt wurde. Systeme zur Versionsverwaltung werden eingesetzt, um Änderungen an Dokumenten festzuhalten. Die Dateiänderungen werden mit Nutzerkennung und Zeitstempel versehen und können so zu einem späteren Zeitpunkt nachvollzogen werden. Das besondere an git im Vergleich zu anderen Versionierungswerkzeugen wie Subversion (SVN) ist, dass es auf einem verteilten Konzept aufbaut. Es gibt keinen zentralen Server, so dass die Entwickler auf lokalen Kopien arbeiten, die jeweils die komplette Historie enthalten. Durch die Erstellung von historisch abhängigen Hash-Werten für die Revisionen („commits“) wird ein nachträgliches Abändern der Versionsgeschichte verhindert und so die Sicherheit erhöht. Außerdem gibt es je nach Bedarf und Anlass private und öffentliche Projekte („repositories“). Viele weitere hilfreiche Funktionen ermöglichen einen effizienten Workflow in beliebig großen Entwicklerteams.

GitHub unterstützt und empfiehlt weiterhin die Erstellung von Entwicklungslinien („branches“), die nach Bedarf wieder durch eine Verschmelzung („merge“) auf die Hauptentwicklungslinie gebracht werden können. Die Organisation von Aufgaben mittels *Issue-Tracking* sowie die Bereitstellung von Projektplanungswerkzeugen wie Meilenstein-Verwaltung runden das Paket ab. Für Unternehmen wird eine kostenpflichtige Variante, GitHub Enterprise, angeboten, während für kleine Projekte und Privatpersonen das System gratis genutzt werden kann.

Sowohl das SNIK Graph Projekt als auch die Inhalte dieser Bachelorarbeit sind im Sinne eines Software-Entwicklungsprojektes auf GitHub gehostet und können auf diese Art nahezu ortsunabhängig bearbeitet werden.

30 vgl. `https://github.com/`

31 dem Erfinder von *Linux* und Träger des Millenium Prize, vgl. `http://dbpedia.org/resource/Linus_Torvals`

3

STAND DER FORSCHUNG

3.1 DIE SNIK ONTOLOGIE

Das Semantische Netz des Informationsmanagements im Krankenhaus (SNIK) ist eine Ontologie, die die Domäne des Managements von Informationssystemen im Gesundheitswesen (MIG) beschreibt. Sie ist nach OWL mittels eines eigenen Metamodells (vgl. Abb. 3.1) formalisiert. Ein Metamodell ist ein Modellierungs-Framework. Es besteht aus der Syntax und Semantik der Modellierung sowie aus einer Repräsentation der benutzten Konzepte, z.B. in grafischer Form (vgl. Winter u. a., 2011). Es existieren drei disjunkte Basisklassen:

- Rolle („role“); beschreibt das *Wer*,
- Funktion („function“); beschreibt das, *was getan wird* und
- Objekttyp („entity Type“); beschreibt, *welche Information dafür benötigt* wird.

Dabei wird nach den Prinzipien von LOD anfangs manuell, später halbautomatisch extrahiertes Wissen aus verschiedenen Quellen (vgl. Tabelle 3.1) über Klassenhierarchien aufbereitet und in Beziehung gesetzt (Höffner u. a., 2019).

Zur Beschreibung der Beziehungen zwischen den durch die Quellen gebildeten Subontologien wird das Simple Knowledge Organization System (SKOS)[1] Vokabular benutzt. Dadurch werden Ähnlichkeiten und Gemeinsamkeiten modelliert.

Ontologie	**Präfix**	**Quelle**
`http://www.snik.eu/ontology/meta`	**meta**	alle (Metamodell)
`http://www.snik.eu/ontology/bb`	**bb**	Lehrbuch[2]
`http://www.snik.eu/ontology/ob`	**ob**	Lehrbuch[3]
`http://www.snik.eu/ontology/he`	**he**	Lehrbuch[4]
`http://www.snik.eu/ontology/ciox`	**ciox**	CIO[5]-Interview
`http://www.snik.eu/ontology/it4it`	**it4it**	Standard[6]

Tabelle 3.1: Die Quellen der SNIK Ontologie

1 SKOS ist ein gebräuchliches Datenmodell für die gemeinsame Nutzung und Verknüpfung von Wissensorganisationssystemen über das Semantic Web. (vgl. `https://lov.linkeddata.es/dataset/lov/vocabs/skos`)
2 Winter u. a., 2011
3 Ammenwerth u. a., 2014
4 Heinrich, Riedl und Stelzer, 2014
5 Chief Information Officer
6 The Open Group, 2017

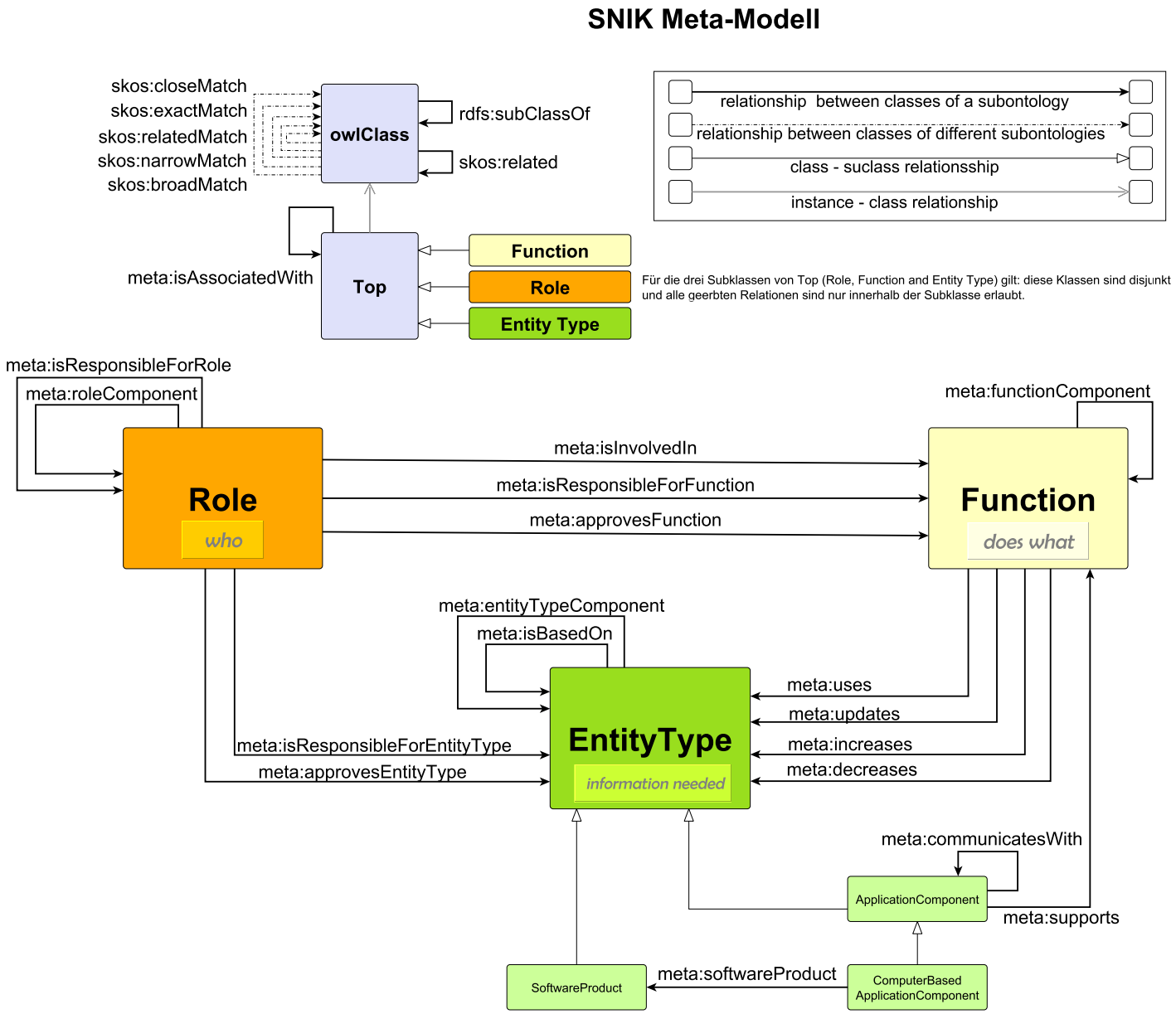

Abbildung 3.1: Das SNIK Metamodell Version 8. Quelle: `https://www.snik.eu/sites/www.snik.eu/files/files/uploads/Ergebnisse/SNIK_Metamodell_V8.pdf`

3.2 SNIK GRAPH

SNIK Graph[7] ist die Visualisierung der Ontologie. Das UI (vgl. Abb. 3.2) ist mithilfe von *CytoscapeJS* implementiert und bietet zahlreiche Funktionen. Die

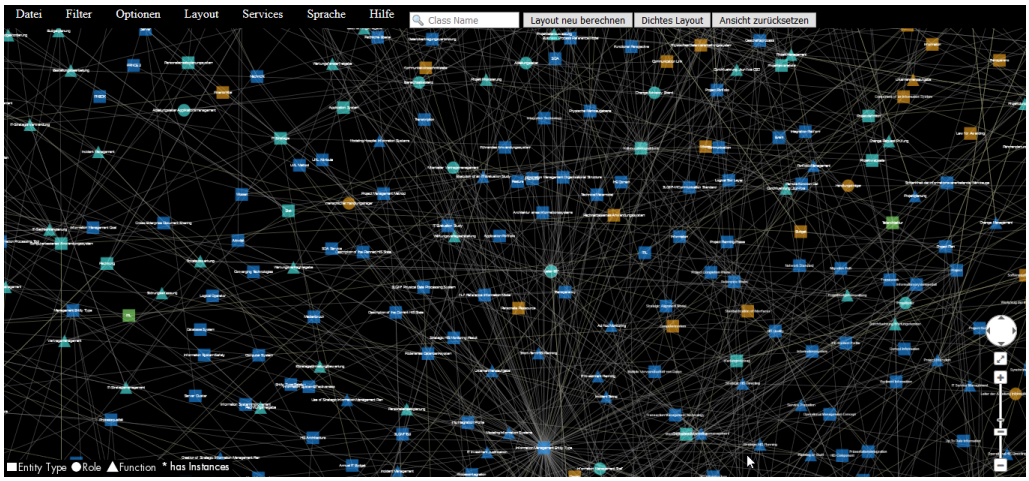

Abbildung 3.2: Die Benutzeroberfläche des SNIK Graph

Objekttypen sind als Quadrate, die Rollen als Kreise und die Funktionen als Dreiecke dargestellt. Die Relationen werden als Kanten visualisiert, die bei höherem Zoomfaktor die Properties als Beschriftung anzeigen. Am rechten unteren Rand befindet sich ein Verschiebe- und Zoom-Werkzeug[8] um durch den Graph navigieren zu können (was allerdings auch durch Maus und Touchpad funktioniert).

3.2.1 *Das Menü*

Das Menü am oberen Rand beinhaltet die Einstellungsmöglichkeiten. Neben *Datei*operationen wie Speichern und Laden gibt es diverse *Filter*, um nur die benötigten Teile der Ontologie anzeigen zu können. Unter *Optionen* finden sich beispielsweise der Tag-Modus und verschiedene Modi für unterschiedliche Nutzergruppen. Dies dient dazu den Funktionsumfang so übersichtlich wie möglich zu halten. *Layout* bietet Varianten zum Anordnen von extrahierten Teilgraphen. Unter *Services* finden sich Links zum SPARQL Endpunkt[9] und dem RDF Browser[10]. Das Menü *Sprache* bietet momentan die Sprachen Englisch, Deutsch und Persisch, wobei letzteres nur rudimentär ausgeprägt ist. Im Bereich *Hilfe* finden sich Tutorials, Links zum YouTube-Channel[11] sowie zum GitHub Repository[12] und Informationen zur Version und zum Bugreporting. Im Suchfeld kann nach Stichworten gesucht werden, die Suche erkennt dabei auch ähnliche Begriffe

7 https://www.snik.eu/graph/
8 das npm Package panzoom; vgl. https://github.com/cytoscape/cytoscape.js-panzoom
9 https://www.snik.eu/sparql/
10 Wir verwenden OntoWiki: https://www.snik.eu/ontology/
11 https://www.youtube.com/channel/UCV8wbTpOdHurbaHqP0sAOng/featured
12 https://github.com/IMISE/snik-cytoscape.js

(„fuzzy search“). Die drei Buttons neben dem Suchfeld bieten Schnellzugriff auf die drei am häufigsten benötigten Layoutoperationen.

3.2.2 *Das Kontextmenü*

Das *Kontextmenü* (Abb. 3.3 zeigt den Standard-Modus) erscheint durch Rechtsklick auf einen beliebigen Knoten oder eine Kante. Es bietet im *Standard-Modus* die

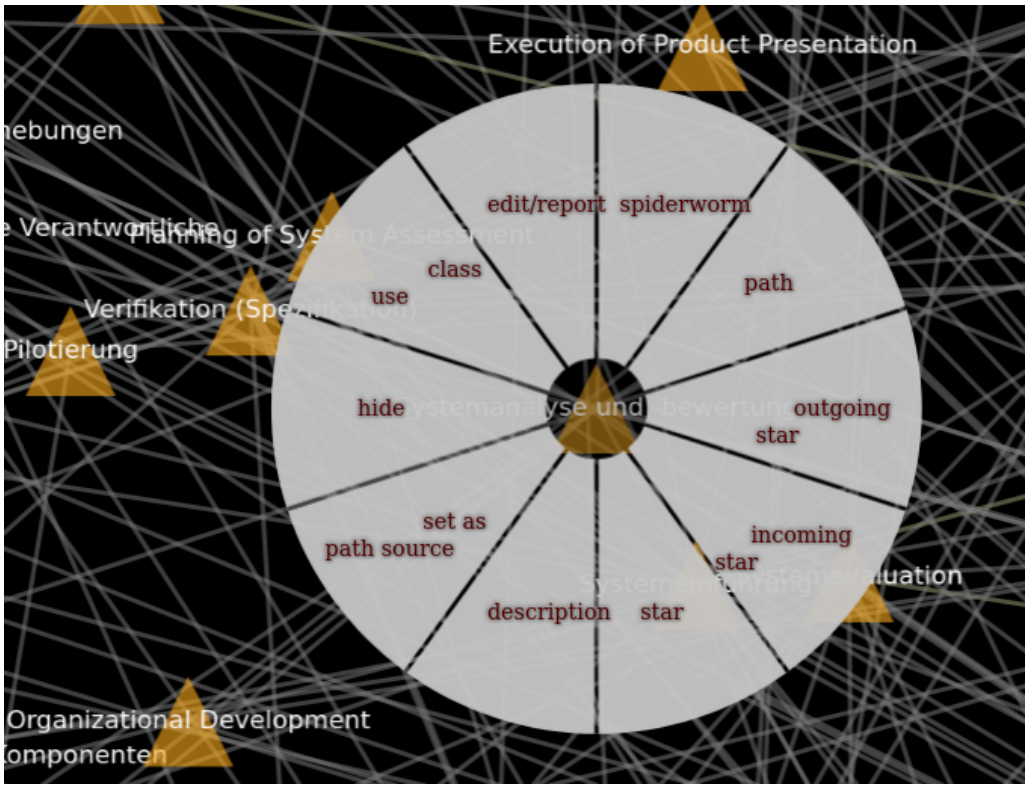

Abbildung 3.3: Das Kontextmenü (Standard-Modus) des SNIK Graph

folgende Auswahl:

- *Description:* Öffnet den Eintrag im RDF Browser.
- *Star:* Hebt den aktuellen Knoten und alle direkt mit ihm verbundenen Knoten hervor.
- *Incoming/Outgoing Star:* Hebt den aktuellen Knoten und alle über eingehende/ausgehende Kanten verbundenen Knoten hervor.
- *Path:* Findet und zeigt den *kürzesten Pfad* von einem zuvor gewählten Startknoten zum aktuellen Knoten.
- *Spiderworm:* Findet ebenfalls den kürzesten Pfad von einem Start zum aktuellen Knoten und zeigt zusätzlich noch alle direkt verbundenen Knoten des aktuellen Knoten an (im Sinne eines Star).

- *Edit/Report:* Hier kann der Nutzer Feedback zu Modellierungsfehlern[13] geben.
- *Combine Close Matches:* Zusammenführen von äquivalenten Klassen verschiedener Subontologien.
- *Class Use:* Visualisierung des Zusammenspiels von Rolle, Funktion und Objekttyp bezogen auf das Metamodell.
- *Hide:* Verstecken des gewählten Elements bis zum Zurücksetzen der Ansicht.
- *Set Path Source:* Setzt den Ausgangspunkt für die Pfadoperationen.
- *Confirm Link:* Eine automatisch generierte[14] Verbindung als korrekt bestätigen.

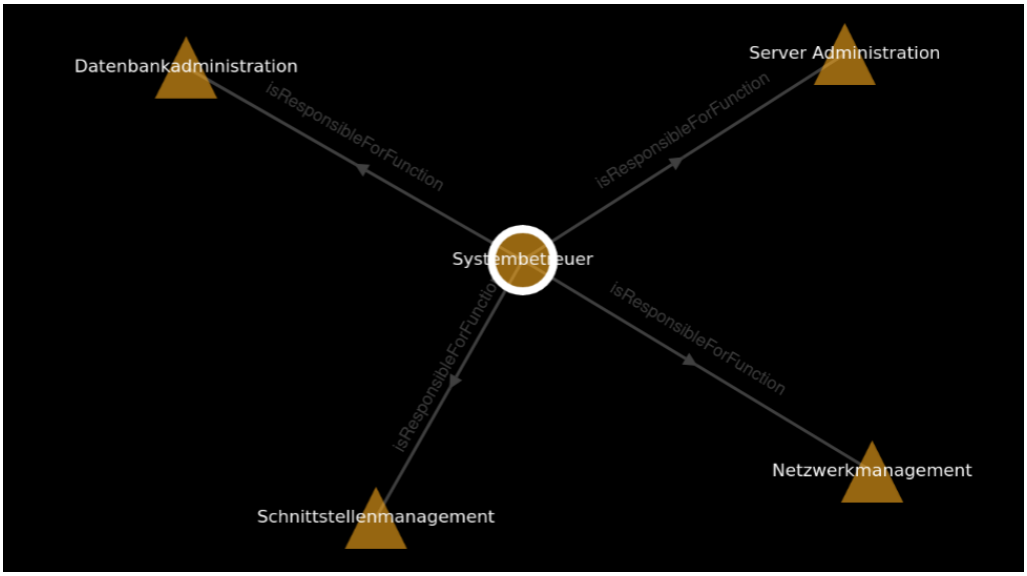

Abbildung 3.4: Star um den Eintrag „Systembetreuer“

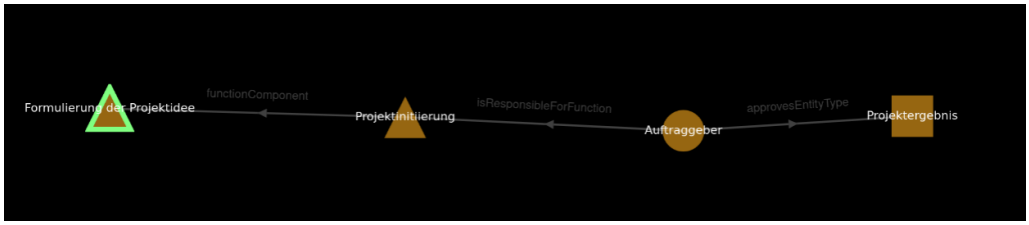

Abbildung 3.5: Pfad von „Formulierung der Projektidee“ zu „Projektergebnis“

Zusätzlich bietet der *Entwickler-Modus* noch folgende Optionen:

- *Remove Permanently:* Über eine GitHub Issue die Löschung eines Elements beantragen. Zusätzlich wird es bis zum Neuladen des Graphs entfernt.

13 über eine GitHub Issue, die dann von den Entwicklern geprüft wird
14 hierfür wird das Framework *Limes* verwendet, vgl. `http://aksw.org/Projects/LIMES.html`

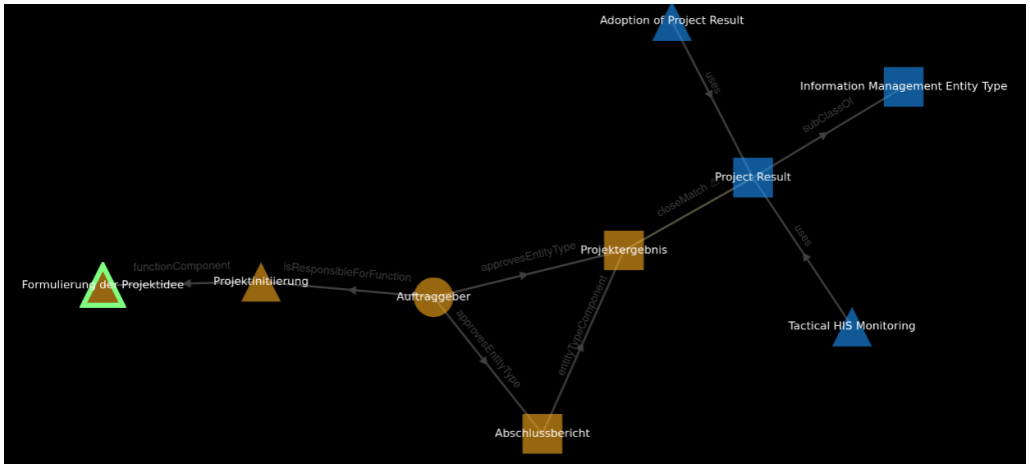

Abbildung 3.6: Spiderworm von „Formulierung der Projektidee“ zu „Projektergebnis“

- *OntoWiki:* Link zum Ontologie Bearbeitungswerkzeug (zugangsbeschränkt).
- *Debug:* Debug-Informationen zum aktuellen Element erhalten (JavaScript Object Notation (JSON)-Datei).

Der *erweiterte Modus* ermöglicht bei aktiviertem Entwickler-Modus vollen Funktionsumfang und hält noch folgende Optionen bereit:

- *Doublestar:* Ähnlich wie der Spiderworm, zusätzlich wird auch am Anfangsknoten ein Star gezeigt.
- *Starpath:* Zeigt einen Pfad und bildet an jedem Knoten auf dem Weg einen Star.
- *Circlestar:* Eine Star Operation, welche die Knoten kreisförmig um den Ausgangspunkt anordnet.
- *LodLive:* Ein exploratives Visualisierungswerkzeug (extern).

