Kernkonzepte der Taxonomiesprache GenDifS

Johannes Busse[1]

Abstract: GenDifS ist eine neue Sprache zur Notation von Terminologien und Taxonomien gemäß der Formel *Genus Proximum, Differentia Specifica*. GenDifS nutzt das Opensource Mindmap Tool Freemind/Freeplane zur Notation und ist in frühen und in späten Phasen des Ontology Engineerings ohne Medienbruch nutzbar. Auf der Sachebene werden das Sprachkonzept, die Syntax und die Semantik der Kernkonzepte der Sprache erstmals vorgestellt. Auf der Metaebene trägt der Aufsatz zur Methodologie der Design Science Research bei: Wie lassen sich Artefakte von Typ Terminologie und Sprache durch Rigor entwickeln und evaluieren?

Keywords: Semantic Web, Ontologie, Taxonomie, SKOS, OWL2, Design Science Research.

1 Problemstellung

Seit Aristoteles unterscheidet man zwei Begriffe, indem man zunächst einen „besten" gemeinsamen Oberbegriff sucht (das sogenannte Genus Proximum, die nächsthöhere Gattung) und dann in Bezug auf diesen Oberbegriff das treffendste spezifische Merkmal (die sogenannte Differentia Specifica, den Artunterschied) bestimmt.

Beispiel: In einer Lehrveranstaltung sei gewünscht, eine Taxonomie zu entwickeln, z.B. zur Domäne „Schnitzel". In der Prüfung werde das Begriffsverständnis überprüft, indem der spezifische Unterschied zwischen verschiedenen Schnitzeln erfragt wird, Beispiele: Q: Unterschied Schweineschnitzel – Kalbschnitzel? A: Das Fleisch kommt vom Schwein / vom Kalb, leere Schnittmenge. Q: Schweineschnitzel – Kinderschnitzel? A: Herkunft des Fleisches, Zielgruppe; große Schnittmenge. Q: Kalbschnitzel – Wiener Schnitzel? A: Das ist das Gleiche.

1 HAW Landshut, Fakultät für Informatik, Am Lurzenhof 1, 84036 Landshut,busse@haw-landshut.de

https://doi.org/10.30844/AKWI_2022_14

Das Beispiel zeigt, dass der charakteristische Unterschied manchmal im Wert eines Attributs, manchmal im Attribut selbst liegt, das wir zur Unterscheidung heranziehen. Offensichtlich gibt es unterschiedliche Unterschiede. Einen noch nicht begrifflich erfassten Gegenstandsbereich begrifflich zu durchdringen bedeutet, über eine logisch konsistente Systematik von Unterschieden zu verfügen.[2]

Terminologien entfalten in den begrifflichen Grundlagen eines Faches eine tiefgreifende Ordnungsmacht. Dies gilt erst recht für die Methoden und Sprachen, mit denen Taxonomien entwickelt und notiert werden. Entsprechend scheint es uns geboten, eine Sprache zur Taxonomie-Entwicklung im Sinne der Design Science Research (DSR) auch mit Rigor zu konstruieren und zu evaluieren. Wir stellen unsere neue Sprache GenDifS also nicht nur vor, sondern reflektieren und begründen unser Vorgehen ausführlich insbesondere in Bezug auf Hevners Darstellung der DSR ([HMPR04], im Folgenden pars pro toto mit *Hevner 2004* zitiert). Wir betreiben also auch Wissenschaftstheorie, zu der teilweise auch „philosophierende" Argumentationsmuster hinzugehören.

1.1 Abstraktes und konkretes Business-Problem

Ein erster Schritt zur konkreten Evaluierbarkeit besteht darin, nicht lediglich abstrakte eine Sprache zu evaluieren, sondern auch im Detail anzugeben, welches konkrete Business-Problem mit der neuen Sprache angegangen werden soll.

Stark abstrahiert bearbeiten wir die Herausforderung, Terminologien und Taxonomien visuell als Mindmap formaler zu fassen. Genauer besteht unser *abstraktes Business Problem* darin, eine Sprache zu entwickeln, mit der die Begriffe eines Gegenstandbereichs als eine Taxonomie (a) visuell in einer Mindmap leicht editiert und in einer Form dargestellt werden können, die (b) formal wohldefiniert ist und die (c) so in die Semantic-Web-Sprache OWL2 übersetzbar ist, dass (d) ein Standard OWL2-Reasoner einzelne Objekte anhand ihrer Merkmale automatisch klassifizieren kann. Zusätzlich soll (e) die Mindmap in einen SKOS-Thesaurus übersetzbar sein.

Das bekannteste Modellierungspattern für die Strukturierung von Taxonomien ist die seit mehr als 2000 Jahren bekannte Formel *Genus Proximum, Differentia Specifica*, die wir im folgenden *GPDS-Modellierungspattern* nennen. *GenDifS* wollen wir die Sprache nennen, die (a) im Kern das GPDS-Modellierungspattern realisiert sowie (b) dieses Modellierungspattern um ausgewählte komplementäre Modellierungspatterns erweitert. GenDifS

2 Unsere Auswahl der Beispiele mag irritieren: Wäre es in einem Publikationsorgan des AKWI nicht angemessener, Begriffe aus der Wirtschaftsinformatik als Beispiele heranzuziehen, statt über Schnitzel, Milch, Pferde zu reden? In diesem Fall nicht. Denn unsere Methode weist Wörtern der Alltagssprache eine hohe Relevanz für die Theoriebildung zu, da Alltagssprache einerseits unmittelbar und ohne theoretischen Hintergrund verstanden wird, aber im Unterschied zur Fachsprache eben noch nicht begrifflich sauber verfasst ist. Um exemplarisch einen Weg von der Alltags- zur Fachsprache aufzuzeigen, können wir nicht schon mit einer Fachsprache beginnen.

verspricht, das abstrakte Business-Problem der Taxonomie-Konstruktion leichter, schneller und schöner lösbar zu machen.

Abstrakt ist dieses Problem, weil es von konkreten Rahmenbedingungen abstrahiert (lat. ab-: weg-; und lat. trahere: ziehen, schleppen). Wir verstehen Hevner 2004 so, dass es allerdings genau die konkreten Rahmenbedingungen sind, die ein Business Problem im Sinne einer Design-Science-Forschung konkret bearbeitbar und insbesondere auch konkret evaluierbar machen. Tatsächlich beschäftigt uns nicht das abstrakte Business Problem, sondern eine Reihe von ähnlichen, *konkreten* Business Problemen. Diese sind dadurch definiert, dass teils systematische, teils zufällige Rahmenbedingungen zu einem konkreten angewandten Problem zusammengewachsen sind. (lat. con-: zusammen; lat. crescere: wachsen).

In unserem *ersten konkreten Business-Problem* erarbeiten sich Studierende den Inhalt eines Lehrbuches zunächst als Mindmap, um dann Teile dieser Mindmap in eine höherwertige Wissensrepräsentation wie insbesondere eine Terminologie oder eine vollwertige Taxonomie überführen. Der Artefakt-Typ „Terminologie“ aus Hevner 2004 ist in der Didaktik im Rahmen der revidierten Lernzieltaxonomie von Anderson und Krathwohl als Lernzieltyp in der *knowledge dimension* auf der ersten und zweiten Stufe angesiedelt und damit Grundlage vieler höherer Lernprozesse: „*A*: Factual Knowledge. *Aa:* Knowledge of terminology. *[...] B:* Conceptual Knowledge […] *Ba:* Knowledge of classifications and categories.“ ([Kra02], p.214).

In der Tat besteht eine wesentliche Aufgabe akademischer Bildung darin, eine analytisch saubere Begriffsbildung voranzutreiben. Vorbildhaft wollen wir hier als Lehrende vorangehen: In unserem *zweiten konkreten Business Problem* sind wir Lehrenden es, die als Domänen-Experten die Begrifflichkeit eines Faches als Taxonomie formalisieren und Studierenden visualisiert als Lehr- und Referenzmaterial zur Verfügung stellen. Abbildung 1: Beispiel Schnitzel zeigt die Formalisierung dieses Beispiels in GenDifS.