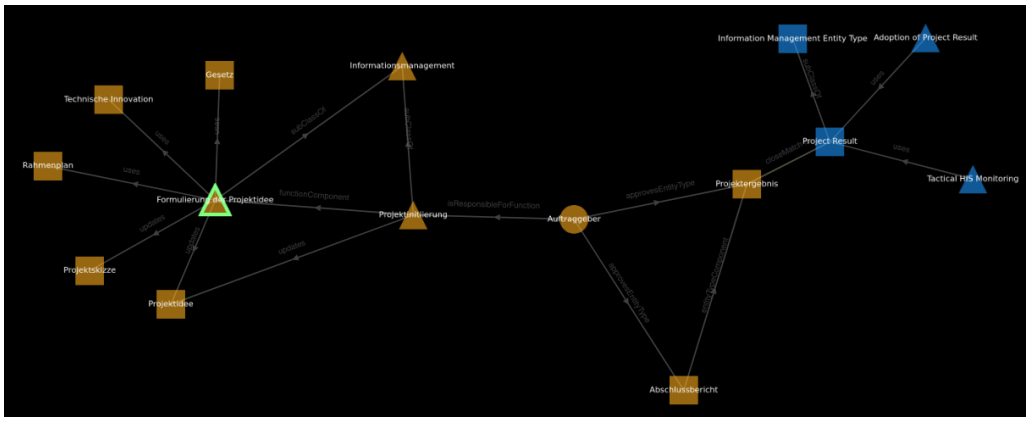

Abbildung 3.7: Doublestar von „Formulierung der Projektidee“ zu „Projektergebnis“

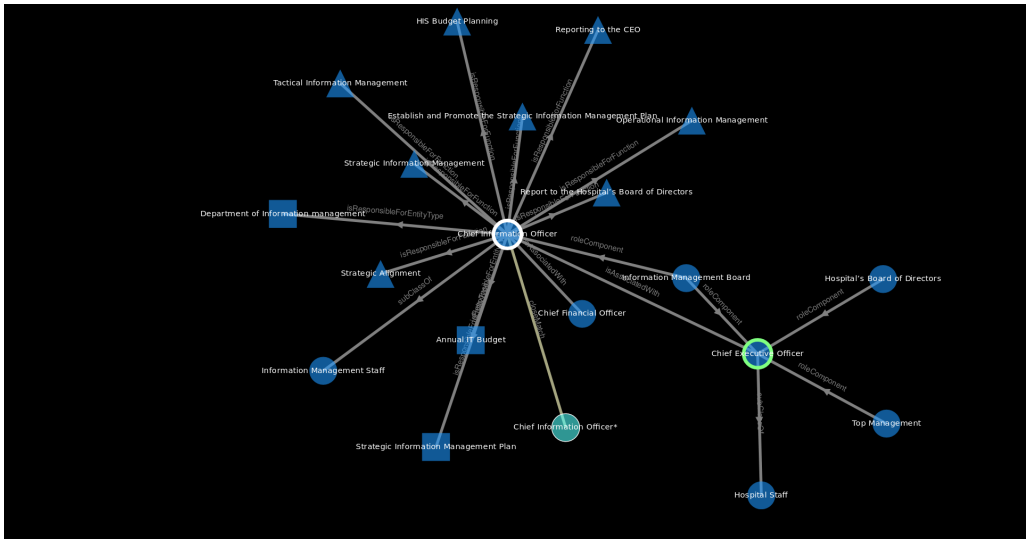

Abbildung 3.8: Starpath von „Chief Information Officer“ bis „Chief Execution Officer“

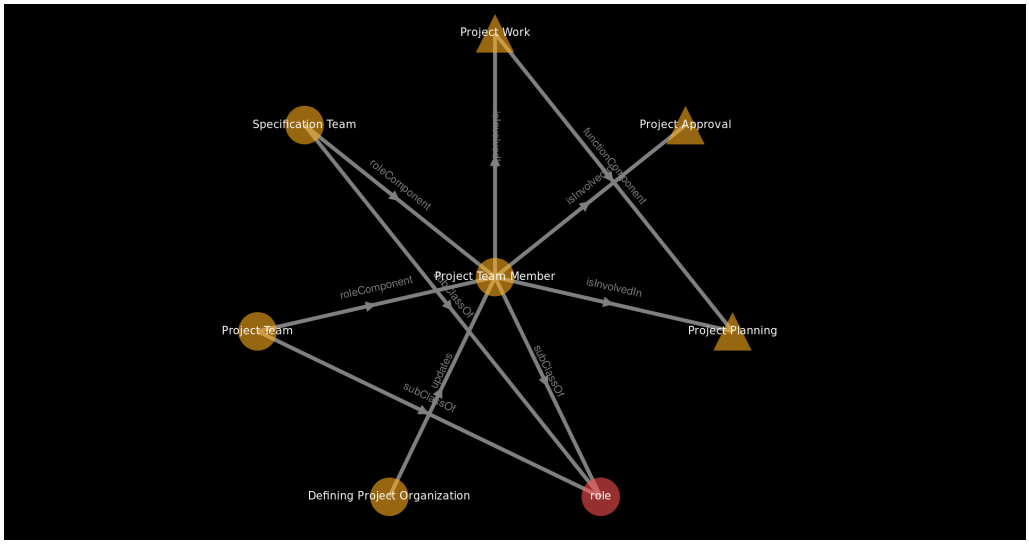

Abbildung 3.9: Circlestar um den Eintrag „Project Team Member“

3.2.3 *Die Code-Struktur*

Da es in JS keine vorgeschriebene Strukturierung für Sourcecode gibt, wird sich bei SNIK Graph an Java Packages angelehnt. Ein Package (hier ein Ordner) gruppiert verwandte Klassen, um besser wartbaren Code zu erhalten und Namenskonflikte zu vermeiden. Beim Download[15] wird der Ordner 'snik-cytoscape.js' erstellt, der den Sourcecode für SNIK Graph enthält. Darin befindet sich die in Tabelle 3.2 beschriebende Ordnerstruktur.

15 über 'git clone git@github.com:IMISE/snik-cytoscape.js.git')

Ordner/Datei	Beschreibung des Inhaltes
css	css-Dateien zur Darstellung von SNIK Graph
docs	quellcodenahe Dokumentation (Generierung mittels npm Skript)
img	Bilder für Nutzerdokumentation
js	JavaScript Code
node	Node Server Code; Generator für das Nutzerhandbuch (nutzt vorhandene Einträge aus Menü und Kontextmenü (js/help.js), um Hilfe und Tooltips zu erstellen)
test	JSDoc Unit Tests (ebenfalls Node Code)
*.html	erzeugte Websites
*.md	Beschreibungen und Checklisten
*.json	Konfigurationsdateien
typecheck Skript	Typescript Typehints werden überprüft

Tabelle 3.2: Ordnerstruktur des Sourcecodes

3.3 DAS KONZEPT DER SINNVOLLEN TEILGRAPHEN

Eine frühere Arbeit zum Thema „Extraktion von Teilgraphen aus der SNIK-Ontologie zur Unterstützung der Lehre“(Anonym, 2019) führt das *Konzept der sinnvollen Teilgraphen* ein. Dafür wurden mit den Dozenten am IMISE Leipzig Workshops durchgeführt und Anwendungsfälle entwickelt, wie die (teil-) automatisierte Extraktion von Teilgraphen aus dem SNIK Graph funktionieren könnte. Die Auswahl der wichtigen Begriffe stellt eine sehr individuelle Entscheidung dar. Dennoch lassen sich einige Muster ableiten, wie ein sinnvoller Teilgraph für den Einsatz in der Lehre aussehen kann. So scheint es nützlich, die Metamodell-Beziehungen auszublenden, da diese für die Studenten eine untergeordnete Rolle spielen. Weiterhin wird eine „maximale Knotenanzahl von 20 Knoten“ und der Wunsch, dass „ein Teilgraph zusammenhängend sein“ soll, ermittelt. Die Ergebnisse der theoretischen Überlegungen aus der zitierten Arbeit sollen als gedankliche Stütze und Vorarbeit für diese Arbeit dienen.

Ein aus dem SNIK Graph extrahierter sinnvoller Teilgraph wird in Abb. 3.10 gezeigt.

3.4 DAS 3LGM^2 TOOL

Das 3-Layer Graph-based Meta-Model (3LGM^2) dient zur Beschreibung, Bewertung und Planung von Informationssystemen im Gesundheitswesen[16]. Es beinhaltet eine Art Baukasten, der basierend auf Unified Modeling Language (UML) Diagrammen die Struktur solcher Systeme darstellt. Die Software kann daher das strategische Informationsmanagement bei der Modellierung von Informati-

16 vgl. https://www.3lgm2.de/

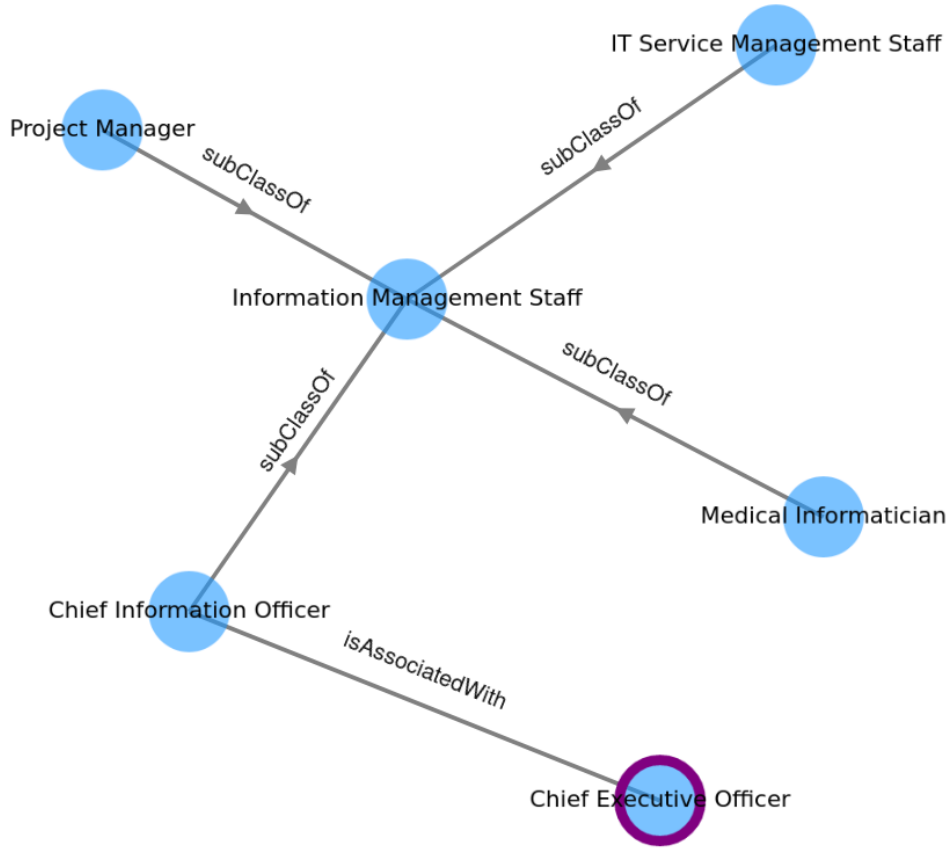

Abbildung 3.10: Ein einfacher RDF Graph. Extrahiert aus dem SNIK-Graph: `https://www.snik.eu/graph/`

onssystemen unterstützen. Das auf Java basierende graphische Werkzeug bietet die Möglichkeit, konforme Modelle zu erstellen, zu analysieren und für den Einsatz in Projekten Teilmodelle zu extrahieren (Winter u. a., 2007 sowie Stäubert u. a., 2015). Dabei wird in drei Ebenen unterschieden: die fachliche, die logische und die physische. Aus dem fachlich benötigten Funktionsumfang sind die logische und physische Werkzeugvielfalt und die Verknüpfungen untereinander sowie zu den anderen Ebenen erkennbar. Das Tool bietet sowohl die Ansicht der drei Ebenen mit verschiedenen Einstellungsmöglichkeiten (Zoom, Winkel und Abstand) sowie eine Einzelebenenansicht (vgl. Abb. 3.11 und Abb. 3.12). Es existieren neben einer umfangreichen Dokumentation inklusive verifizierter UML Modelle auch einige praktische Anwendungsfälle (unter anderem Brigl u. a., 2004 und Wendt u. a., 2004).

Insbesondere die Möglichkeit der Extraktion von Teilmodellen ist für diese Arbeit relevant, denn sie erfolgt in einer Art, in der die Beziehungen zum Gesamtmodell (im übertragenen Sinne SNIK Graph) implizit erhalten bleiben und bei Bedarf für die weitere Verarbeitung (die explorative Erweiterung von Teilgraphen) genutzt werden können.

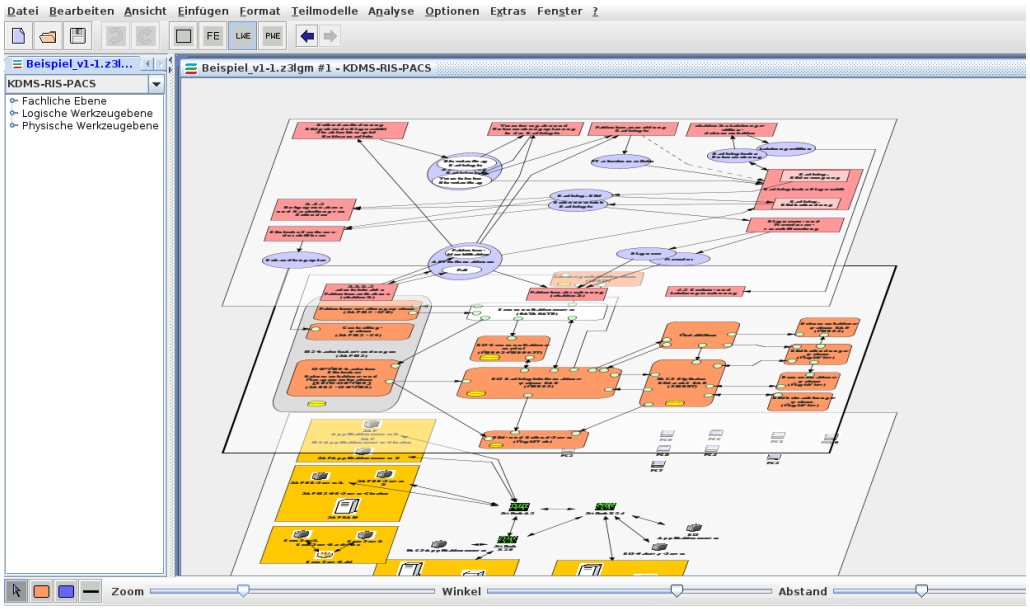

Abbildung 3.11: Die 3-Ebenen-Ansicht im 3LGM^2 Tool

3.5 LINKED-DATA-VISUALISIERUNGEN

In der Literatur sind zahlreiche Ansätze zu finden, wie LOD visualisiert werden können. Das Buch Po u. a., 2020 fasst die wichtigsten Konzepte der letzten zwei Jahrzehnte zusammen und bringt sie in einen zeitlichen Zusammenhang (vgl. Abb. 3.13). Es wird unterschieden in Browser- und explorative Tools, spezielle Visualisierungen für zum Beispiel mobile Endgeräte, reine Ontologievisualisierungen, graphbasierte sowie Multi-Visualisierungen, die als Mischformen der zuvor genannten zu verstehen sind. Da diese Arbeit die Beschreibung der gesamten Vielfalt dieser Werkzeuge nicht abdecken kann, wird sich hier auf einen zusammenfassenden Überblick zu den *graphbasierten Visualisierungen* beschränkt.

Wie bereits im Grundlagen-Kapitel beschrieben, lassen sich RDF-Daten als gerichtete Graphen beschreiben und somit auch im Layout von durch Kanten (*Properties*) verbundenen Knoten (*Subjekte* und *Objekte*) darstellen. Nutzer können so visuell durch die Daten steuern und Erkenntnisse gewinnen. Auf der Grundlage eines SPARQL Endpunktes oder einer Datendatei (N-Triples, Turtle oder ähnliches) bieten generische Visualisierungswerkzeuge interaktive Funktionalitäten wie Zoom und Filter, Schlüsselwortsuche und das Editieren von Knoten und Kanten. Beispiele sind *IsaViz* (Pietriga, 2002), *RDF-Gravity*[17], *Fenfire* (Hastrup, Cyganiak und Bojars, 2008), *LODmilla* (Micsik, Turbucz und Györök, 2014 sowie Micsik, Tóth und Turbucz, 2014), *GIG* (Graziosi u. a., 2018) und *LOG (Linked Open Graph)* (Bellini, Nesi und Venturi, 2014). Zusätzlich nutzt *Aemoo* (Nuzzolese u. a., 2013) externe Quellen wie Google News, Wikipedia und Twitter zur Bereicherung der Visualisierung mit zusätzlichen Informationen. *Tarsier* (Viola u. a., 2018) bietet eine interaktive 3D-Darstellung zur Visualisierung von LOD.

Andere Tools analysieren vorrangig die Assoziationen, also die Properties, zwischen einzelnen Entitäten. Das webbasierte Werkzeug *RelFinder* (Heim, Loh-

17 http://semweb.salzburgresearch.at/apps/rdf-gravity

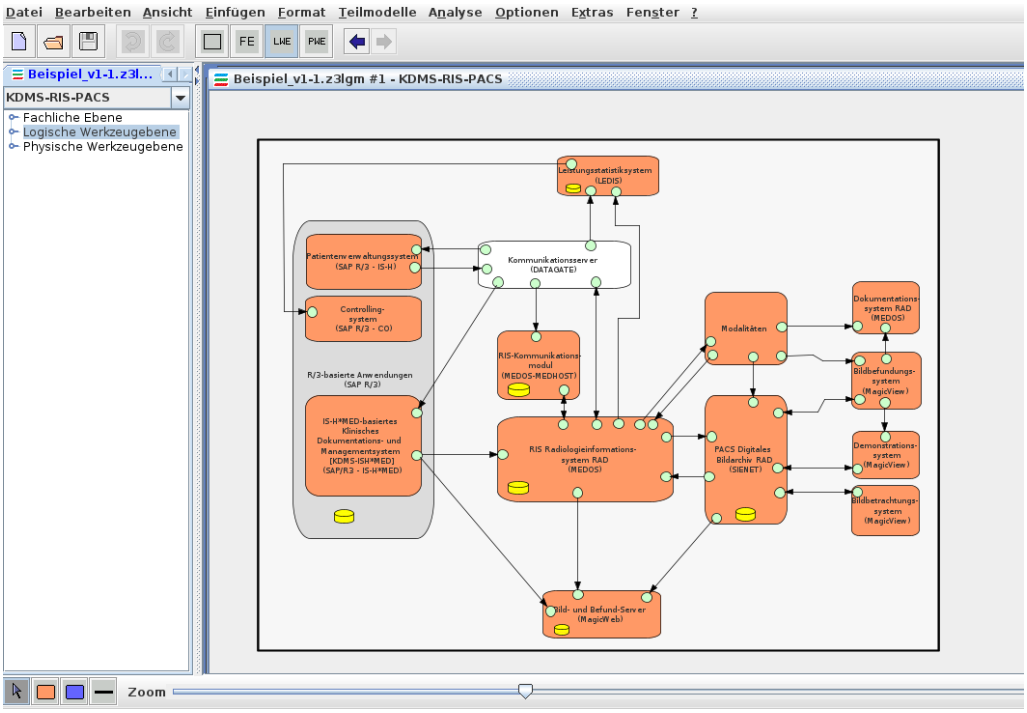

Abbildung 3.12: Die logische Ebene am Beispielmodell im 3LGM^2 Tool

mann und Stegemann, 2010) bietet Möglichkeiten zur interaktiven Exploration und Darstellung von Beziehungen zwischen ausgewählten Ressourcen. Weiterhin versuchen andere Ansätze, besonders repräsentative, also „wichtige“ bzw. „sinnvolle“ Pfade darzustellen oder Pfadzusammenfassungen zu erstellen. *Explass* (Cheng, Zhang und Qu, 2014) arbeitet mit Pfadmustern und *RelClus* (Zhang, Cheng und Qu, 2014) verwendet zur Erzeugung von Pfadhierarchien vorhandene Klassen- und Eigenschaftshierarchien. Um die relevantesten Pfade zwischen zwei Ressourcen anzuzeigen, nutzt *WiSP (Weighted Shortest Paths for RDF Graphs)* (Tartari und Hogan, 2018) Algorithmen für gewichtete kürzeste Pfade. Dabei werden die Gewichte anhand mehrerer Metriken (wie z.B. PageRank und Knotengrad) ermittelt.

Mit *LODVader (LODVisualization, Analytics and DiscovERy)* (Baron Neto u. a., 2016) können verknüpfte Datensätze nach dem in der LODCloud verwendeten Layout visualisiert werden. Dabei sind die Knoten die Datensätze und die Kanten die Verbindungen zwischen ihnen. Das Tool verwendet eine Indexstruktur und ein Datenbanksystem, welches eine effiziente Analyse der Verbindungen und der Ähnlichkeit der Datensätze ermöglicht.

Das Paradigma der inkrementellen Visualisierung beginnt statt mit dem gesamten Graphen mit einem durch zum Beispiel Textsuche bestimmten Ausgangspunkt (einem Knoten). Im Anschluss wird basierend auf Nutzerinteraktion ein immer größer werdender Bereich des Graphen dargestellt. Beispiele sind das bereits erwähnte *Fenfire* (Hastrup, Cyganiak und Bojars, 2008) sowie die Web-Explorationswerkzeuge *LodLive* (Camarda, Mazzini und Antonuccio, 2012) und *LodView*[18], die auch für SNIK verwendet werden. Weiterhin sind die Systeme *RDF*

18 https://lodview.it

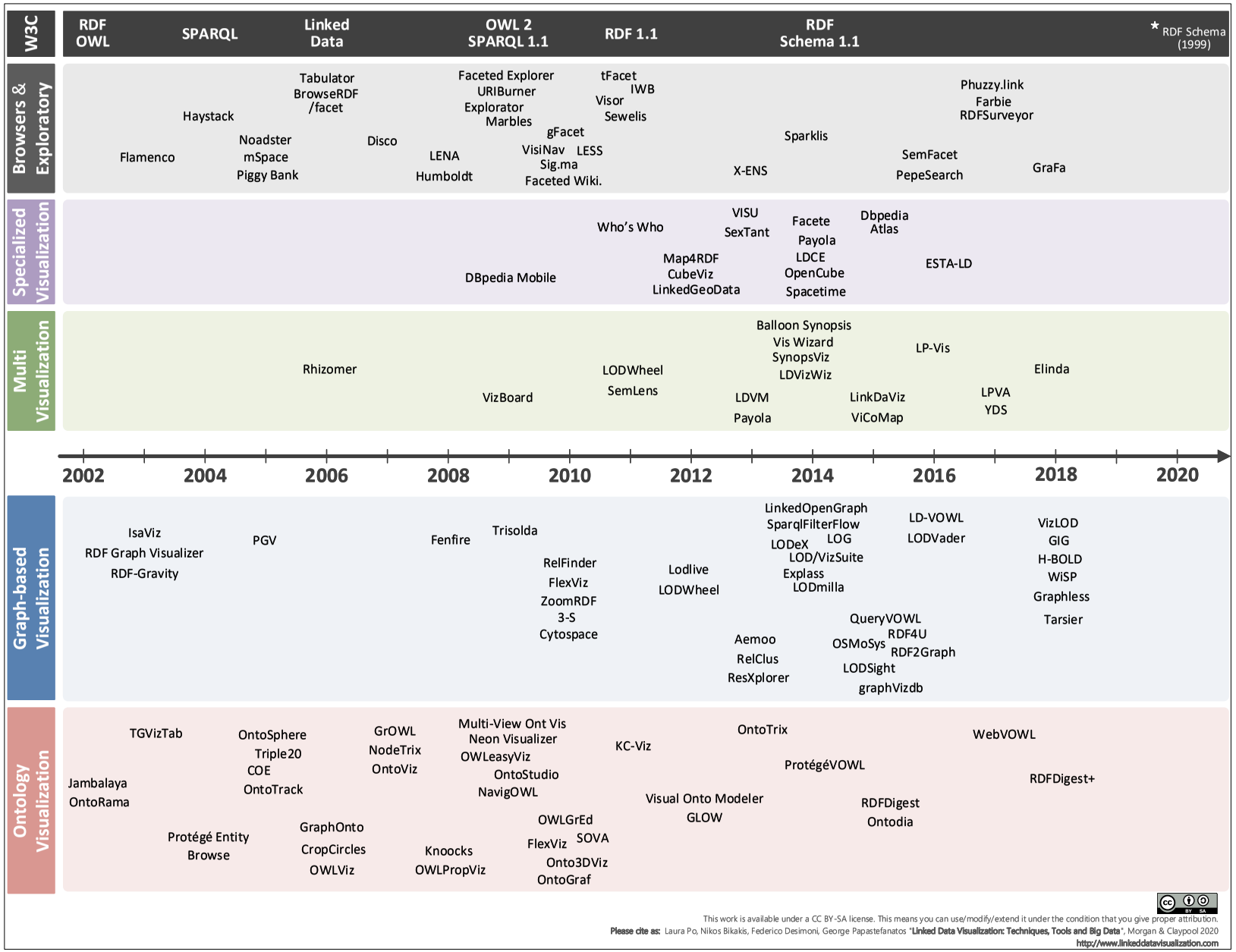

Abbildung 3.13: Linked Data Visualisierung im Verlauf der Zeit. Quelle: Morgan & Claypool, 2020, siehe Po u. a., 2020, Link: `http://linkeddatavisualization.com/resources/Data.Visualization.Tools.Timeline.pdf`

Graph Visualizer (Sayers, 2004), *ResXplorer* (De Vocht u. a., 2013) und die *LOD/Viz-Suite* (De Vocht u. a., 2015) zu nennen, die in ähnlicher Weise knotenzentrierte Exploration inkrementell zulassen.

Die Multi-Level-Graph-Visualisierung bietet visuelle Exploration auf verschiedenen Ebenen mit unterschiedlichen Detaillierungsgraden und verwendet beispielsweise Clusteringtechniken, um Graphknoten für die bessere Übersicht zu verschmelzen. Dadurch können Tools wie *OSMoSys* (Psyllidis, 2015), *Trisolda* (Dokulil und Katreniakova, 2009) oder *graphVizdb* (Bikakis u. a., 2015 sowie Bikakis u. a., 2016) auch mit großen RDF Graphen umgehen. Durch die verschiedenen interaktiven Operationen wie Zoom, Filter, Stichwortsuche und Teilgraphenauswahl in Kombination mit oben erwähnten Multi-Level-Techniken wird dem bei großen Graphen wichtigen Problem des Overplotting, also der zu starken Überlagerung von Graphelementen und deren Beschriftungen, entgegen gewirkt.

Ein weiterer Ansatz diesem Problem zu begegnen sind durch Datenreduktionstechniken erstellte Graphzusammenfassungen, sogenannte „zusammengefasste Visualisierungen". Das für das Graphvisualiserungstool *Cytospace* (Shannon u. a., 2003) erstellte Plugin *3-S* (Sundara u. a., 2010) verwendet Teilmengen, Zusammenfassungen und Stichproben von RDF Daten. *LODeX* (Benedetti, Bergamaschi und Po, 2014) erstellt eine repräsentative Zusammenfassung einer Linked Data Quelle. Mit einem SPARQL Endpunkt als Input generiert es eine von statistischen, strukturellen und Scheminformationen angereicherte visuelle Zusammenfassung der Quelle. Eine Erweiterung von LODeX ist *H-BOLD (High-Level visualization over Big Open Linked Data)* (Po und Malvezzi, 2018b sowie (Po und Malvezzi, 2018a)). Es bietet inkrementelle Mehrebenen-Exploration, bei der ein Detektionsalgorithmus verwendet wird, um die abstrakten Ebenen effektiv zu konstruieren. Auch *RDF4U* (Chawuthai und Takeda, 2016) ermöglicht die Visualisierung zusammengefasster Graphen und verwendet dazu ontologische Argumentation, um redundante Triple-Folgen auf der Grundlage von Eigenschaftstypen zu entfernen. Das System *Graphless* (Santana-Pérez, 2018) generiert Zusammenfassungen auf der Grundlage statistischer Dateninformationen wie z.B. Konnektivitätsgrad der Knoten und Häufigkeit der Eigenschaften. Im Tool *LODSight* (Dudáš, Svátek und Mynarz, 2015) wird anfangs eine Zusammenfassung des Schemas als Graph visualisiert. Anschließend kann der Nutzer über die Klassen navigieren und die Instanzen abrufen.

Unter Verwendung von Techniken zum Zeichnen von Graphen konzentrieren sich einige Werkzeuge auf die Bereitstellung eines effektiven Graph-Layouts. *ZoomRDF* (Zhang u. a., 2010) verwendet einen raumoptimierenden Algorithmus, um die Anzahl der angezeigten Ressourcen zu erhöhen. *PGV (Paged Graph Visualization)* (Deligiannidis, Kochut und Sheth, 2007) nutzt einen Ferris-Wheel[19]-Ansatz zur Darstellung von Knoten mit hohem Grad.

Das noch junge Forschungsgebiet der Schemaextraktion im Zusammenhang mit LOD befasst sich mit der Verarbeitung der Daten, um daraus auf das Ontologieschema schließen zu können. *VizLOD* (Anutariya und Dangol, 2018), *LD-VOWL* (Weise, Lohmann und Haag, 2016a sowie Weise, Lohmann und Haag, 2016b) und *RDF2Graph* (Dam u. a., 2015) nutzen SPARQL Queries zur Verarbeitung von RDF Triples um daraus Schemainformationen abzuleiten. Dabei

19 ein zirkuläres Layout; einem Riesenrad (engl. ferris wheel) nachempfunden

werden zunächst die repräsentativsten Konzepte identifiziert, z.B. werden Klassen mit mehr Instanzen als besonders repräsentativ betrachtet. Anschließend wird das Ontologieschema progressiv als Graph visualisiert, unter Verwendung verschiedener interaktiver Operationen.

Das Konzept der graphbasierten visuellen Abfragekomposition soll Nutzern[20] ohne Vorkenntnisse der Semantic Web Technologien ermöglichen, auf einfache Weise SPARQL Queries unter Verwendung visueller Diagrammdarstellungen zu generieren. Das Web-Tool *SparqlFilterFlow* (Haag, Lohmann und Ertl, 2014) basiert auf einem Filter/Fluss-Modell (Haag u. a., 2014). Die Abfragen werden mithilfe von grafischen Elementen in einer baumbasierten Visualisierung formuliert. In ähnlicher Weise verwendet der Nutzer in *QueryVOWL* (Haag u. a., 2015a sowie Haag u. a., 2015b) visuelle Elemente, die auf grafischen VOWL-Elementen basieren (Lohmann u. a., 2016), um Graphen zu konstruieren, die in SPARQL Queries transformiert werden.

3.6 MULTIVIEW-WERKZEUGE

Multiview-Werkzeuge sind auch im Bereich des WWW inzwischen zahlreich vorhanden und verbreitet. Das während der Corona-Zeit populärste Beispiel ist das *ARCGIS-Dashboard*[21], welches die globale Verteilung, Verläufe und Daten zur Ausbreitung des Virus' übersichtlich darstellt (vgl. Abb. 3.14). Es wurde

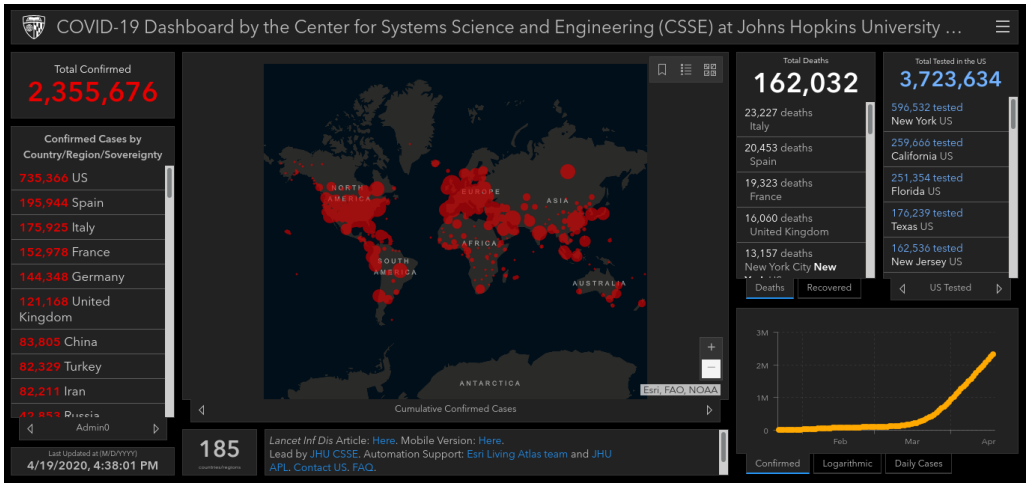

Abbildung 3.14: Das ARCGIS-Dashboard. Quelle: `https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd49423467b48e9ecf6`

am Center for Systems Science and Engineering (CSSE) der John-Hopkins Universität in Baltimore (Maryland) entwickelt und kann auch mit anderen Daten genutzt werden. Dabei können die einzelnen Bereiche separat vergrößert werden und stehen im Austausch miteinander. Zum Beispiel führt die Auswahl eines Landes in der Liste oder auf der Karte mittels der jeweils bekannten Daten zu

20 In dieser Arbeit sind Bezeichnungen von Personen immer in generischer und damit geschlechtsunabhängiger Form zu verstehen.

21 ARCGIS ist eine Sammlung proprietärer Geoinformationssoftwaresysteme der Firma ESRI; vgl. `https://www.esri.de/de-de/home`

Infizierten- und Todeszahlen. Dieser Funktionsumfang ist zur Anwendung nicht ausreichend. Ferner steht das Werkzeug nicht zur freien Verfügung.

Ein wichtiger Punkt für diese Arbeit ist es, zwischen den einzelnen Fenstern kommunizieren und bidirektional Daten austauschen zu können. Weiterhin muss es möglich sein, von einem Startpunkt aus über mehrere Fensterebenen hinweg Pfade des (Teil-)Graphen erkunden zu können. Dafür werden bislang unter anderem auf *jQuery*[22] basierende Frameworks verwendet, in denen Daten und Ansichten gekapselt und weitergegeben werden. Diese container-artige Verwaltung der Daten wird von *PhosphorJS*[23], *dockspawn*[24], *jQueryLayout*[25] und *ExtJS*[26] genutzt. Weiterhin gibt es noch die *Isotope*[27] Library, die sich unter anderem des *Masonry*[28] Layouts bedient. Dabei werden neue Fenster, wie die Steine in einer Mauer, nach dem vertikal vorhandenen Platz gruppiert und ausgerichtet.

Bei der Recherche und ersten Test-Implementierungen erwies sich das Framework *Golden Layout*[29] als vielversprechend. Es nutzt als Ansatz den seit der Antike von Euklid von Alexandria als universelles Designgesetz formulierten *goldenen Schnitt* (vgl. Abb. 3.15) zur Optimierung der Anordnung der Anzeigeflächen. Der Goldene Schnitt (lateinisch sectio aurea, proportio divina) Φ (manchmal auch τ) einer Strecke ist wie folgt definiert[30]:

$$\Phi = \frac{a}{b} = \frac{a+b}{a} \approx 1,618$$

Das Verhältnis des Ganzen einer Strecke zu seinem größeren Teil (dem *Major*, hier als a bezeichnet) soll stets dem Verhältnis des Major zum kleineren Teil (auch *Minor*, hier b) entsprechen. Das Ergebnis ist eine irrationale Zahl. Das Konzept ist auch auf andere geometrische Objekte erweiterbar. Neben der großen Bedeutung dieses Verhältnisses in Mathematik, Architektur und Kunst findet es sich auch in der Natur wieder, beispielsweise in der Anordnung von Blütenblättern verschiedener Pflanzen oder bei den Körperproportionen des Menschen. Übertragen auf den Anwendungsfall der Multiview-Werkzeuge wird dadurch eine optimale Platznutzung und automatische Anpassung auch an unterschiedliche Bildschirmgrößen und -aufösungen erreicht.

22 vgl. `https://jquery.com/`
23 vgl. `https://phosphorjs.github.io/`
24 vgl. `https://node-projects.github.io/dock-spawn-ts/`
25 vgl. `http://layout.jquery-dev.com/`
26 vgl. `https://www.extjs.com/`
27 vgl. `https://isotope.metafizzy.co/`
28 vgl. `https://masonry.desandro.com/`
29 offizielle Webseite mit Live-Demo: `http://golden-layout.com/`
30 vgl.: `https://glossar.item24.com/glossarindex/artikel/item/goldener-schnitt.html`

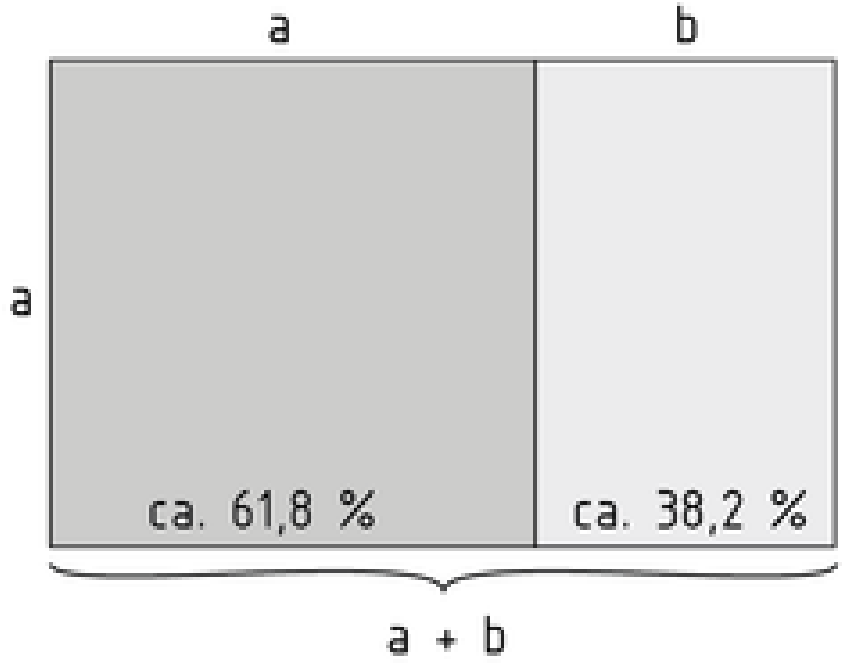

Abbildung 3.15: Das Verhältnis des goldenen Schnittes. Quelle: https://www.toushenne.de/files/robertweller/img/2018/goldener-schnitt-teilung.png

4 LÖSUNGSANSATZ

Dieses Kapitel soll den Weg zur Lösung der genannten Probleme beschreiben. Die Erläuterung der Ausführung erfolgt im nächsten Kapitel. Anschließend werden die erreichten Resultate dargestellt und im Kapitel sieben kritisch diskutiert.

4.1 LÖSUNGSANSATZ FÜR PROBLEM P1

Das erste Problem befasst sich mit der Arbeit an mehr als einem Teilgraphen. Daher soll das vorhandene System SNIK Graph mithilfe einer Layout-Manager-Bibliothek erweitert werden.

Der Ablauf orientiert sich dabei an Ammenwerth u. a., 2014. Nach der im letzten Kapitel erfolgten Systemanalyse und -bewertung folgen die Systemspezifikation und -auswahl sowie die Einführung der Systemerweiterung. Hierbei ist speziell der technische Aufbau von SNIK Graph zu beachten und in die Problemlösung zu integrieren. Ein wichtiger Unterschied zu dem in Ammenwerth u. a., 2014 beschriebenen Ablauf liegt in der Systemeinführung. Diese wird eher als Installation und Nutzbarmachung des neuen Systems beschrieben, wobei bereits vorhandene Komponenten genutzt werden. Hier wird ein völlig neuer Teil der Anwendung implementiert und zur Verfügung gestellt (Deployment). Zukünftige Nutzer werden insofern geschult, als dass Beispielszenarien und eine Anleitung zur Verfügung gestellt werden. Die Beschreibung der einzuführenden Erweiterung wird mittels Wireframes[1] dargestellt. Es folgt die Beschreibung der technischen Umsetzung der Aufgaben A1.1.2 (die Integration der Erweiterung in SNIK Graph), A1.2.1 (die Implementierung der Austauschfunktionen zwischen den Fenstern) sowie A1.3.1 (das Isolieren der Positions- und Sichtbarkeitseigenschaften innerhalb der Fenster).

4.2 LÖSUNGSANSATZ FÜR PROBLEM P2

Das Problem P2 betrachtet den hohen Zeitaufwand für die Extraktion von Teilgraphen aus SNIK Graph.

Zunächst wird die technische Umsetzung der Anpassung in der Suchfunktion (Aufgaben A2.1.1 und A2.1.2) beschrieben. Im Anschluss werden den Lehrkräften für die Lehrveranstaltung „Einführung in die medizinische Informatik und das taktische Informationsmanagement im Krankenhaus“ (auf Basis von Ammenwerth u. a., 2014) an die Hand gegeben. Dabei soll für drei in Vorlesungen behandelte Teilgebiete jeweils ein Teil des behandelten Wissensausschnitts in Form von Teilgraphen übersichtlich und verständlich dargestellt werden. Diese in den drei Szenarien extrahierten Teilgraphen werden als JSON-Dateien und als Bilder bereitgestellt.

1 Wireframes sind Bildschirmentwürfe, die einen Eindruck der zu gestaltenden Beutzeroberfläche geben sollen.

Zum Abschluss wird eine Anleitung entwickelt, wie zukünftig effizienter Teilgraphen extrahiert und mittels der neuen Erweiterung organisiert werden können. Hierzu werden Ablaufdiagramme entwickelt und vorgestellt. Diese orientieren sich an Business Process Model and Notation (BPMN) 2.0[2]. Mittels Beispielszenarien zur Organisation von extrahierten Teilgraphen und einer Anwendungsdokumentation werden die zukünftigen Nutzer geschult.

2 https://www.omg.org/spec/BPMN/2.0/

5 AUSFÜHRUNG DER LÖSUNG

In diesem Kapitel wird die im vorherigen Kapitel angedeutete Methodik ausgeführt und die Ergebnisse entwickelt. Die Gliederung orientiert sich hierbei an Ammenwerth u. a., 2014 mit Ausnahme der in diesem Fall etwas angepassten Systemeinführung.

5.1 SYSTEMSPEZIFIKATION

Zur Spezifikation der Anforderungen an die Layoutbibliothek zur Erweiterung von SNIK Graph wurden Gedanken und Meinungen aus verschiedenen Gesprächen erfragt. Diese wurden mit Prof. Dr. Alfred Winter, mit Sebastian Stäubert und Alexander Strübing (Gemeinsamkeiten zu 3LGM^2) sowie Dr. Birgit Schneider (Nutzerin von SNIK Graph für die Durchführung von SNIK Bingo (siehe Abschnitt 1.1.1) geführt. Auch wurden die Ideen von Masterstudenten aus dem Architekturmodul ausgewertet und persönliche Nutzererfahrungen eingearbeitet. Die Resultate dieser Erhebungen werden in Tabelle 5.1 dargestellt. Die Anforderungen wurden mit der Arbeitsgruppe um Prof. Dr. Winter diskutiert und Anmerkungen in der letzten Spalte vermerkt.

Die für die Auswahl relevanten Punkte sind 1, 5, 8 und 9, die restlichen Punkte adressieren die vorhandene Implementierung von SNIK Graph und müssen neu überdacht und gegebenenfalls angepasst werden. Es wird eine möglichst performante Multiview-Bibliothek gesucht, die sich als ES6-Modul integrieren lässt.