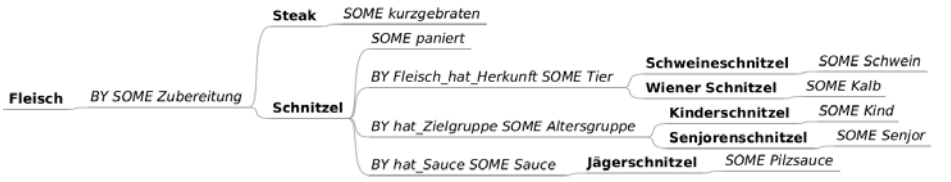

Abb. 1: Beispiel Schnitzel

Die Gemeinsamkeit unserer konkreten Business-Probleme besteht darin, dass einzelne Menschen im Zuge ihres Lernens und Lehrens sich eine Wissensdomäne erarbeiten und mit einem Mindmap-Tool strukturiert zusammenfassen. Typische Entwicklungsstadien der entstehenden Wissensrepräsentationen sind (a) eine gewöhnliche Mindmap als eine im wesentlichen baumartig strukturierte und oft mit reichhaltigem Layout, Icons etc. gestaltete Sammlung kleiner Informationseinheiten; (b) ein Glossar als eine Sammlung von Begriffen und Erläuterungen; falls in den Erläuterungen auf andere Glossarbegriffe Bezug genommen wird, werden diese Bezüge z.B. durch eine Verlinkung deutlich gemacht, es entsteht ein Wiki; (c) ein Begriffssystem als semiformale oder formale Strukturierung der Bezüge der Glossarbegriffe untereinander, insbesondere als Taxonomie.

Eine Analyse der intendierten konkreten Business Probleme ergibt folgende konkrete Anforderungen: Die GenDifS-Mindmap soll (a) über möglichst alle Phasen des Ontology-Engineering-Prozesses (gewöhnliche Mindmap, Glossar, Taxonomie) verwendbar sein und (b) alle Phasen nach- und nebeneinander ohne Medienbruch unterstützen. (c) Die Mindmap soll von Domain-Experten, die eine leichtgewichtige Einführung in Semantic-Web-Technologien erhalten haben, mit aktiver Sprachkompetenz in Eigenverantwortung aktiv produziert (d.h. geschrieben) werden können, und (d) insbesondere in der akademischen Lehre von Studierenden nach einer kurzen Einführung in GenDifS passiv rezipiert (d.h. gelesen) werden können.

1.2 Lösungsidee zum einleitenden Beispiel

Ursprünglich hatten wir nur einziges, etwas unspezifisches Artefakt als Entwicklungsziel vor Augen, nämlich eine domänenspezifische „Sprache“ – was auch immer man sich unter „Sprache“ vorstellen mag. Im Zuge der Überlegungen, wie man diese Sprache im Sinne von Hevner (2004) mit *Rigor* evaluieren könnte, ergaben sich viele Möglichkeiten, viele Abhängigkeiten, viele verdeckte Variablen, und auch immer mehr philosophische Fragen.

Schnell wurde klar, dass das scheinbar monolithische Artefakt „Sprache“ sich tatsächlich aus mindestens vier z.T. sehr unterschiedlichen Teil-Artefakten zusammensetzt. Im Einzelnen haben wir vor uns: a) eine *Methode*, um mit dieser Sprache Taxonomien zu entwickeln; b) eine kleine Menge von weiteren *Modellierungspatterns*, die zusätzlich zum grundlegenden GPDS-Modellierungspattern erst die angestrebte Praxisrelevanz sicherstellen; c) eine *konkrete Syntax und Semantik* für die Notation dieser Modellierungspatterns in einer Mindmap; d) eine prototypische *Implementierung* dieser Sprache in Form eines Skriptes, das eine GenDifS-Mindmap insbesondere in eine OWL-Ontologie und einen SKOS-Thesaurus übersetzt.

Im vorliegenden Aufsatz stellen wir die Kernelemente, namentlich die Modellierungspatterns sowie die Syntax und Semantik, von GenDifS dar.