5.2 SYSTEMAUSWAHL

Aus der im Kapitel 3 beschriebenen Recherche zu vorhandenen Bibliotheken, die die benötigte Funktionalität bereitstellen und den erhobenen Anforderungen, wird *GoldenLayout* ausgewählt. Die Gründe liegen dabei hauptsächlich in der im Vergleich zu den anderen betrachteten Multiview-Werkzeugen sehr detaillierten Dokumentation. Des weiteren lässt sich die Bibliothek als acES6-Modul einbinden. Zur Vorbereitung der Funktionalität wird mit Wireframes auf der Basisanzeige von GoldenLayout gearbeitet, die im Rahmen des Bachelorseminars vorgestellt und zur Diskussion gestellt wurden (Abb. 5.1 bis Abb. 5.3).

5.3 IMPLEMENTIERUNG

Die Einbindung von GoldenLayout erfolgt unter den technischen Voraussetzungen von SNIK Graph und CytoscapeJS. Das bedeutet insbesondere die Implementierung als ES6-Modul in purem JavaScript. Zur strukturierten Bearbeitung wird mit GitHub-Issues gearbeitet, die mittels der dafür definierten Meilensteine

Nr.	Anforderung	adressiert	Bemerkung
1	Möglichkeit, in mehreren Instanzen von SNIK Graph arbeiten zu können	Multiview-Bibliothek	
2	inkrementelle Exploration	SNIK Graph	Start von einem Knoten, daraus Entwicklung von Teilmodellen
3	Knotengrad bestimmt Größe des Knotens	SNIK Graph	ist bereits implementiert, wird jedoch aktuell nicht benötigt
4	Undo-Funktion, um Änderungen rückgängig machen zu können	SNIK Graph	ohne Backend sehr schwer umsetzbar
5	Möglichkeit, Sessions und Teilgraphen speichern zu können	Multiview-Bibliothek	
6	intuitivere Menüführung	SNIK Graph	
7	Grid-Funktion zur einfacheren Organisation von Teilgraphen	SNIK Graph	
8	Anreicherung mit zusätzlichen Informationen (z.B. die Einblendung des *LodView* Eintrages des betrachteten Objekts)	Multiview-Bibliothek	momentan nicht gewünscht bzw. benötigt
9	vergleichbare Performance gegenüber dem Zustand vor der Implementierung der Erweiterung von SNIK Graph	Multiview-Bibliothek	

Tabelle 5.1: Die Anforderungen an die Erweiterung von SNIK Graph

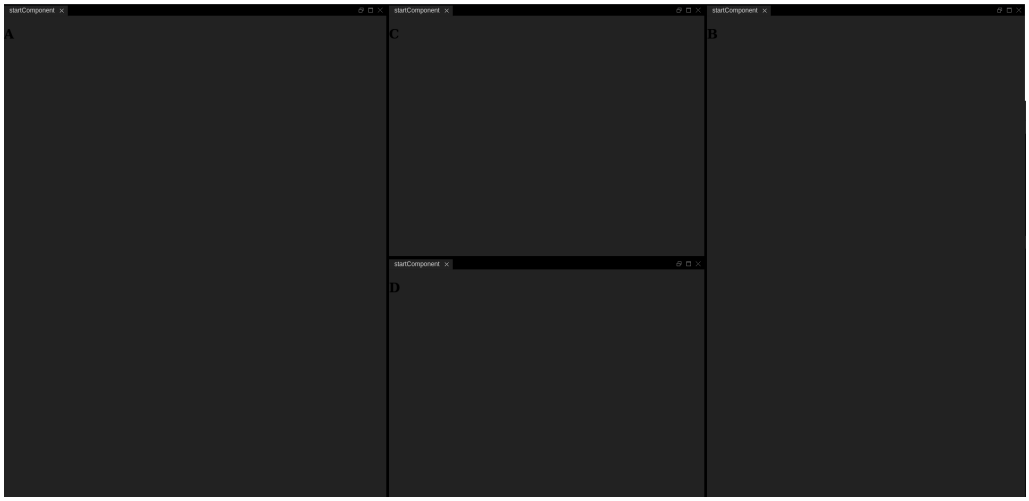

Abbildung 5.1: Benutzeroberfläche mit GoldenLayout

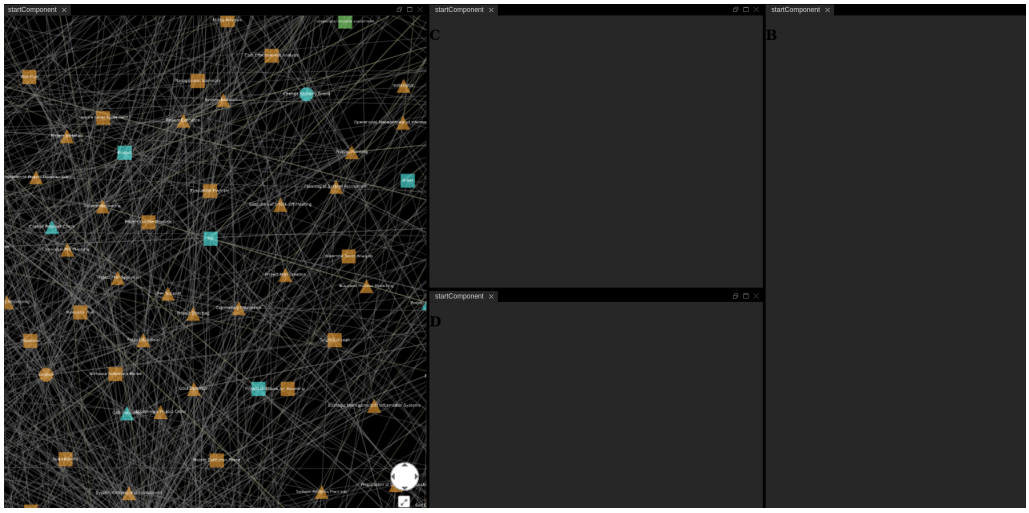

Abbildung 5.2: Der komplette Graph als Ausgangspunkt

tp-must[1] und *tp-optional*[2] klassifiziert werden. Im folgenden wird nach einem kurzen Überblick zum Framework GoldenLayout die Einbindung in SNIK Graph anhand der wichtigsten Gesichtspunkte beschrieben.

GRUNDSÄTZLICHER AUFBAU VON GOLDENLAYOUT Zum Verständnis der Implementierungsbeschreibung wird zunächst der prinzipielle Aufbau von GoldenLayout dargestellt. Die Einbindung des npm-Modules erfolgt in der Hauptseite *index.html* durch Laden der Dependencies für JS und Cascading Style Sheets (CSS) sowie von jQuery[3], was zur DOM-Manipulation und -Navigation benötigt wird. Danach erfolgt die Konfiguration des initialen Layouts. Dieses kann dann später beliebig vom Nutzer verändert werden. GoldenLayout unterscheidet dabei zwischen den Strukturen *Reihe (row)*, *Spalte (column)* und einer Anordnung mittels

1 https://github.com/IMISE/snik-cytoscape.js/issues?q=milestone%3Atp-must
2 https://github.com/IMISE/snik-cytoscape.js/issues?q=milestone%3Atp-optional
3 https://jquery.com/

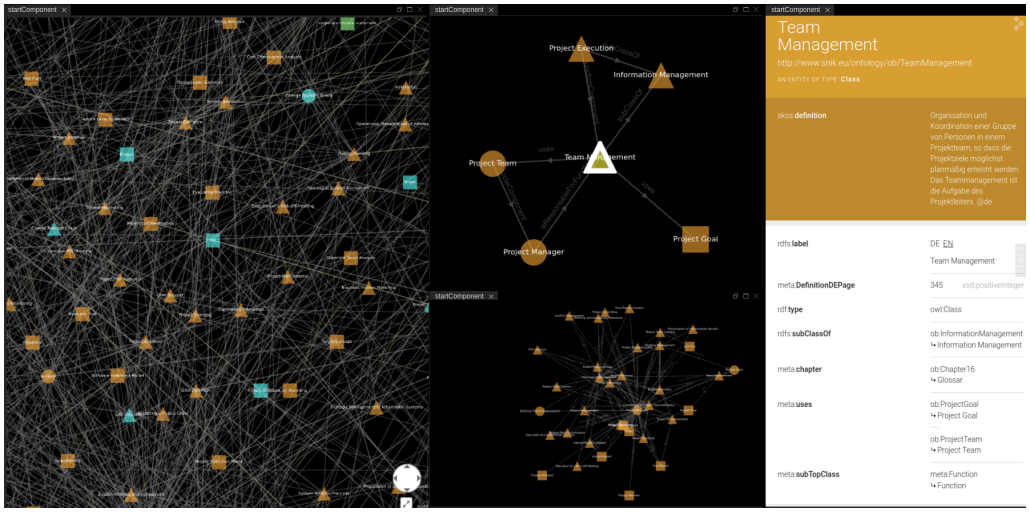

Abbildung 5.3: Die Arbeit mit mehreren Teilgraphen

Registerkarten (stack). Sie dienen als Container für die *Komponenten (components)*, die den Inhalt darstellen. Die Konfiguration wird in einem JSON-Objekt hinterlegt, welches auch den *Inhalt (content)* enthält. Die Einträge heißen *content items*. Für die Implementierung wird die stack-Struktur als Standardkonfiguration ausgewählt. Anfangs soll es nur eine Registerkarte (*Tab*) geben, die den gesamten Graph umfasst. Der Nutzer kann zu einem späteren Zeitpunkt neue Tabs anlegen, aber auch beliebig in Spalten und Reihen organisieren.

EINBINDUNG IN SNIK GRAPH Die Einbindung in SNIK Graph erfolgt anhand zweier neuer Klassen, *view.js* und *viewLayout.js*. View.js enthält einen Konstruktor, der einen neuen View (also eine Anzeigeinstanz als JS-Objekt) erzeugt und mit Titel und Inhalt versieht. Im Objekt wird auch festgelegt, dass wenn es sich um das Erste im Layout handelt, dass dieses über die Benutzeroberfläche nicht geschlossen werden kann. Der Inhalt ist dabei entweder, für den Fall das es keine weitere Instanz gibt, der komplette SNIK Graph oder eine Kopie des Graphen, bei der alle Knoten und Kanten auf nicht sichtbar gestellt sind. Dies wird in der Funktion *fill(graph)* realisiert. Weiterhin gibt es noch die Funktion *reset()*, welche die durch GoldenLayout erzeugte Baumstrtuktur zurücksetzt, um ein neues Layout aufbauen zu können. Dies wird wichtig beim Laden von Sessions. Die Klasse view.js wird komplettiert durch zwei Methoden, die den Zustand (*activeState()*) sowie die View-Instanz selbst (*activeView()*) des jeweils aktuellen Views zurückgeben.

Die Klasse viewLayout.js steuert die Gestaltung des neuen Layouts. Hier werden sowohl die Anfangsstruktur des Layouts (stack) festgelegt als auch der *Header* erzeugt. Der rechte Bereich des Headers (siehe Abb. 5.4) beinhaltet diverse selbst definierte Buttons, deren Funktionen in Tabelle 5.2 beschrieben werden.

Abbildung 5.4: Die Schaltflächen im Header

Symbol	Funktion	Beschreibung
Wirbel	Recalculate layout	berechnet das Layout des angezeigten Graphen neu
Pfeile	Tight layout	die angezeigten Knoten werden nah beieinander angezeigt
Clusterstruktur	Compound layout	die Knoten werden nach Zusammenhang gruppiert
Pfeil aus Quadrat	Open new tab	ein neuer View (Tab) wird geöffnet
Plus	Zoom in	Vergrößerung der Anzeige
Minus	Zoom out	Verkleinerung der Anzeige

Tabelle 5.2: Erklärung der Header-Schaltflächen (von links nach rechts)

WICHTIGE ANPASSUNGEN AN SNIK GRAPH Das vorhandene Zoomelement Panzoom[4] wird durch eine platzsparende Alternative ersetzt. Diese besteht aus zwei Schaltelementen im Header des neuen Layouts (das Plus und das Minus).

Die bislang genutzten Buttons für die wichtigsten Layoutfunktionen wurden ebenfalls in den Header ausgelagert. Damit wird das Menü übersichtlich strukturiert.

Zur besseren Bedienbarkeit wird mit Tooltips gearbeitet, die angezeigt werden, sobald der Mauszeiger über einem entsprechenden Symbol ruht. Dafür wird weiterhin das Framework Tippy.js[5] genutzt.

Alle neuen Features werden zusätzlich zu den englischen Bezeichnern auch ins Deutsche übersetzt und sind über die entsprechende Option im Menüpunkt „Language“ änderbar. Diese Anpassungen zur Laufzeit werden ebenfalls in allen Views umgesetzt.

Um der Anforderung an eine bessere Organisation von Teilgraphen Rechnung zu tragen, wird eine neue Option implementiert, die ein Koordionatensystem auf die Zeichenfläche projiziert. An diesem können die Elemente der Teilgraphen einfacher ausgerichtet werden. Diese Möglichkeit wird über den Menüpunkt Optionen ⇒ Raster anzeigen aufgerufen und wird auf alle Views gleichzeitig angewandt.

4 https://github.com/cytoscape/cytoscape.js-panzoom
5 https://atomiks.github.io/tippyjs/

Zur Personalisierung und besseren Übersicht können die Teilmodelle umbenannt werden. Die vergebenen Namen bleiben auch beim Speichern und Laden erhalten. Zur Umbenennung wird ein Prompt benutzt, welches beim Auswählen von Layout ⇒ Titel des aktuellen Views ändern die Eingabe des Nutzers speichert und zur Verarbeitung weitergibt.

Die Frage, wieviele und welche Modelle zu Beginn, also beim Aufrufen der Seite, gezeigt werden, muss für den Entwicklungs- und den Produktivbetrieb unterschiedlich beantwortet werden. In Abstimmung mit dem Betreuer und Hauptentwickler von SNIK Graph, Konrad Höffner, soll im Produktivbetrieb lediglich ein Tab mit dem Gesamtmodell angezeigt werden. Für die Entwicklung sind mindestens zwei Tabs zu empfehlen, um neue Features schneller testen zu können. Daher wird in der Datei *config.dist.js* ein neues Feld *multiview* mit dem Parameter *initialTabs* definiert, der die Anzahl der initialen Tabs zunächst auf eins setzt. Beim Einrichten der Code-Basis zur Entwicklung und auf dem Server wird diese Datei in eine lokale Datei *config.js* kopiert, in der die Anzahl dann nach eigenem Bedarf angepasst werden kann. So ist zunächst definiert, dass es produktiv nur einen View geben soll, jedoch kann der Entwickler jederzeit problemlos adjustieren ohne darauf einen Einfluss zu nehmen.

IMPLEMENTIERUNG DER AUSTAUSCHFUNKTIONEN Die Austauschfunktionen legen fest, wie Elemente aus dem Gesamtmodell in neue Teilmodelle gelangen. Diese Funktionen werden kaskadierend angelegt. Wenn aus einem Teilmodell eine Menge von Knoten ausgewählt wird, sollen diese auch in weitere Teilmodelle transferiert werden.

Dafür müssen zunächst einige Vorbereitungen getroffen werden. Beim Erzeugen eines neuen Views wird eine Kopie des Gesamtmodells initiiert und alle Elemente versteckt. Die Idee ist, über die IDs die zu kopierenden Elemente (Knoten und Kanten) zu identifizieren und beim Einfügen deren Sichtbartkeit zu aktivieren. Dafür muss zunächst der aktive Tab und der aktive Stack, in dem er sich befindet identifizert werden, damit diese gezielt angesprochen werden können. Durch Markieren eines Teiles des Graphen oder die Auswahl über die Suche wird die zu kopierende Teilmenge bestimmt. Beim Kopieren mittels der Tasten C oder S wird diese Menge in eine Variable *clipboard* gespeichert. Das Einfügen in einen neuen View erfolgt mit den Tasten P oder V. Iterativ wird für jedes Element in der Variable die Sichtbarkeit auf *visible* gesetzt.

ANPASSUNG DER SUCHE Um die explorative Erstellung von Teilgraphen zu ermöglichen wird die Suchfunktion abgeändert. Momentan werden sichtbare Elemente grün, unsichtbare gelb und Elemente, die grundsätzlich zur Ontologie gehören, aber nicht im Graph enthalten sind, in rot angezeigt. Damit wird eine gute Orientierung zu den nutzbaren Elementen gegeben. Um auch in den Teilmodellen explorativ arbeiten zu können, muss eine weitere Kategorie eingeführt werden. Elemente, also Knoten, die bewusst durch Filter oder manuelles Löschen entfernt wurden, sollen auch gefiltert bleiben. Elemente, die durch das Kopieren eines Teils des Gesamtmodell ausgewählt wurden, sollen allerdings durch Stern- und Pfadoperationen wieder erweitert werden können. Dies soll auch geschehen, wenn sie momentan nicht sichtbar sind. Dafür werden diese Operationen

abgewandelt und in der Suche eine neue orange Kategorie eingeführt, die für Elemente steht, die wieder angezeigt werden können, auch wenn sie aktuell nicht sichtbar sind.

SPEICHERN UND LADEN VON VIEWS UND SESSIONS Damit der Nutzer seine Arbeit teilen und auch zu einem späteren Zeitpunkt weiterarbeiten kann, soll es eine Speicher- sowie Lademöglichkeit geben. Da SNIK Graph auch weiterhin ohne Backend, ausgenommen der SPARQL-Endpunkt, auskommen soll, sind die Optionen hierbei eingeschränkt. Die Umsetzung erfolgt über JSON-Objekte, die lokal gespeichert und in die Anwendung geladen werden können. Es gibt drei verschiedene Arten des Speicherns in der Erweiterung von SNIK Graph. Save the full SNIK Graph speichert die komplette Ontologie in eine Cytoscape JSON Datei. Save currently active view speichert den Teilgraphen im aktuellen View und Save Session speichert alle momentan angezeigten Tabs und Teilgraphen in eine Datei.

Dieses Verhalten impliziert auch eine neue Art des Ladens von Teilmodellen. So können mittels Load Partial Graph into Session einzelne Views mit ihren Teilgraphen in eine gerade aktive Sitzung hineingeladen werden, ohne die Sitzung zu verlieren. Load Session hingegen überschreibt die aktuelle Arbeit und lädt die zuvor gespeicherte Sitzung komplett neu. Damit hat der Nutzer die Möglichkeit, nach unterschiedlichen Bedürfnissen und Ansprüchen zu handeln. Beispielsweise kann ein Nutzer vorbereitete Teilmodelle, zum Beispiel durch das Versenden der JSON-Datei per E-Mail, mit einem anderem Nutzer teilen oder an einem anderen PC weiter bearbeiten. Die komplette Sitzung kann entweder neu geladen oder (mittels einer zweiten Instanz der Webanwendung) einzelne Modelle gespeichert und zu einer anderen Sitzung hinzugefügt werden.

Zur besseren Übersicht und Nutzbarkeit ist es möglich, die Titel der Teilmodelle zu ändern (siehe oben). Diese individuellen Benennungen werden auch in den JSON-Dateien gespeichert und geladen.

DIE UMSTRUKTURIERUNG DES MENÜS Ein wichtiger Punkt ist die Umstrukturierung des Menüs. Viele Punkte sind bisher nur auf die eine globale Canvas angewendet worden und müssen daher nun neu überdacht werden. Ein Großteil dieser zu treffenden Entscheidungen wurde zunächst aufgrund der Urlaubszeit ohne die Arbeitsgruppe getroffen und muss daher in einem separaten Treffen noch einmal neu überdacht und diskutiert werden. In einem Meeting mit der Arbeitsgruppe am 15.09.2020 wurde der aktuelle Stand besprochen und abgestimmt. Die Ergebnisse und Erkenntnisse werden im Kapitel 6 beschrieben und im Kapitel 7 diskutiert.

TESTEN DER NEUEN FUNKTIONALITÄTEN Zu jeder Implementierung gehören auch entsprechende Softwaretests, sei es, um die korrekte Funktionalität zu validieren oder um die Erfüllung der Akzeptanzkriterien nachzuweisen (Spillner u. a., 2014). Durch Tests wird die Qualität der Software überwacht und gesichert. Da es sich bei SNIK Graph und speziell bei der implementierten Erweiterung um clientseitige Funktionalität handelt, ist die Durchführung von Unit-Tests nur sehr beschränkt möglich. Manuelle Tests der Benutzeroberfläche werden

zudem eher rudimentär und unregelmäßig anhand der Releasecheckliste[6] durchgeführt. Daher soll hier ein anderer Weg gegangen werden und die Anwendung anhand von Oberflächentests, oder auch Ende-zu-Ende-Tests (E2E-Tests), auf Stabilität und korrekte Funktionalität geprüft werden. Bislang wurden Tests einzelner Komponenten mittels Mocha[7] und Chai[8] geschrieben und mithilfe eines npm-Skriptes automatisch angesteuert. Die darin verwendete Syntax kann auch für die neuen Tests genutzt werden. Dafür wird das npm-Package Cypress[9] verwendet. Mithilfe dieses Frameworks können auf leicht verständliche und gut lesbare Weise E2E-Tests geschrieben werden. Die Testoberfläche bietet die Möglichkeit, sowohl auf verschiedenen Browsern als auch interaktiv zu testen. Dabei arbeitet Cypress im Hintergrund mit einem Node.js Serverprozess, was das Testen ganzer Webapplikationen in Echtzeit ermöglicht. Dabei wird direkt aus der Anwendung heraus agiert, so dass ein Zugriff auf jedes einzelne Element, sei es aus dem DOM, der Anwendungsinstanz usw. möglich ist. Die mitgelieferte Anwendung Cypress Test Runner bietet organisierten Zugang zu den Testskripten und ermöglicht die Auswahl zwischen allen auf dem Rechner installierten Browsern zur Durchführung der Tests. Jeder Testschritt kann einzeln betrachtet und analysiert werden. Cypress bietet auch weitere Funktionen zum Debugging an und eignet sich durch seine Architektur ferner für den Test Driven Development (TDD) Ansatz.

Um die neuen Funktionalitäten zu testen wird ein Szenario entworfen, was einen möglichst breiten Durchstich bietet. Dabei geht es nicht so sehr um Vollständigkeit als vielmehr um die Einführung des grundlegenden Konzeptes von E2E-Tests von SNIK Graph. Dieses Konzept kann für spätere Arbeiten verwendet und erweitert werden.

5.4 SYSTEMEINFÜHRUNG

Der Ablauf einer Systemeinführung wie in Ammenwerth u. a., 2014 findet für diese Arbeit nur bedingt Anwendung, da es um die Erweiterung eines bereits vorhandenen Systems geht. Dennoch kommen Teile der beschriebenen Methodik zum Einsatz. Die *Umstellung* auf das neue System von SNIK Graph erfolgt nach dem Prinzip der Schlagumstellung. Das bedeutet, dass der volle Funktionsumfang mit Abschluss der Implementierungen und der Bachelorarbeit an einem noch zu bestimmenden Stichtag unter der bekannten URL von SNIK Graph[10] zur Verfügung gestellt wird. Technisch findet die Umstellung über die Zusammenführung der Entwicklungslinie (Branch) *feature-multiview* mit der *master*-Branch auf dem GitHub-Repository statt. Anschließend wird der aktuelle Stand in den entsprechenden Ordner auf dem Institutsserver (bruchtal) kopiert (via git pull). Während der Implementierungsphase und der Erstellung dieser Arbeit wird die Erweiterung bereits im Rahmen eines *Testbetrieb*s verfügbar gemacht. Die

6 https://github.com/IMISE/snik-cytoscape.js/blob/master/releasechecklist.md
7 https://mochajs.org/
8 https://www.chaijs.com/
9 https://www.cypress.io/
10 https://www.snik.eu/graph

Realisierung erfolgt über eine zweite Subdomain[11], die ebenfalls auf dem Server liegt.

Eine Adaptierung des neuen Systems, wie im Buch beschrieben, findet hier keine Anwendung, da die Erweiterung grundsätzlich auf den vorgesehenen Anwendungszweck maßgeschneidert wird. Auf Schulungen muss aufgrund des begrenzten Rahmens dieser Bachelorarbeit verzichtet werden. Um jedoch bestmögliche Anreize zur Nutzung der neuen Erweiterung von SNIK Graph zu geben, wird im folgenden eine Anleitung für Endnutzer entwickelt sowie einige Beispielszenarien vorgestellt

5.4.1 *Entwickeln einer Anleitung*

Benutzer von SNIK Graph sollen mit Hilfe der neuen Erweiterung effizient Teilgraphen extrahieren und organisieren können. Um die Bedienung zu erleichtern, wird der empfohlene Arbeitsablauf hier im Fließtext geschildert. Im Kapitel sechs wird dann ein daraus abgeleitetes BPMN-Modell vorgestellt, welches als Arbeitsvorlage dienen soll.

Zu Beginn, also beim Aufruf der Webapplikation, wird der komplette Graph angezeigt. Nun besteht die Möglichkeit, eine bereits gespeicherte Arbeit über Datei ⇒ Lade gespeicherte Sitzung aufzurufen und damit weiter zu arbeiten. Weiterhin kann auch mit der Entwicklung eines neuen Teilgraphen begonnen werden. Der Nutzer hat die Möglichkeit, entweder durch visuelles Auswählen (z.B. mithilfe der Zoom-Buttons) oder durch die Suche einen Start für seinen Teilgraphen auszuwählen. Hier ergibt sich bereits die Chance für eine inkrementelle Exploration. Dafür wird ein neues Tab geöffnet (durch den Button „Neues Tab öffnen“) und der Startpunkt über die Suche festgelegt. Dieser ist dann das einzige gezeigte Element und kann mit den Stern- und Pfadoperationen beliebig erweitert werden. Bei der Auswahl aus dem gesamten Graphen wird ebenfalls zunächst ein Begriff ausgewählt und mittels der Graph-Operationen bearbeitet. Durch Halten der STRG-Taste und Ziehen mit der Maus kann eine Auswahl markiert und mit der Taste ’C’ oder ’S’ kopiert werden. Das Kopieren wird durch ein grünes Hinweis-Popup am linken unteren Rand bestätigt. Nun wird ein neues Tab geöffnet und mittels der Tasten ’V’ oder ’P’ die Auswahl zur weiteren Bearbeitung eingefügt. Die Bestätigung erfolgt per Popup.

Dieses auch iterativ nutzbare Verfahren stellt die Kernfunktionalität dar. Für weitere Graphoperationen und allgemeine Funktionalitäten von SNIK Graph sei auf das Manual[12] verwiesen. Weiterhin können die verschiedenen Teilgraphen (bzw. Views) beliebig verschoben, gruppiert und geordnet werden. Die Speicherung für eine weitere Verwendung erfolgt über Datei ⇒ Speichere aktuelle Sitzung .

11 https://www.snik.eu/pgraph

12 https://www.snik.eu/graph/manual.html

5.4.2 *Erstellung von praxisnahen Beispielszenarien*

Im Folgenden wird dargestellt, wie die neue Erweiterung von SNIK Graph in der Praxis nutzbar ist. Dafür werden drei Beispielszenarien zur Extraktion und Organisation von Teilgraphen beschrieben, die für die Vorlesung „Einführung in die medizinische Informatik und das taktische Informationsmanagement im Krankenhaus“ genutzt werden können. Diese Veranstaltung behandelt die Inhalte des in dieser Arbeit oft zitierten „orangenen Buches“(Ammenwerth u. a., 2014). Sie findet im Rahmen des Bachelorstudiums der Informatik unter Leitung von Prof. Dr. Alfred Winter im Wintersemester statt. Da der Lehrstoff sehr definitionslastig ist, bietet sich gerade hier die Nutzung von SNIK Graph als Bereicherung des Unterrichts und auch des selbstständigen Lernens an. Die Zusammenhänge sind so sehr gut erkennbar und die Anbindung an LodView ermöglicht die Verfügbarkeit der Definitionen. Die Extraktion der vorgestellten Teilgraphen wird in diesem Kapitel beschrieben, die inhaltliche Vorstellung erfolgt im nächsten Kapitel. Die Teilgraphen stehen als Bilder und als JSON Dateien zur Verfügung.

SZENARIO 1 In diesem einfachen Szenario sollen die Zusammenhänge zwischen *Daten*, *Information* und *Wissen* dargestellt werden. Dafür werden zunächst die Filteroptionen angepasst, sodass nur noch Einträge aus dem orangenen Buch (OB, Ammenwerth u. a., 2014) angezeigt werden. Da es sich um Bezeichnungen in deutscher Sprache handelt, wird noch die Sprache unter [Language] auf [german] gestellt. Diese Filter und Einstellungen werden in allen 3 Szenarien angewandt. Im Anschluss wird über das Suchfeld einer der drei Begriffe eingegeben und mit der Enter-Taste bestätigt. Nach der Auswahl des gewünschten Eintrages wird dieser als markiert angezeigt. Mit einem Rechtsklick wird das Kontextmenü geöffnet und die Funktion [Star] gewählt. Der nun angezeigte Teilgraph wird mit gedrückter STRG-Taste und der Maus markiert und mit der Taste [C] oder [S] kopiert. In einem neu geöffneten Tab kann dieser nun mit [V] oder [P] eingefügt werden. Mit dem Button [Tight Layout] werden die Elemente näher aneinander dargestellt und können nun noch zum Beispiel in Form eines Dreiecks organisiert werden. Zur besseren Identifizierung wird der Tab noch umbenannt in „Daten-Information-Wissen“ und kann nun über [Datei] ⇒ [Speichere aktuellen Teilgraphen] gespeichert und zur späteren Nutzung in eine neue Session geladen werden.

SZENARIO 2 Szenario 2 stellt eine explorative Entwicklung und Organisation von Teilgraphen dar. Dafür soll mit einem leeren View begonnen werden, der über die Schaltfläche zum Öffnen eines neuen Tabs erzeugt wird. Der Startpunkt liegt bei dem Knoten „System“. Er wird über die Suche aufgerufen. Eine Staroperation zeigt alle mit ihm verbundenen Komponenten an und wird mit [Tight Layout] zentriert dargestellt. Dieses Teilmodell soll in „Systeme“ umbenannt und wiederum in ein neues Tab kopiert werden. In dem neuen View soll ein Star auf dem Objekt „Informationssystem“ aufgerufen werden. Die neuen Elemente können nun wiederum anschaulich sortiert und mithilfe der Grid-Funktion organisiert

werden. Der neue Tab wird umbenannt in „System–>Informationssystem“ und die gesamte Session als JSON Datei gespeichert.

SZENARIO 3 Das dritte Szenario soll die Entwicklung eines Teilgraphen beschreiben, der mittels einer Pfadoperation erzeugt wird. Dafür wird der Ablauf eines Projektes gewählt, von der Initiierung zum Abschluss. Nach der Anwendung der Filter wird wieder explorativ entwickelt. Dafür wird zunächst in einem leeren Tab nach „Projektinitiierung“ gesucht und anschließend nach „Projektabschluss“. Die beiden Elemente werden mit Tight Layout bearbeitet und anschließend an den linken und rechten Rand platziert. Anschließend soll „Projektinitiierung“ als Quelle des Pfades definiert werden. Dies funktioniert durch Markieren und über den Punkt Set as path source im Kontextmenü. Der Knoten „Projektabschluss“ soll das Ende des Pfades sein und wird deshalb über den Punkt „Path“ im Kontextmenü ausgewählt. Der neue Mittelpunkt „Projektarbeit“ kann mittels eines Stars erweitert werden. Die gewonnenen Ergebnisse können anschließend um die einzelnen Projektphasen und die daran beteiligten Rollen und Aufgaben ergänzt werden. Dafür wird ein Star auf „Projektdurchführung“ erzeugt und die nicht benötigten Einträge mittels der Taste Entf entfernt. So wird der Pfad eines Projektablaufes sichtbar und kann als Übersicht für Vorlesungen und Übungen genutzt und weiter bearbeitet werden. Das Ergebnis wird in einen neuen View kopiert und in „Projektpfad“ umbenannt.

6 ERGEBNISSE

Dieses Kapitel präsentiert die Ergebnisse der Arbeit und prüft anhand der in Abschnitt 1.3 definierten Ziele, ob diese erreicht werden konnten.

6.1 INTEGRATION DER ERWEITERUNG IN SNIK GRAPH

Der erste Teil beschreibt die Umsetzung der Ziele Z1.1 bis Z1.3 sowie Z2.1. Diese geben den Rahmen für die Umsetzung der Erweiterung von SNIK Graph vor. Neben der eigentlichen Erweiterung des Systems um eine Mutliview-Funktionalität (Z1.1) geht es um die Implementierung der Austauschfunktionen (Z1.2). So soll es möglich sein, Teilgraphen kopieren und in anderen Teilmodellen einfügen zu können. Die einzelnen Modelle sollen weitestgehend entkoppelt sein (Z1.3). Das bedeutet, dass Änderungen an Position und Sichtbarkeit von Elementen keinen Einfluss auf diese Elemente in anderen Teilmodellen haben. Schließlich wird die Suchfunktion angepasst (Z2.1), um Elemente miteinander in Beziehung setzen und explorativ Teilmodelle extrahieren zu können. Diese Funktionalität wurde, wie in Abschnitt 5.3 beschrieben, leicht angepasst.

Aus den im Rahmen der Systemspezifikation und der Ziele dieser Arbeit erhobenen Akzeptanzkriterien wurde die Systemauswahl zugunsten von GoldenLayout abgeleitet. Die Implementierung wurde umfassend im vorherigen Kapitel beschrieben. Hier soll nun geprüft werden, ob und in welchem Umfang die Ziele umgesetzt werden konnten.

Zunächst wurde die Benutzeroberfläche an die neuen Herausforderungen angepasst und funktional angebunden (siehe Abb. 6.1). Es gibt nun die Möglichkeit, an mehreren Graphen zu arbeiten. Initial wird das Gesamtmodell angezeigt. Neue Teilmodelle können angelegt, organisiert und umbenannt werden. Außerdem ist es möglich, einzelne Teilmodelle und ganze Sitzungen zu speichern und zu laden. Dabei ist zu beachten, dass ein Laden eines Teilgraphen die bestehende Sitzung erweitert, während das Laden einer kompletten Sitzung die aktuelle Sitzung überschreibt. Der Nutzer wird darauf durch einen Bestätigungsdialog hingewiesen und hat dadurch die Möglichkeit, den Vorgang abzubrechen. Wie die Arbeit mit mehreren Teilmodellen aussehen kann, wird beispielhaft in Abb. 6.2 gezeigt. Teilgraphen können auch inkrementell exploriert, also von einem definierten Startpunkt aus aufgespannt werden. Die Funktion, den Knotengrad mit der Größe des angezeigten Knotens zu verknüpfen wurde bereits implementiert. Zum aktuellen Zeitpunkt wird diese Funktion nicht gewünscht und daher deaktiviert.

Die Organisation des Menüs wurde verändert und optimiert. Das teilmodellspezifische Menü am rechten oberen Rand im Header wurde mit Tooltips versehen, um für besseres Verständnis und eine intuitivere Menüführung zu sorgen. Ein Beispiel hierfür zeigt Abb. 6.3. Des Weiteren wurden einige Benennungen abgeändert und alle Menüpunkte in die deutsche Sprache übersetzt.

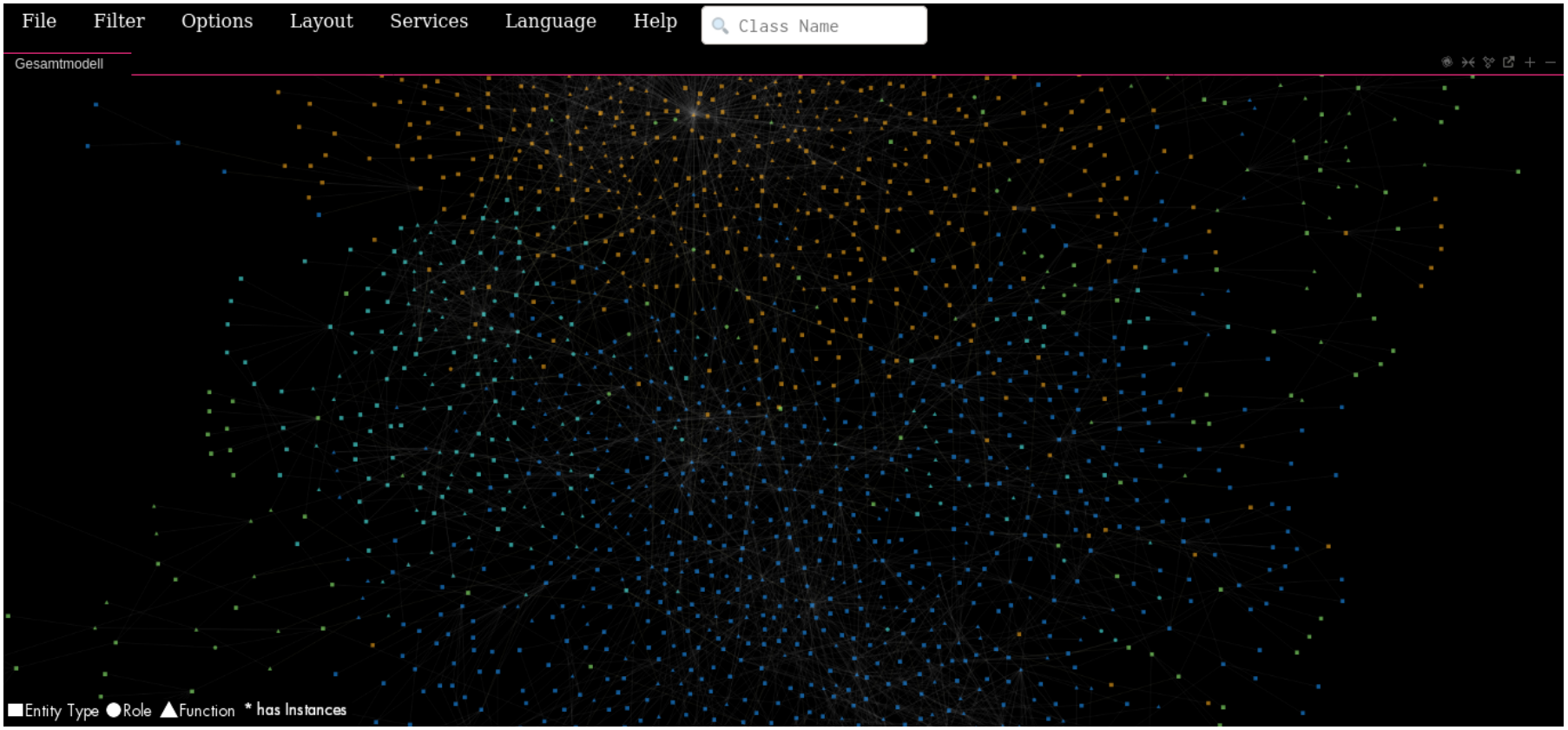

Abbildung 6.1: Die neue Benutzeroberfläche von SNIK Graph

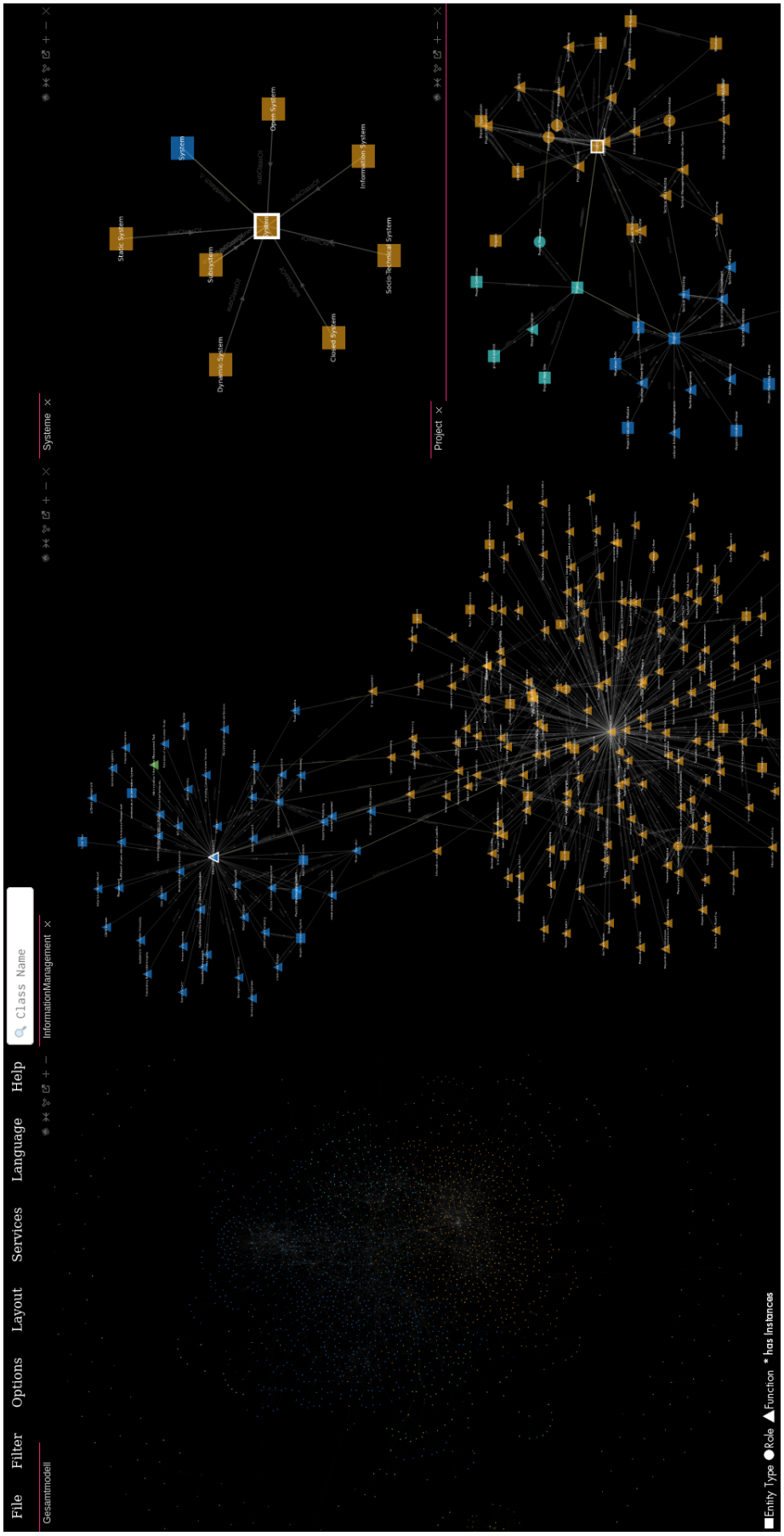

Abbildung 6.2: Die Erweiterung von SNIK Graph in der praktischen Anwendung

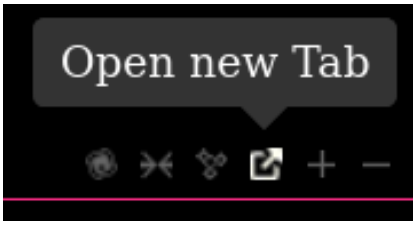

Abbildung 6.3: Ein Tooltip im teilmodellspezifischen Menü

Die gewünschte Gridfunktion wurde implementiert und erleichtert die Organisation und Ausrichtung der Teilgraphen (siehe Abb. 6.4) Sie ist über Optionen

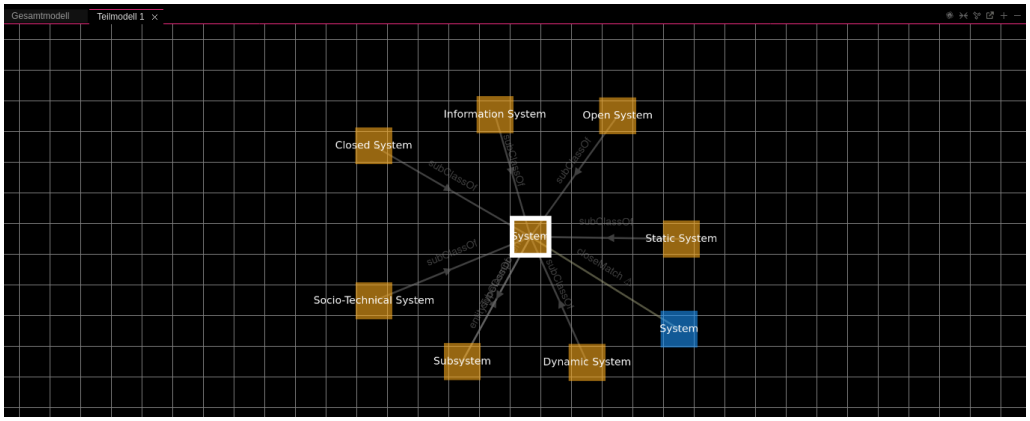

Abbildung 6.4: Die Grid-Funktion

⇒ Grid anzeigen zu erreichen.

Die Anreicherung mit zusätzlichen Informationen durch die Anbindung von LodView im neuen Layout wird momentan nicht gewünscht, kann aber bei Bedarf leicht eingebaut werden.

Zum Vergleich der Performance mit dem Zustand vor der Erweiterung wurde das bereits vorhandene Benchmark-Feature[1] verwendet (siehe Abb. 6.5). Dabei

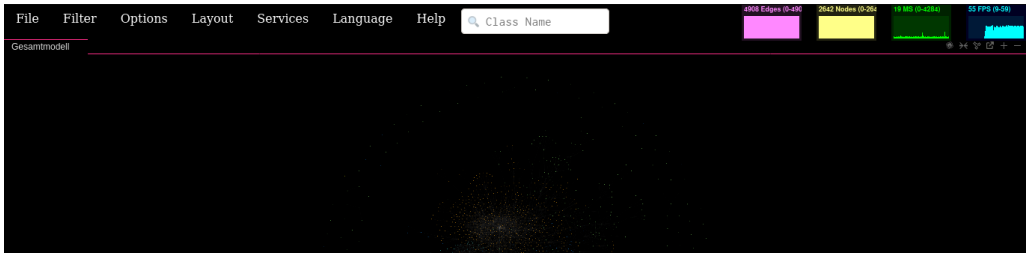

Abbildung 6.5: Die Benchmark-Funktion

werden neben der Anzahl der geladenen und angezeigten Knoten und Kanten auch die Bildfrequenz oder Framerate in FPS (frames per second) und die Millisekunden pro Frame angezeigt und können somit verglichen werden. Dabei fällt auf, dass es bei der Bildfrequenz keine Abweichungen gibt. Die Bildrate in Ruhe ist seitens CytoscapeJS auf 60 FPS begrenzt. Bei Benutzeraktionen, die

1 Aufruf über `https://www.snik.eu/graph/?benchmark`

eine Neuberechnung des Layouts bedingen, liegt die Begrenzung bei 30 FPS[2]. Die Wartezeit beim Laden und Anzeigen von Elementen ist unverändert geblieben. Am längsten dauert das initiale Laden[3], wenn das Layout noch nicht im Cache gespeichert ist. Je nach technischen Gegebenheiten kommt es zu einer kurzen Wartezeit[4]beim Öffnen neuer Views. Weiterhin leidet die Performance bei längeren Sitzungen, was allerdings ein bereits bekanntes Problem ist[5]. Das Ergebnis war, dass die Nutzung mit dem Browser Google Chrome (bzw. Chromium, der freien Variante für Linux) weniger leistungseinschränkend ist als die Nutzung unter Firefox. Daher wird die Nutzung von Google Chrome empfohlen. Für Entwicklungs- und Testzwecke wurde allerdings hauptsächlich mit Firefox Developer Edition gearbeitet. Gründe hierfür sind das gute Entwicklertooling und persönliche Präferenz. Ferner erlaubt Firefox auch ohne Webserver, das heißt lokal, Module vom File Protocol zu laden, wodurch das Testen von neu implementierten (Sub-) Features erheblich erleichtert wird.

Die Ergebnisse der Diskussion bezüglich der Umstrukturierung des Menüs wurden, wie besprochen, umgesetzt. Für die Punkte Filter und Optionen wurde die Minimallösung gewählt. Die nutzerzentrierte Umsetzung und Anwendung könnte in einer fortführenden Arbeit näher untersucht werden. In Tabelle 6.1 sind alle Menüpunkte und ihre jeweiligen Wirkungsbereiche aufgeführt.

Alle neuen Funktionen und ihre Interaktionen mit den bestehenden Funktionalitäten wurden umfassend manuell getestet. Dabei entdeckte Fehler wurden beseitigt und im GitHub-Repository dokumentiert. Weiterhin wurden Oberflächentests mit dem Framework *Cypress* erstellt, die die Navigation durch die Anwendung abbilden und die Funktionalität der Steuerelemente nachweisen.

Die Systemeinführung wurde mit dem Release der Version 2.0.0 am 2. Oktober 2020 durchgeführt. In Abstimmung mit der Arbeitsgruppe um Prof. Dr. Winter wurde die neue Version sowohl auf *graph*[6] als auch auf *pgraph*[7] veröffentlicht. Zusätzlich wurden die unten beschriebenen Szenarien als JSON Dateien mit dem Release verknüpft[8]. Der Sourcecode von SNIK Graph[9] ist dieser Arbeit nicht beigelegt, da es sich dabei nicht um komplett eigene Arbeit handelt. Das Vorgehen wurde mit dem Betreuer dieser Arbeit abgesprochen. Die Unterscheidung kann über die Historie des Sourcecodes getroffen werden. Jedoch sind einzelne Codeabschnitte nicht allein funktionsfähig, da es sich um eine Erweiterung der vorhandenen Implementierung handelt.

Die im Abschnitt 1.3 definierten Ziele wurden erreicht und mit den im Laufe der Ausarbeitung entstandenen und zusätzlich umgesetzten Anforderungen übererfüllt. Die Aufteilung in unabhängige Teilmodelle von SNIK Graph (Ziel Z1.1) sowie die Austauschoperationen zwischen diesen Ansichten (Ziel Z1.2) wurden implementiert. Auch sind Positions- und Sichtbarkeitsänderungen von

2 siehe `https://github.com/cytoscape/cytoscape.js/issues/984`
3 ca. zwölf Sekunden unter Firefox auf Windows mit Intel Core i7-8565U CPU 1.80 GHz und 32 GB RAM
4 ca. drei Sekunden unter Firefox auf Windows mit Intel Core i7-8565U CPU 1.80 GHz und 32 GB RAM
5 siehe `https://github.com/IMISE/snik-cytoscape.js/issues/103`
6 erreichbar unter `https://www.snik.eu/graph/`
7 erreichbar unter `https://www.snik.eu/pgraph/`
8 siehe `https://github.com/IMISE/snik-cytoscape.js/releases/tag/2.0`
9 siehe `https://github.com/IMISE/snik-cytoscape.js`

Menü	Eintrag	Wirkungsbereich
File	Load from SPARQL endpoint	gesamte Anwendung
	Save session	alle Views
	Save currently active view	aktueller View
	Save the full SNIK Graph	Ontologie wird gespeichert
	Recalculate layout and replace in browser cache	nur auf Gesamtmodell
	Save Image of Current View	aktiver View
	Save Image of Whole Graph	Gesamtmodell (ungefiltert)
	Save Image of Current View (high res)	aktiver View
	Save Image of Whole Graph (high res)	Gesamtmodell (ungefiltert)
	Load Partial Graph into Session	aktive Session
	Load Session	aktive Session
Filter	alle	alle Views
Options	alle	alle Views
Layout	Show close matches	aktiver View
	recalculate layout	aktiver View
	tight layout	aktiver View
	compound layout	aktiver View
	move matches on top of each other	aktiver View
	move matches nearby	aktiver View
	BB/OB chapter search	aktiver View
	subontology connectivity	erzeugt neues Tab
	reset view	nur auf Gesamtmodell
	change title of active View	aktiver View
Services	SPARQL Endpoint	Fenster (externer Link)
	RDF Browser	Fenster (externer Link)
Language	alle Einträge	alle Views und Menü
Help	alle Einträge	Fenster (teils externe Links)

Tabelle 6.1: Das aktualisierte Menü

Elementen unabhängig von anderen Views möglich (Ziel Z1.3). Die Beziehung der Elemente über die Suche wurde mithilfe der neuen Suchkategorie umgesetzt. Ferner können die Beziehungen durch die inkrementelle Entwicklung von Teilmodellen nun auch explorativ erkundet werden (Ziel Z2.1).

6.2 UNTERSTÜTZUNG DER LEHRKRÄFTE UND NUTZBARKEIT DER ERWEITERUNG

Der zweite Teil dieses Kapitels setzt sich mit der Umsetzung von Ziel Z2.2 auseinander. Hierbei geht es um die Unterstützung der Nutzer von SNIK Graph. Im Speziellen sollen die Lehrkräfte und Studenten unterstützt und Anwendungsfälle für die Lehrveranstaltung „Einführung in die medizinische Informatik und das taktische Informationsmanagement im Gesundheitswesen“ im Wintersemester 2020/2021 gegeben werden. Die im Rahmen dieses Abschnittes entwickelten Grafiken werden in den Anhang auslagert. So können Sie auch ohne die Lektüre dieses Kapitels leichter gefunden und verwendet werden.

BESCHREIBUNG DES PROZESSMODELLES Zur Unterstützung der Lehrkräfte wird ergänzend zur Anleitung im Fließtext aus dem letzten Kapitel noch eine mithilfe von BPMN 2.0 modellierte grafische Anleitung entwickelt. Diese ist mit Absicht so einfach wie möglich gehalten, da sie nur als Orientierung zur Arbeit mit der Erweiterung von SNIK Graph dienen und damit gut lesbar sein soll. Das Ergebnis der Modellierung wird in Abb. A.1 dargestellt und soll nun kurz erläutert werden.

Es wird der mögliche Ablauf einer Sitzung beschrieben. Das Startevent ist der Sitzungsbeginn, also der Zeitpunkt des Aufrufens der URL der Webanwendung. Initial wird die Benutzeroberfläche mit einem offenen Tab gezeigt, der das Gesamtmodell enthält. An dieser Stelle hat der Benutzer die Wahl, eine bereits gespeicherte Session bzw. einen einzelnen Teilgraphen zu laden oder eine neue Extraktion zu starten. Danach beginnt der iterative Prozess der Extraktion und Organisation von Teilgraphen. Es wird mit einem neuen Tab (Schaltfläche Neuen Tab öffnen) begonnen oder aus dem Gesamtmodell Bereiche oder einzelne Elemente ausgewählt. Dies kann durch visuelle Auswahl oder die Suchfunktion geschehen. Der Beginn in einem neuen, leeren Tab stellt den Beginn einer explorativen Extraktion dar. Nach der Auswahl eines oder mehrerer Startknoten über die Suchfunktion wird die Ontologie dann durch Stern- und Pfadoperationen erkundet. Dadurch werden nach und nach weitere Knoten und Kanten sichtbar und ein Teilgraph entsteht.

Das Kopieren und Einfügen von Teilmodellen erfolgt nach einem wiederkehrenden Muster. Zunächst wird ein Ausschnitt oder ein einzelner Knoten markiert. Zum Markieren von mehreren Elementen wird bei gedrückter STRG - oder Shift -Taste ein rechteckiger Rahmen gezogen. Die Tasten C oder S kopieren den markierten Bereich. In einem neuen Tab kann dieser dann eingefügt werden. Zum Einfügen sind die Tasten V oder P vorgesehen.

Die einzelnen Teilgraphen können über die Layoutoperationen und manuell (evtl. unter Zuhilfenahme der Grid-Funktion) neu organisiert werden. Die Tabs sind durch Drag&Drop per Maus verschiebbar. Die Session wird mit Speiche-

rung des aktuellen Standes oder einzelner Teile abgeschlossen. Dabei werden Datenobjekte (JSON-Dateien) erzeugt. Die Sitzung gilt als beendet, sobald das Browserfenster aktualisiert wird, zu einer anderen Webadresse navigiert oder den Browser schließt.

Im letzten Teil dieses Kapitels erfolgt die inhaltliche Erklärung der entwickelten Szenarien.

SZENARIO 1 – DATEN-INFORMATION-WISSEN Das erste Szenario (siehe Abb. A.2) stellt den Zusammenhang zwischen *Daten, Information* und *Wissen* dar. Aus dem Teilmodell lässt sich dieser Zusammenhang anhand der Kantenbezeichnung ableiten. Daten sind verbunden mit Informationen und Wissen. Wissen ist eine Subklasse von Information. Semantisch deutet dies auf den Zusammenhang hin, dass Information aus Daten abgeleitet werden kann, während Wissen eine spezielle Art von Information darstellt. Transitiv bedeutet dies, dass auch Wissen aus Daten abgeleitet werden kann. Dies deckt sich mit den in Kapitel 2.1 von Ammenwerth u. a., 2014 beschriebenen Zusammenhängen. Dort werden Daten als „Gebilde aus Zeichen[...], die aufgrund bekannter oder unterstellter Abmachungen Informationen und Wissen darstellen können" beschrieben. Wissen wird als „Information im weiteren Sinne" definiert. Die grafische Darstellung dieser Begriffe ist eingängig und kann gut in Vorlesung und Übung genutzt werden.

SZENARIO 2 – VON SYSTEMEN ZU INFORMATIONSSYSTEMEN In Szenario 2 (siehe Abb. A.5) werden zunächst Systeme allgemein betrachtet. Hierbei zeigen sich verschiedene Subklassen-Beziehungen. Der Begriff *System* ist demnach Superklasse der Begriffe *offenes/geschlossenes System, statisches/dynamisches System, sozio-technisches und Informationssystem* sowie *Teilsystem*. Die Besonderheit der Beziehung von System und Teilsystem ist eine Komponentenbeziehung. Ein Teilsystem ist demnach immer komplett im System enthalten. Auffallend in diesem Beispiel ist, dass auch ein Teilsystem ein System ist, also gilt die Teilmengenbeziehung. Dies kann gut mit den Studierenden im Seminar diskutiert werden. Die Zusammenhänge werden in Abb. A.3 dargestellt.

In Abb. A.4 wird die weitere Entwicklung ausgehend vom Begriff *Informationssystem* gezeigt. Auch hier sind Teilmengenbeziehungen zu finden. Neben der trivialen Beziehung zu *Sub-Informationssystem* ist die Subklassenbeziehung zum *rechnerunterstützten Informationssystem* und zum *Krankenhausinformationssystem* zu erkennen. Weiterhin existieren die Komponenten *Anwendungssystem* und *computerunterstützter Teil eines Informationssystems*. Informationssysteme sind verwandt (*related*) mit dem Begriff *Informations- und Kommunikationssystem*. Aus dem Teilmodell lassen sich auch funktionale Beziehungen ableiten. So benutzt *Informationsmanagement* Informationssysteme, genauso wie *strategisches Informationsmanagement* und die *Rahmenplanung von Informationssystemen*. Auch die *Unternehmensaufgabe*, welche mit dem *Geschäftsprozess (Business Process)* verbunden ist, steht in einer Komponentenbeziehung zum Informationssystem.

Komplettiert wird das Modell durch drei Funktionen des (taktischen) Informationsmanagements: Systemanalyse und -bewertung, Systemspezifikation und Systemevaluation. Dieses Szenario ist wie im letzten Kapitel beschrieben als

Sitzung abgelegt und bietet daher die Möglichkeit, in jedem der Schritte nahezu beliebig weiter zu arbeiten.

SZENARIO 3 – DER PROJEKTPFAD Im letzten Szenario wird der Ablauf eines Projektes dargstellt. Dieser bildet den „roten Faden“ des Lehrbuches Ammenwerth u. a., 2014 und der zugehörigen Lehrveranstaltung. Er findet sich bereits auf dem Buchcover wieder und soll nun näher erläutert werden: Die *Projektarbeit* umfasst alle Phasen von der *Projektinitiierung* bis zum *Projektabschluss*. Diese Phasen sind im Modell extra markiert und als Kette von Funktionen dargestellt. Nach der Initiierung, die unter anderem den *Projektauftrag* aktualisiert, folgt die *Projektplanung*. Sie nutzt den Projektauftrag und führt zur *Projektdurchführung*. Dieser Abschnitt wird in die Phasen *Systemanalyse und -bewertung*, *Systemauswahl*, *Systemeinführung* und *Systemevaluation* unterteilt. Das *Projekt* wird weiterhin für die Funktionen *Projektüberwachung*, *Projektsteuerung* und *Projektmarketing* genutzt. Der *Projektabschluss* bildet das finale Glied des Projektpfades und stellt eine wichtige Phasen des Projektes dar. Schließlich ist ein Projekt unter anderem als ein Vorhaben mit klarer zeitlicher Beschränkung definiert und die Übergabe der Ergebnisse von Projektleiter zu Auftraggeber der finale Meilenstein.

In diesem Szenario fehlen zur besseren Übersicht die im Lehrbuch beschriebenen Rollen. Dabei handelt es sich neben der Projektleitung um den Auftraggeber, die Projektmitarbeiter und den Projektlenkungsausschuss. Die Erarbeitung und Zuordnung ist ein guter Einstiegspunkt für eine gemeinsame Arbeit mit den Studierenden in Vorlesung oder Übung.

7 DISKUSSION UND AUSBLICK

Das abschließende Kapitel beleuchtet und diskutiert kritisch die erzielten Ergebnisse. So werden besondere Herausforderungen und Grenzen bei der Umsetzung erläutert und aufgezeigt. Des Weiteren wird ein Ausblick in die Zukunft gewagt, der sowohl Ideen als auch Vorschläge des Autors enthält, wie das SNIK Projekt und speziell die grafische Repräsentation SNIK Graph künftig weiter voran gebracht werden können. Zur besseren Übersicht ist dieses Kapitel in zwei Teile gegliedert. Zunächst wird die Implementierung und allgemein die Visualisierung der SNIK Ontologie diskutiert. Im Anschluss geht es um die Hilfestellungen, die zur umfangreicheren Nutzung von SNIK Graph beitragen sollen.

7.1 ZUR IMPLEMENTIERUNG VON SNIK GRAPH

Der Umstieg von einer Singleview- hin zu einer Multiview-Anwendung ist mit weitreichenden Konsequenzen verbunden. Besonders sei hier die Umgestaltung des Menüs erwähnt. Die Einträge waren vorher nur auf eine Instanz ausgelegt und mussten nun komplett neu überdacht werden. Die Einführung eines teilmodellspezifischen Menüs konnte hier als guter Ansatz umgesetzt werden, um zumindest einen großen Teil der Herausforderungen zu lösen. Diese Aufgabe ist jedoch bei weitem noch nicht abgeschlossen und muss gerade im Bereich Filter und Optionen noch einmal grundlegend neu bewertet und untersucht werden. Die nun gefundenen Lösungen sind im Rahmen der Gegebenheiten benutzerfreundlich umgesetzt, jedoch gibt es auch in diesem Bereich noch Spielraum für Verbesserungen.

Eine Begrenzung im Vorgehen ist der Anspruch, SNIK Graph ohne Backend oder externe Datenbank (vom SPARQL Endpunkt einmal abgesehen) betreiben zu wollen. Eine History-Funktion ist nicht sinnvoll umsetzbar und damit eine integrierte Nutzerverwaltung mit gespeicherten Sitzungen nicht möglich. Die erweiterten Lösungen zum Speichern und Laden von einzelnen Teilmodellen und ganzen Sitzungen funktionieren gut und zuverlässig, sind allerdings empfindlich gegenüber Veränderungen in den verwendeten Datenstrukturen. So führt eine strukturelle Änderung der JSON-Dateien bzw. der Save/Load-Algorithmen dazu, dass eventuell vorher gespeicherte Modelle nicht mehr ohne weiteres geladen und weiter bearbeitet werden können. Um diesem Problem entgegen zu wirken, ist ein guter Ansatz, in die jeweiligen Dateien eine Art Versionierung zu integrieren. Dazu wurde bereits in die JSON-Dateien ein neues Feld „Version“ eingefügt, welches auf die aktuelle Release-Version gesetzt wird. Beim Laden erfolgt eine Abfrage auf den Wert. Wenn der Vergleich mit der aktuellen Version eine Abweichung ergibt, wird eine Warnung ausgegeben. Dieses Vorgehen

kann verbessert werden, um zum Beispiel nur bei einem Major-Release[1], einem Release, das keine Abwärtskompatibilität garantiert, zu warnen.

Zur Erweiterung der JSON-Dateien zu besseren Speicherobjekten könnte noch die Config-Klasse gespeichert werden, um unter anderem Werte wie gesetzte Filter, Daymode oder die Grid-Funktion zu sichern und im Nachhinein laden zu können. Momentan wird zusätzlich ein State-Objekt (Klasse *state.js*) gespeichert, was neben der Version zum Zeitpunkt des Speicherns die Stati der Punkte Optionen und Filter enthält. Diese werden beim Laden einer neuen Session entsprechend gesetzt und zur Laufzeit angewendet. Das Vorgehen sollte noch einmal genauer untersucht werden. Hier gibt es nach Meinung des Autors drei Betrachtungsebenen für Konfigurationen: die der gesamten Anwendung, die des Status der Anwendung und die der nutzerspezifischen Einstellungen.

Auf der Ebene der gesamten Anwendung finden sich allgemeine Einstellungen wie das Farbschema der Subontologien und der benutzte SPARQL-Endpunkt. Diese werden in der Klasse *config.js* aggregiert. Der Vorteil ist die Möglichkeit von zentralen Änderungen, ohne dass tiefe Implementierungsdetails bekannt sein müssen. Damit lässt sich SNIK Graph auch von Nutzern, die andere Ontologien darstellen möchten, relativ einfach administrieren.

Die zweite Ebene ist der Status der Anwendung selbst. Dieser wird in der Klasse (*state.js*) gespeichert. Dort können neben der Versionsnummer, den gesetzten Optionen und aktiven Filtern beispielsweise das Layout, der verwendete Graph und die aktiven Tabs gespeichert werden.

Schließlich wäre eine Nutzerkonfiguration sinnvoll, die initial nutzerspezifische Präferenzen setzt. Dazu zählen unter anderem die Sprache und die Auswahl des Daymode.

Zu erwähnen ist, dass die Umsetzung der Integration und Konfiguration von GoldenLayout bisher nach eigenem Ermessen betrieben wurde. Hier sollte in einer anschließenden Arbeit mit der Benutzergruppe eine Validierung und Optimierung erfolgen. Überlegungen in diese Richtung betreffen neben dem teilmodellspezifischen Menü auch die Option, Teilmodelle in komplett eigene Browserfenster oder -tabs auslagern zu können. Diese Funktionalität ist technisch zumindest seitens GoldenLayout möglich, jedoch ist nicht sicher wie sich dies mit CytoscapeJS verträgt. Ein weiterer Optimierungsansatz ist das Auslagern sämtlicher tabspezifischer Funktionalitäten in den Header, um auch optisch eine klare Trennung zu den globalen Funktionen zu schaffen. Es ist zu überlegen, die Filter tabspezifisch zu setzen. Ein Vorteil ist, dass auch Modelle unterschiedlicher Quellen in der selben Sitzung bearbeitet werden können. Nachteile ergeben sich aus einem potentiell unübersichtlichen teilmodellspezifischen Menü, da aktuell zahlreiche Filteroptionen angeboten werden. Ein weiterer interessanter Punkt ist die Möglichkeit der Nutzung verschiedener Ontologien mittels SNIK Graph. So kann die Visualisierung einer breiteren Nutzerschaft dienen und besser genutzt und weiterentwickelt werden. Dies ist prinzipiell bereits möglich und wird auch schon umgesetzt[2], jedoch ergeben sich weitere Herausforderungen

1 dieser Begriff stammt aus dem Konzept der semantischen Versionierung, siehe Spezifikation: `https://semver.org/lang/de/`

an die Erweiterung. So müssten auch Informationen zur Ontologie in die Speicherobjekte geschrieben werden, damit die Daten vom richtigen Endpunkt abgerufen werden können. Sonst ist es unmöglich, an vorher gespeicherten Teilgraphen adäquat weiterarbeiten zu können.

Das Thema Performance ist aktuell noch nicht endgültig und sicher bewertbar. Ladezeiten lassen sich gut mithilfe von Cypress messen und vergleichen. Für diese Arbeit wurde das bereits implementierte Benchmark-Tool zum Vergleich zwischen der Singleview- und der Multiview-Anwendung herangezogen. Die Auswertung ist hierbei als händisch und oberflächlich zu bezeichnen und sollte bei Bedarf tiefer erforscht werden, um frühzeitig Memory Leaks und andere Performance-Bottlenecks erkennen zu können. Die vorhandenen Probleme, an denen fortlaufend gearbeitet wird, bleiben jedoch zunächst bestehen. Dazu zählen die teils sehr komplexen und damit teuren SPARQL Abfragen und das aufwändige Layouting. Die Abfragen treten jedoch nur beim initialen Laden und beim Erstellen des Suchindex auf. Das Layouting leidet an der Canvas-basierten Darstellung von CytoscapeJS. Eine parallelisierte und optimierte Desktop-Anwendung ohne JavaScript würde hier Abhilfe schaffen. Bei der eigenen Nutzung wurde festgestellt, dass der Arbeitsspeicher des Systems bei längeren Sitzungen voll lief und damit das System zum „Einfrieren“ kam. Jedoch konnte hier nicht endgültig geklärt werden, ob die Ursache an einem potentiellen Memory Leak von SNIK Graph oder an dem für die Programmierung verwendeten Sourcecode-Editor lag. Hier sollte noch eine Klärung erfolgen.

Die Arbeit mit SNIK Graph und speziell mit der Erweiterung ist attraktiver geworden und wird zukünftig einen größeren Nutzerkreis erfreuen. Eine nachhaltigere Entwicklung wäre hier wünschenswert, um zukunftsfähig zu werden und zu bleiben. Ein Entwicklerteam, welches größer als 1,5 Personen ist, könnte eine architektonische Neuausrichtung angehen.

Hierfür würde sich das Framework *React*[3] anbieten, was sehr gut mit GoldenLayout harmoniert und auch für die Performance einige Vorteile bringen könnte. So wäre eine deutlich bessere User Experience zu erzielen, da React unter anderem durch dynamisches Laden von Inhalten das Gefühl einer nativen Web-App ermöglicht. Es existiert ein großes Ökosystem inklusive vieler Tutorials und Dokumentationen. Es gibt bereits ein Projekt, welches CytoscapeJS Graphen mittels React darstellt[4]. Die Architektur einer Singlepage-Anwendung ist bereits implizit vorhanden (kleinere Seiten wie das Handbuch können über Routing angesprochen werden). Mithilfe von React lässt sich eine bessere Trennung von Belangen in Bezug auf die Client-Server-Lastverteilung erzielen. So kann clientseitig nur noch die Anzeige und serverseitig die restliche Businesslogik realisiert werden.

Um sich dem Thema Performance zu nähern wurde auch zum Thema Parallelisierung von Prozessen recherchiert. Vor der Erweiterung war dies kein Thema, jedoch wird nun in verschiedenen Modellen gearbeitet und entsprechend „gerechnet“. Obwohl JavaScript an sich keine parallelen Prozesse im Sinne von

2 beispielsweise kann die Health IT Ontology (HITO) (Dornauer u. a., 2020) dargestellt werden: `https://hitontology.eu/ontology/`

3 siehe `https://reactjs.org/`

4 siehe `https://github.com/plotly/react-cytoscapejs`

Multithreading unterstützt, könnte über das Konzept von Webworkers[5] eine Parallelisierung der unterschiedlichen Teilmodelle erreicht werden. Dies ist in einer Issue aus dem optionalen Meilenstein[6] festgehalten. Des Weiteren könnte über eine passende Backend- und Datenbanklösung nachgedacht und entschieden werden.

Die neue Grid-Funktion bietet noch Möglichkeiten zur Verbesserung. So wäre eine automatische Ausrichtung am Raster eine zusätzliche Hilfe zur Organisation von Teilgraphen. Auch ein „Einrast-Mechanismus" ist denkbar. Beim Verschieben ist weiterhin ein Raster sichtbar. Bei der Annährung an die Grenzen eines Rasterquadrates schnappt das bewegte Element in dessen Mitte ein. Dieser Mechanismus ist erweiterbar auf das gleichzeitige Verschieben und Einrasten mehrerer Elemente.

Bei der praktischen Arbeit mit SNIK Graph gibt es nach Meinung des Autors auch einige Optimierungsmöglichkeiten. Zunächst ist das Konzept der Kapitelsuche, die momentan nur für Winter u. a., 2011, das sogennante Blaue Buch, implementiert wurde, auch für andere Quellen sinnvoll. Dies bedingt allerdings noch einiges an ontologischer Extraktionsarbeit, die sicherlich auch die Datenqualität weiter voranbringen würde. Der Wunsch, die grafische Darstellung mit zusätzlichen Informationen, wie z.B. Definitionen über LodView, anzureichern, bleibt gerade aus Sicht eines Studierenden bestehen und ließe sich mithilfe der neuen Erweiterung gut umsetzen.

7.2 ZUR NUTZUNG DER VISUALISIERUNG

Um die neue Erweiterung gut nutzbar zu machen und allgemein die Nutzerzahlen zu erhöhen, werden verschiedene Hilfestellungen gegeben. Neben der Aktualisierung des Handbuches[7] wurde ein einfach gehaltenes Prozessmodell (siehe Abb. A.1) entwickelt, dass nur den Basisprozess abbilden soll. Es wartet daher nicht mit allen funktionalen Optionen auf und vermeidet damit unnötige Verwirrung. Als Modellierungssprache bietet sich BPMN 2.0 an, da sie bereits im Bachelorstudium behandelt wird und gut verständlich ist. Die Arbeit mit SNIK Graph ist anhand dieses Basisprozesses bereits gut möglich und regt den Nutzer zum Probieren und Finden von eigenen Wegen und Abläufen an.

Ein ähnliches Ziel wird mit den entwickelten Sitzungen und Teilmodellen angestrebt. Sie stellen nur Teile des vermittelten Wissens dar und können unterstützend oder zur Anregung von Diskussionen in Vorlesungen und Übungen eingesetzt werden. Die zur Verfügung gestellten JSON-Dateien sollen den Studierenden und Lehrenden als Anregung dienen, damit zu arbeiten oder selbst weitere Teilmodelle zu entwickeln. Im Hinblick auf die Prüfungsvorbereitung befördert die Arbeit mit SNIK Graph das Erkennen von Zusammenhängen und hilt beim Lernen der Definitionen. Der Autor dieser Arbeit hat damit äußerst gute Erfahrungen gemacht und auch im Kommilitonenkreis stieß dieses System auf Interesse und Begeisterung. Eine gezielte Einführung bereits in der Bachelorveranstaltung „Einführung in die medizinische Informatik und das taktische

5 siehe `https://developer.mozilla.org/de/docs/Web/API/Web_Workers_API`
6 siehe `https://github.com/IMISE/snik-cytoscape.js/issues/279`
7 zu finden unter `https://www.snik.eu/graph/manual.html`

Informationsmanagement im Krankenhaus“ würde den Einfluss von SNIK Graph als E-Learning-System erhöhen und eventuell sogar die Rekrutierung fähiger Studenten zur Weiterentwicklung des Systems befördern.

Abschließend wird ein Folgeprojekt zum weiteren Ausbau empfohlen. Mit einer besseren Anbindung an die Semantic Web Community und der noch ausgereifteren Möglichkeit, auch fremde Ontologien darstellen zu können, ließe sich die Nutzerzahl deutlich erhöhen. Mit dem aktuellen Stand bietet SNIK Graph eine gute Basis für ein E-Learning-System. Die eigene Erfahrung gerade in Zeiten der Corona-Pandemie hat gezeigt, wie wichtig gute Online- und E-Learning Systeme sind. Diese Erfahrung und die gesellschaftlichen Lehren aus dieser Zeit werden die Entwicklung solcher Systeme hoffentlich zusätzlich befördern und unterstützen.

7.3 EIN KURZES SCHLUSSWORT

Der neue Stand der Visualisierung macht den Autor sehr stolz und glücklich. Die Arbeit bereitete stets große Freude und war interessant. In Zusammenarbeit mit dem Betreuer dieser Arbeit, Konrad Höffner, wurde stets nach optimalen Wegen auch außerhalb der eigenen Komfortzone gesucht. Dieses Vorgehen hatte einen großen Einfluss auf diese Arbeit und auch auf das Studium allgemein. Der Autor freut sich sehr auf die weiteren Herausforderungen im Master-Studium mit dem Schwerpunkt medizinische Informatik und hofft, dass die neue Erweiterung optimal in den Vorlesungsbetrieb integriert werden kann.

8

ZUSAMMENFASSUNG

Diese Arbeit aus dem Bereich der medizinischen Informatik befasst sich mit der Organisation von Wissen auf dem Teilgebiet des Managements von Krankenhausinformationssystemen. Dabei wird eine graphische Visualisierung von semantischen Daten, die aus Lehrbüchern und anderen Quellen extrahiert wurden, analysiert und erweitert.

Das Ziel ist, das Vorgehen zur Erstellung und Organisation von für die Lehre sinnvollen Teilgraphen zu optimieren. Dafür wird die Webanwendung um die Funktion erweitert, an mehreren Teilmodellen gleichzeitig arbeiten zu können. So soll dieses Verfahren beschleunigt und für den Einsatz in der Lehre attraktiver gestaltet werden. Methodisch orientiert sich die Arbeit an dem Ablauf eines Projektes, wie er in Ammenwerth u. a., 2014 beschrieben wird. Nach der Analyse und Bewertung der bestehenden Anwendung erfolgt die Anforderungsanalyse und die Umsetzung im Sinne der Einführung des erweiterten Systems. Die Evaluation ist nicht Bestandteil dieser Arbeit.

Die erhobenen Anforderungen und Aufgabenstellungen wurden erfolgreich umgesetzt. Das Arbeiten in mehreren Teilmodellen unterstützt die Nutzer des Systems dabei, schneller bessere Ergebnisse bei der Erstellung und Organisation von Teilgraphen zu erzielen. Erstellte Szenarien können als Sitzungen gespeichert und weiter bearbeitet werden. Zusätzlich wird der Basisprozess beschrieben und modelliert. Um die anforderungsgemäße Funktionalität der Erweiterung zu belegen und Lehrende sowie Studierende zu unterstützen werden mehrere Teilgraphen fachlich beschrieben und extrahiert. Dies wird mittels selbst gewählter Szenarien erreicht, die in Vorlesungen und Übungen eingesetzt werden können.

Die im Rahmen dieser Arbeit erzielten Ergebnisse bereichern das Arbeiten mit der graphischen Visualisierung und können hoffentlich eine breitere Nutzerschaft ansprechen. Dennoch werden zahlreiche Anregungen und Vorschläge zur weiteren Entwicklung der Visualisierung als E-Learning System gegeben.

LITERATUR

Allemang, Dean und James Hendler (2008). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann Publishers Inc.

Ammenwerth, Elske, Reinhold Haux, Petra Knaup-Gregori und Alfred Winter (2014). *IT-Projektmanagement im Gesundheitswesen: Lehrbuch und Projektleitfaden Taktisches Management von Informationssystemen*. Schattauer Verlag.

Anonym (2019). *Extraktion von Teilgraphen aus der SNIK-Ontologie zur Unterstützung der Lehre (Bachelorarbeit)*.

Anutariya, C. und R. Dangol (2018). „VizLOD: Schema Extraction And Visualization Of Linked Open Data". In: *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, S. 1–6.

Aranda, Carlos Buil u. a. (2013). *SPARQL 1.1 Overview*. W3C Recommendation. W3C. URL: http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ (besucht am 25. 03. 2020).

Baron Neto, Ciro u. a. (2016). „LODVader: An Interface to LOD Visualization, Analyticsand DiscovERy in Real-Time". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, S. 163–166. ISBN: 9781450341448.

Bellini, Pierfrancesco, Paolo Nesi und Alessandro Venturi (2014). „Linked Open Graph: browsing multiple SPARQL entry points to build your own LOD views". In: *Journal of Visual Languages & Computing* 25.6, S. 703–716.

Benedetti, Fabio, Sonia Bergamaschi und Laura Po (2014). „A visual summary for linked open data sources". In: *ISWC 2014 Posters & Demonstrations Track*. Bd. 1272, S. 173–176.

Berners-Lee, Tim (2006). *Geschichte des W3C*. URL: https://www.w3c.de/about/geschichte/ (besucht am 25. 03. 2020).

Berners-Lee, Tim (2010). *5 Sterne Datenmodell nach Tim Berners-Lee*. URL: https://www.w3.org/DesignIssues/LinkedData.html (besucht am 25. 03. 2020).

Berners-Lee, Tim, Roy T. Fielding und Larry Masinter (Jan. 2005). *Uniform Resource Identifier (URI): Generic Syntax*. Techn. Ber. 3986. Internet Engineering Task Force (IETF), Network Working Group. URL: https://tools.ietf.org/html/rfc3986 (besucht am 25. 03. 2020).

Berners-Lee, Tim, James Hendler und Ora Lassila (2001). „The semantic web". In: *Scientific american* 284.5, S. 34–43.

Berners-Lee, Tim, Larry Masinter und Mark McCahill (Dez. 1994). *Uniform Resource Locators (URL)*. Techn. Ber. 1738. Internet Engineering Task Force (IETF), Network Working Group. URL: https://tools.ietf.org/html/rfc1738 (besucht am 25. 03. 2020).

Berners-Lee, Tim u. a. (1994). „The world-wide web". In: *Communications of the ACM* 37.8, S. 76–82.

Bewersdorff, Jörg (2018). *JavaScript: Der Start*. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-21077-9.

Bikakis, N. u. a. (2016). „graphVizdb: A scalable platform for interactive large graph visualization“. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, S. 1342–1345.

Bikakis, Nikos u. a. (2015). „Towards Scalable Visual Exploration of Very Large RDF Graphs“. In: *The Semantic Web: ESWC 2015 Satellite Events*. Springer International Publishing, S. 9–13. ISBN: 978-3-319-25639-9.

Brigl, Birgit, Anke Häber, Thomas Wendt und Alfred Winter (2004). „Ein 3LGM2 Modell des Krankenhausinformationssystems des Universitätsklinikums Leipzig und seine Verwertbarkeit für das Informationsmanagement“. In: *Modellierung betrieblicher Informationssyteme–MobIS 2004*.

Camarda, Diego Valerio, Silvia Mazzini und Alessandro Antonuccio (2012). „LodLive, Exploring the Web of Data“. In: *Proceedings of the 8th International Conference on Semantic Systems*. Association for Computing Machinery, S. 197–200. ISBN: 97814503111120.

Chawuthai, Rathachai und Hideaki Takeda (2016). „RDF Graph Visualization by Interpreting Linked Data as Knowledge“. In: *Semantic Technology*. Springer International Publishing, S. 23–39. ISBN: 978-3-319-31676-5.

Cheng, Gong, Yanan Zhang und Yuzhong Qu (2014). „Explass: Exploring Associations between Entities via Top-K Ontological Patterns and Facets“. In: *The Semantic Web – ISWC 2014*. Springer International Publishing, S. 422–437. ISBN: 978-3-319-11915-1.

Dam, Jesse CJ van u. a. (2015). „RDF2Graph a tool to recover, understand and validate the ontology of an RDF resource“. In: *Journal of biomedical semantics* 6.1, S. 39.

De Vocht, Laurens u. a. (2013). „Resxplorer: interactive search for relationships in research repositories“. In: *International Semantic Web Conference: Semantic Web Challenge, Abstracts*. Citeseer, S. 8.

De Vocht, Laurens u. a. (2015). „A visual exploration workflow as enabler for the exploitation of linked open data“. In: *IESD'14 Proceedings of the 3rd International Conference on Intelligent Exploration of Semantic Data*. Bd. 1279, S. 30–41.

Deligiannidis, Leonidas, Krys J. Kochut und Amit P. Sheth (2007). „RDF Data Exploration and Visualization“. In: *Proceedings of the ACM First Workshop on CyberInfrastructure: Information Management in EScience*. Association for Computing Machinery, S. 39–46. ISBN: 9781595938312.

Diestel, Reinhard (2012). *Graphentheorie*. 4.Auflage, 1.korr. Nachdr. Heidelberg, Germany: Springer. ISBN: 9783642149122.

Dokulil, J. und J. Katreniakova (2009). „Using Clusters in RDF Visualization“. In: *2009 Third International Conference on Advances in Semantic Processing*, S. 62–66.

Dornauer, Verena u. a. (Juni 2020). „Use of Natural Language Processing for Precise Retrieval of Key Elements of Health IT Evaluation Studies“. In: *Studies in health technology and informatics* 272, S. 95–98.

Dudáš, Marek, Vojtěch Svátek und Jindřich Mynarz (2015). „Dataset Summary Visualization with LODSight“. In: *The Semantic Web: ESWC 2015 Satellite Events*. Springer International Publishing, S. 36–40. ISBN: 978-3-319-25639-9.

Graziosi, Alice u. a. (2018). „Customising LOD Views: A Declarative Approach“. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, S. 21852192. ISBN: 9781450351911.

Gruber, Thomas R. u. a. (1993). „A translation approach to portable ontology specifications“. In: *Knowledge acquisition* 5.2, S. 199–221.

Haag, Florian, Steffen Lohmann, Steffen Bold und Thomas Ertl (2014). „Visual SPARQL querying based on extended filter/flow graphs“. In: *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, S. 305–312.

Haag, Florian, Steffen Lohmann und Thomas Ertl (2014). „SparqlFilterFlow: SPARQL Query Composition for Everyone“. In: *The Semantic Web: ESWC 2014 Satellite Events*. Springer International Publishing, S. 362–367. ISBN: 978-3-319-11955-7.

Haag, Florian, Steffen Lohmann, Stephan Siek und Thomas Ertl (2015a). „QueryVOWL: A Visual Query Notation for Linked Data“. In: *The Semantic Web: ESWC 2015 Satellite Events*. Springer International Publishing, S. 387–402. ISBN: 978-3-319-25639-9.

Haag, Florian, Steffen Lohmann, Stephan Siek und Thomas Ertl (2015b). „QueryVOWL: Visual Composition of SPARQL Queries“. In: *The Semantic Web: ESWC 2015 Satellite Events*. Springer International Publishing, S. 62–66. ISBN: 978-3-319-25639-9.

Hastrup, Tuukka, Richard Cyganiak und Uldis Bojars (2008). „Browsing linked data with fenfire“. In:

Heim, Philipp, Steffen Lohmann und Timo Stegemann (2010). „Interactive Relationship Discovery via the Semantic Web“. In: *The Semantic Web: Research and Applications*. Springer Berlin Heidelberg, S. 303–317. ISBN: 978-3-642-13486-9.

Heinrich, Lutz J, René Riedl und Dirk Stelzer (2014). *Informationsmanagement: Grundlagen, Aufgaben, Methoden*. De Gruyter.

Höffner, Konrad u. a. (2019). „Open and Linkable Knowledge About Management of Health Information Systems.“ In: *Studies in Health Technology and Informatics* 264, S. 1678–1679.

Köhler, Claus O (1973). *Historie der Medizinischen Informatik in Deutschland von den Anfängen bis 1980*.

Lohmann, Steffen, Stefan Negru, Florian Haag und Thomas Ertl (2016). „Visualizing ontologies with VOWL“. In: *Semantic Web* 7.4, S. 399–419.

Micsik, András, Zoltán Tóth und Sándor Turbucz (2014). „LODmilla: Shared Visualization of Linked Open Data“. In: *Theory and Practice of Digital Libraries – TPDL 2013 Selected Workshops*. Springer International Publishing, S. 89–100. ISBN: 978-3-319-08425-1.

Micsik, András, Sándor Turbucz und Attila Györök (2014). „LODmilla: A linked data browser for all“. In: *SEMANTICS*, S. 31–34.

Nuzzolese, Andrea Giovanni u. a. (2013). „Aemoo: Exploring Knowledge on the Web“. In: *Proceedings of the 5th Annual ACM Web Science Conference*. Association for Computing Machinery, S. 272–275. ISBN: 9781450318891.

Pietriga, Emmanuel (2002). „IsaViz: a Visual Environment for Browsing and Authoring RDF“. In:

Po, Laura, Nikos Bikakis, Federico Desimoni und George Papastefanatos (2020). *Linked Data Visualization: Techniques, Tools and Big Data*. Synthesis Lectures on Data, Semantics and Knowledge. Morgan & Claypool Publishers, S. 1–157. ISBN: 9781681737256.

Po, Laura und Davide Malvezzi (2018a). „Community Detection Applied on Big Linked Data." In: *J. UCS* 24.11, S. 1627–1650.

Po, Laura und Davide Malvezzi (2018b). „High-level Visualization Over Big Linked Data." In: *International Semantic Web Conference (P&D/Industry/BlueSky)*.

Psyllidis, Achilleas (2015). „OSMoSys: A Web Interface for Graph-Based RDF Data Visualization and Ontology Browsing". In: *Engineering the Web in the Big Data Era*. Springer International Publishing, S. 679–682. ISBN: 978-3-319-19890-3.

Richard Cyganiak David Wood, Markus Lanthaler (2014). *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C. URL: https://www.w3.org/TR/rdf11-concepts/ (besucht am 25. 03. 2020).

Santana-Pérez, Idafen (2018). „Graphless: Using Statistical Analysis and Heuristics for Visualizing Large Datasets." In: *VOILA ISWC*, S. 1–12.

Sayers, Craig (2004). „Node-centric rdf graph visualization". In: *Mobile and Media Systems Laboratory, HP Labs* 71.

Shannon, Paul u. a. (2003). „Cytoscape: a software environment for integrated models of biomolecular interaction networks". In: *Genome research* 13.11, S. 2498–2504.

Spillner, Andreas, Thomas Roßner, Mario Winter und Tilo Linz (2014). *Praxiswissen Softwaretest-Testmanagement: Aus-und Weiterbildung zum Certified Tester-Advanced Level nach ISTQB-Standard*. dpunkt. verlag.

Stäubert, S u. a. (2015). „Modeling Interoperable Information Systems with 3LGM^2 and IHE". In: *Methods of information in medicine* 54.05, S. 398–405.

Sundara, S. u. a. (2010). „Visualizing large-scale RDF data using Subsets, Summaries, and Sampling in Oracle". In: *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, S. 1048–1059.

Tartari, Gonzalo und Aidan Hogan (2018). „WiSP: Weighted shortest paths for RDF graphs". In: 2187:37–52.

The Open Group (2017). „The Open Group IT4IT Reference Architecture, Version 2.1". In: *The Open Group*.

Viola, Fabio u. a. (2018). „Interactive 3d exploration of rdf graphs through semantic planes". In: *Future Internet* 10.8, S. 81.

Weise, Marc, Steffen Lohmann und Florian Haag (2016a). „Extraction and visualization of tbox information from sparql endpoints". In: *European Knowledge Acquisition Workshop*. Springer, S. 713–728.

Weise, Marc, Steffen Lohmann und Florian Haag (2016b). „Ld-vowl: Extracting and visualizing schema information for linked data". In: *2nd International Workshop on Visualization and Interaction for Ontologies and Linked Data*, S. 120–127.

Wendt, Thomas, A Häber, B Brigl und Alfred Winter (2004). „Modeling Hospital Information Systems (Part 2): using the 3LGM2 tool for modeling patient record management". In: *Methods of Information in Medicine* 43.03, S. 256–267.

Winter, A. u. a. (2011). *Health Information Systems: Architectures and Strategies*. Health Informatics. Springer London. ISBN: 9781849964418. URL: https://books.google.de/books?id=RzvmrgwCWncC.

Winter, Alfred und Barbara Paech (2020). *Abschlussbericht des SNIK-Projektes*.

Winter, Alfred u. a. (2007). „3LGM2-Modeling to support management of health information systems“. In: *international journal of medical informatics* 76.2-3, S. 145–150.

Zhang, Kang, Haofen Wang, Duc Thanh Tran und Yong Yu (2010). „ZoomRDF: Semantic Fisheye Zooming on RDF Data“. In: *Proceedings of the 19th International Conference on World Wide Web*. Association for Computing Machinery, S. 1329–1332. ISBN: 9781605587998.

Zhang, Yanan, Gong Cheng und Yuzhong Qu (2014). „Towards Exploratory Relationship Search: A Clustering-Based Approach“. In: *Semantic Technology*. Springer International Publishing, S. 277–293. ISBN: 978-3-319-06826-8.

ANHANG

A

PROZESSMODELL UND SZENARIEN

Der Anhang beinhaltet das BPMN Modell des Basisprozesses zur Bedienung der Erweiterung von SNIK Graph.

Weiterhin finden sich die hier die Bilder der Teilmodelle, die im Rahmen der Szenarien erstellt wurden.

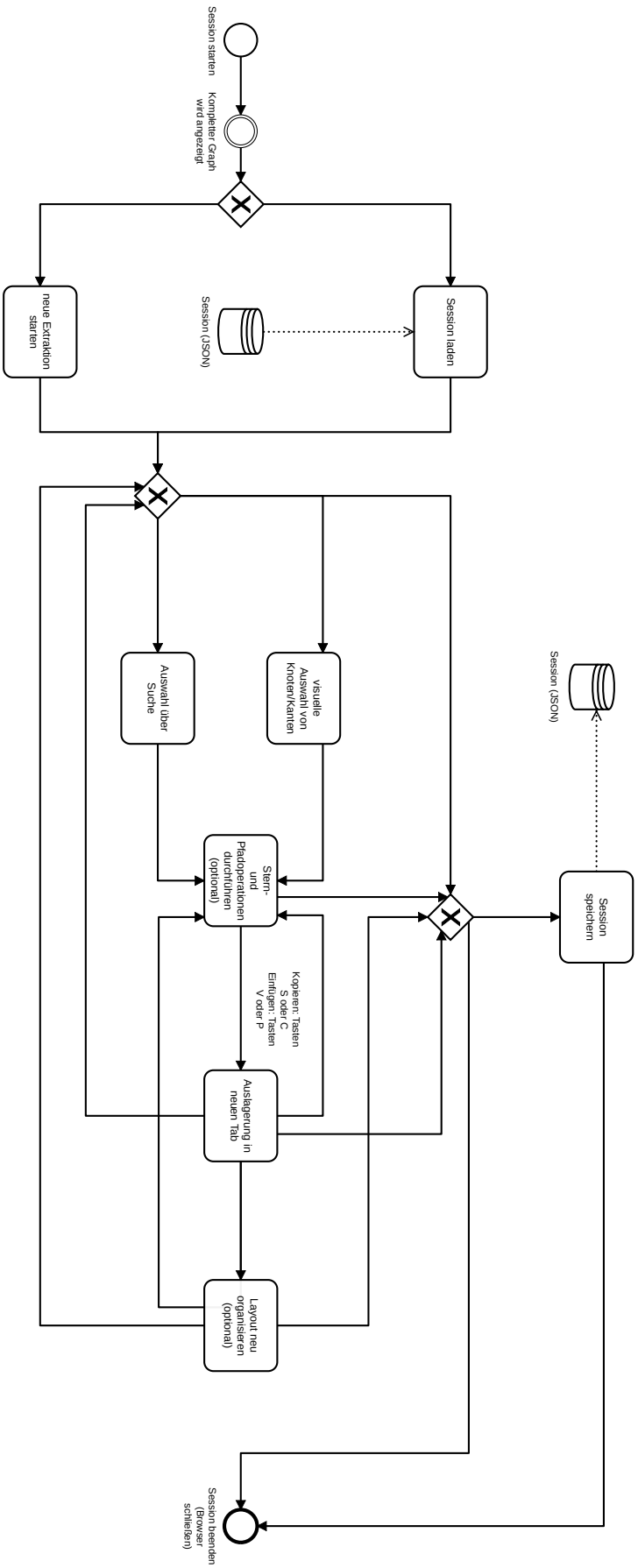

Abbildung A.1: Der Arbeitsprozess mit der Erweiterung von SNIK Graph

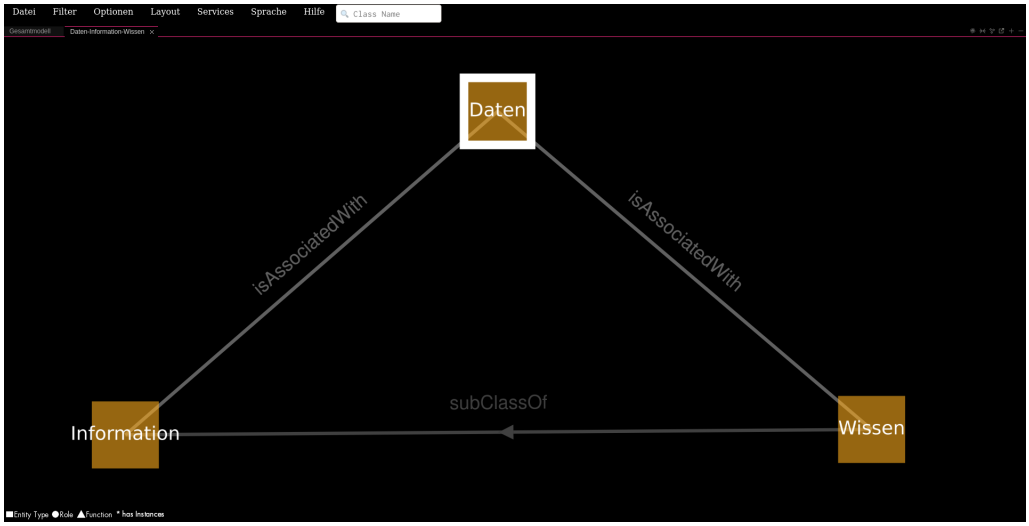

Abbildung A.2: Szenario 1: Daten-Information-Wissen

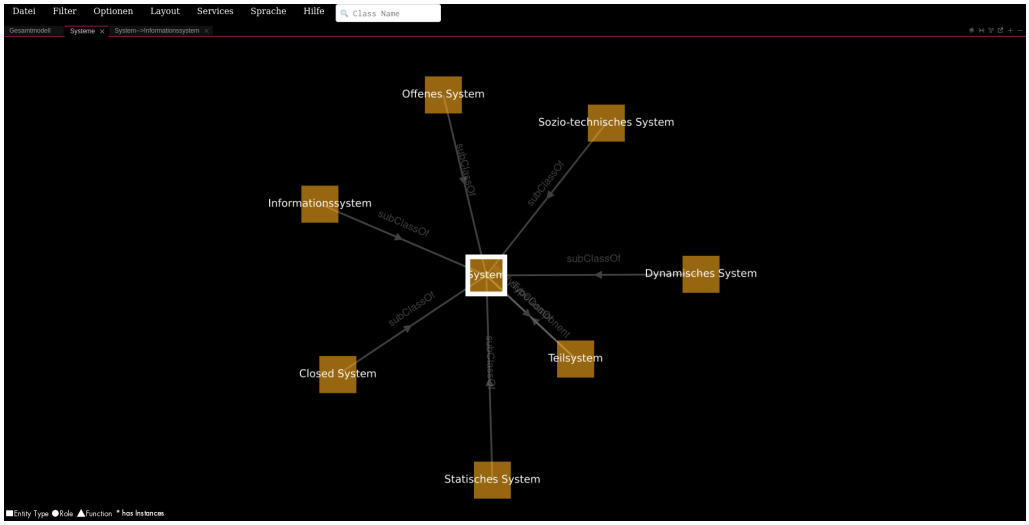

Abbildung A.3: Szenario 2.1: Ausprägungen von „Systemen“

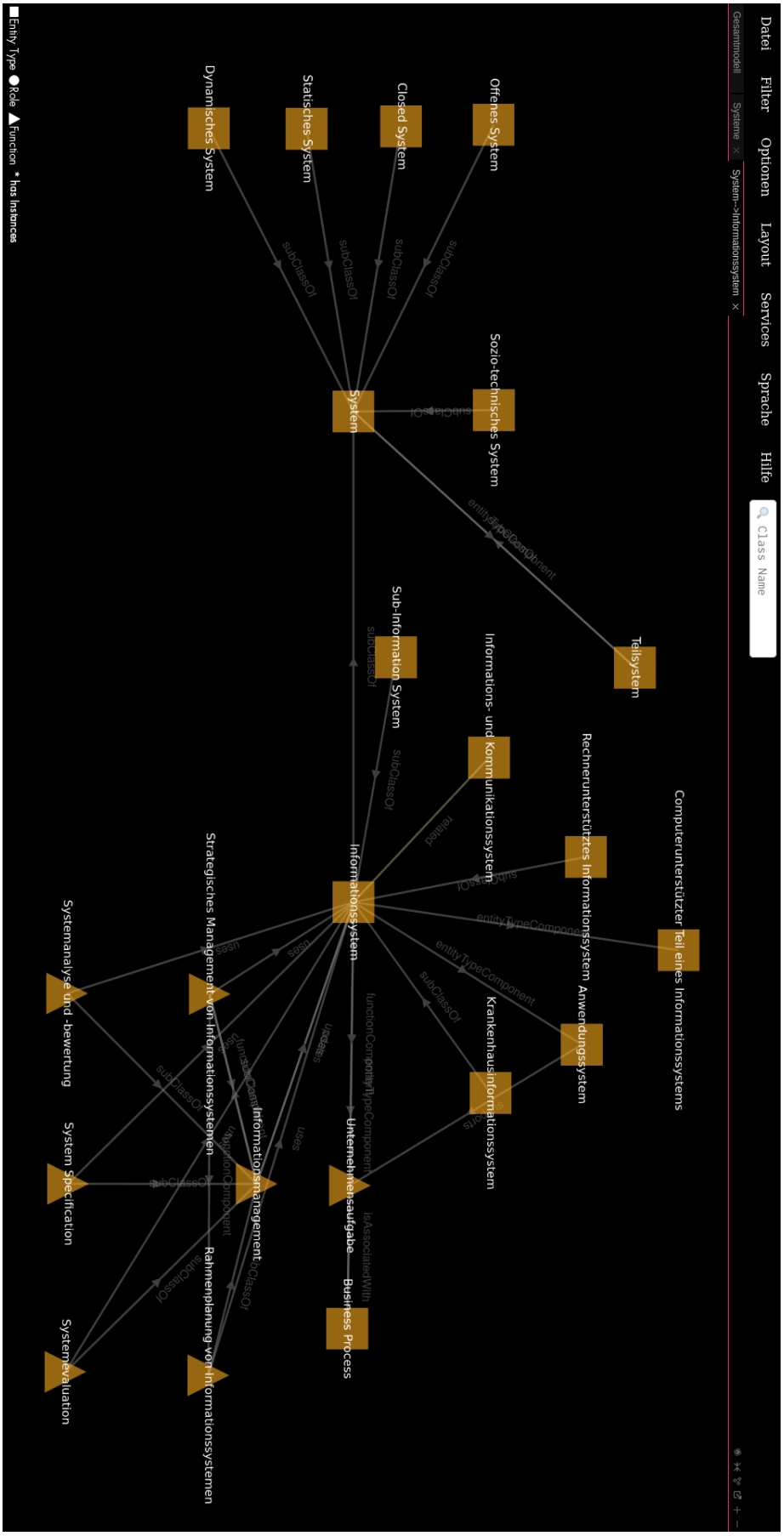

Abbildung A.4: Szenario 2.2: Von Systemen zu Informationssystemen (Einzelansicht)

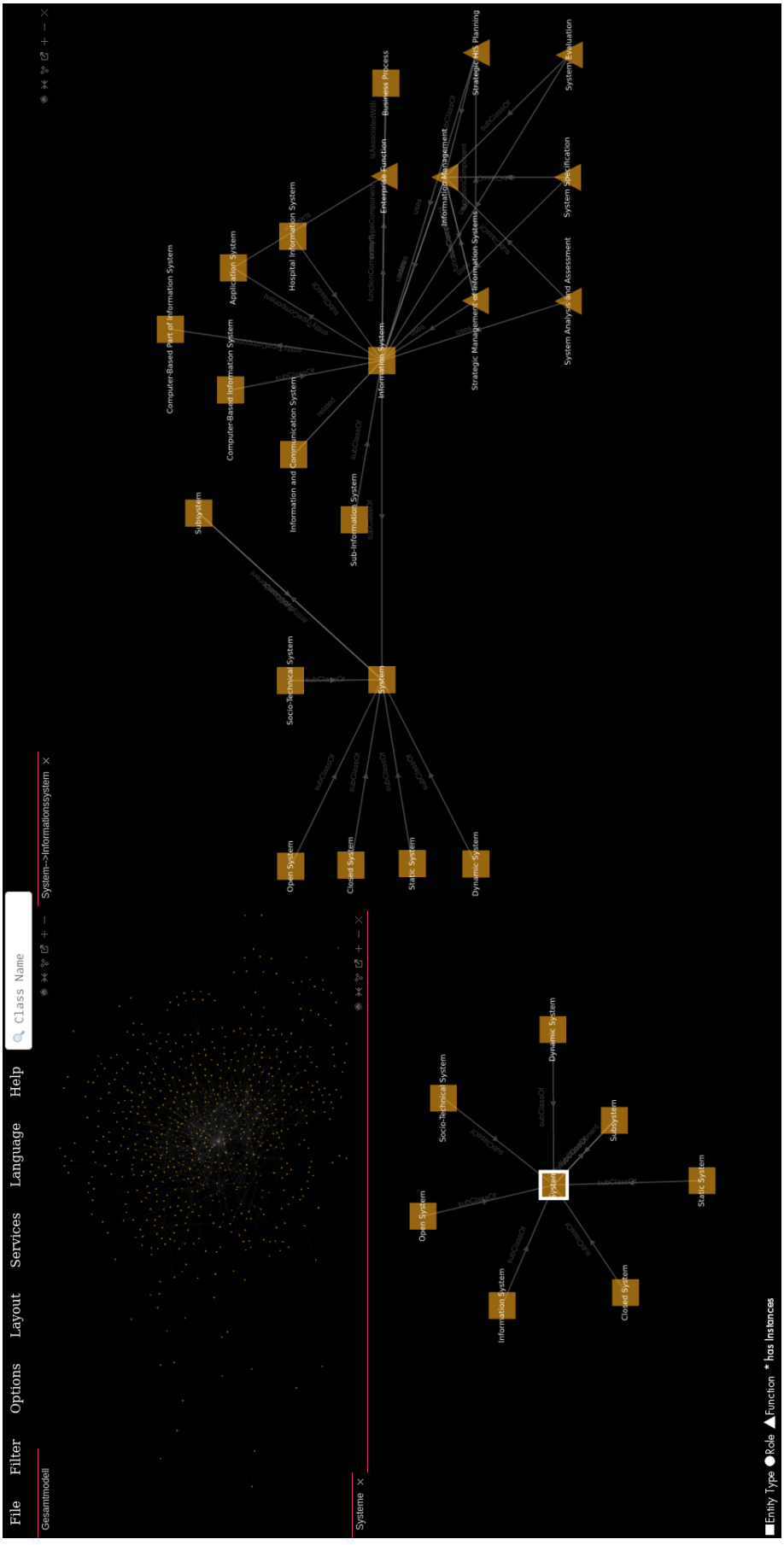

Abbildung A.5: Szenario 2: Von Systemen zu Informationssystemen (Gesamtansicht)

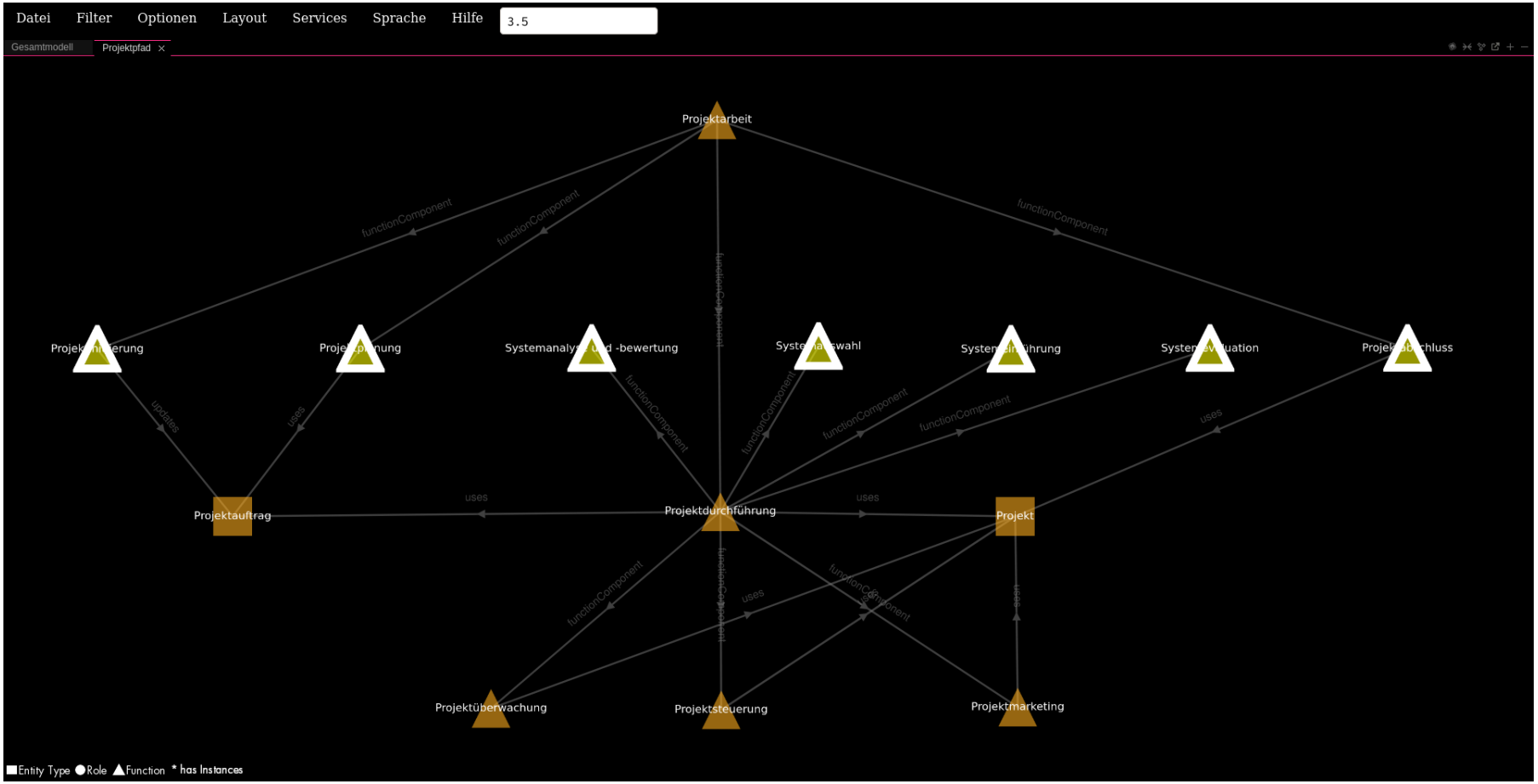

Abbildung A.6: Szenario 3: Der Projektpfad

ERKLÄRUNG

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet.
Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.
Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt.

Leipzig, 19. Oktober 2020

Thomas Pause