2 Stand von Forschung und Technik

Methoden: Eine Übersicht über Ontology-Engineering-Methoden geben z.B. [SP21] und [IMMS13] sowie [BSWZ07], [SSS04]. Es fällt auf, dass die Mehrzahl der Methoden davon ausgeht, dass man sich schon in frühen Phasen einer Modellierung für ein Tool und/oder eine Modellierungssprache entscheiden muss – typischerweise RDF(S), SKOS oder OWL, wie z.B. auch in [SGL12]. Allerdings widerspricht dies u.E. Tom Grubers Design-Kriterium des *minimal encoding bias*: „The conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding. An *encoding bias* results when representation choices are made purely for the convenience of notation or implementation. Encoding bias should be minimized, because knowledge-sharing agents may be implemented in different representation systems and styles of representation." [Gru95].

Die Sprache GenDifS adressiert das Designkriterium des *minimal encoding bias* im Kern, indem sie die Wahl zwischen RDF(S), OWL oder SKOS nicht schon zur Modellierungszeit erfordert, sondern OWL und SKOS gleichzeitig erzeugt, und zwar in einer Weise modularisiert, in der das OWL- und das SKOS-Modul sowohl stand-alone wie auch in beliebiger Kombination gemeinsam verwendet werden können.

Modellierungspatterns: Mit dem GPDS-Modellierungspattern zur Modellierung von Wissensstrukturen können wir uns auf einen Stand der Technik berufen, der seit mehr als 2000 Jahren besteht. In [SKOS] erkennen wir das GPDS-Modellierungspattern auch in sogenannten Collections wieder. Tatsächlich gab der Baum des Porphyr in Verbindung mit den Collections des Milch-Beispiels aus dem SKOS-Primer den maßgeblichen Anstoß zur Entwicklung von GenDifS. Das Schema von SKOS würdigt die Bedeutung dieses Ordnungsprinzips, indem es dazu eine eigene Klasse `skos:Collection` mit zugehörigen Relationen wie `skos:member` bereithält. In GenDifs schlagen sich Collections in `BY SOME` als maßgebliches zentrales Modellierungspattern nieder.

Neben dem Export einer GenDifS-Mindmap in einen SKOS-Thesaurus suchen wir auch eine Lösung für einen Export in eine Taxonomie, die in der Semantic-Web-Sprache OWL2 formuliert ist und mit einem geeigneten OWL-Reasoner eine Klassierung von Objekten anhand ihrer Attribut-Wert-Paare ermöglicht. Für unser abstraktes Business Problem verfügen wir in der W3C-Publikation [Gro05] mit dem *Pattern 2: Values as subclasses partitioning a feature / using a fact about the individual* über eine OWL-Codierung, die genau der gesuchten Formalisierung entspricht. Dieser Stand der Technik löst das abstrakte Business Problem, nicht jedoch unsere zwei konkreten Business Probleme. Denn das zitierte Modellierungspattern mit einem Tool wie Protegé [Mus15] direkt in OWL zu formulieren ist selbst für Experten kompliziert, aufwändig und fehleranfällig. Für unsere Zielgruppen ist dieses Vorgehen gänzlich ungeeignet.

Der Lösungsansatz für GenDifS besteht darin, (a) das GPDS-Modellierungspattern in Gestalt der `skos:Collection` mit der existierenden formalen Lösung des W3C für *OWL value sets* konzeptuell zusammenzuführen, und (b) die erforderlichen Informationen in einer geeigneten Syntax und Semantik formal derart darzustellen, dass (c) ein Compiler eine auto-

matische Übersetzung nicht nur nach SKOS, sondern auch nach OWL2 vornehmen kann.

Mindmaps und Ontologien: Mindmap-basierte Notationen, die eine Ontologie oder eine vollwertige Taxonomie erzeugen, konnten wir nicht recherchieren. Zwar lassen sich aus einer Mindmap mit geringem Aufwand Klassenbäume erzeugen, die sich aber mangels Informationsgehalt nicht zur Klassierung von Instanzen anhand von Eigenschaften nutzen lassen. „Nackte“ Klassenhierarchien sind in der Welt der Taxonomien nur halbfertige Gebilde. Reine RDF(S)-Tools, Thesaurus- oder SKOS-Editoren sind i.A. zu ausdrucksschwach, um eine Taxonomie so zu notieren, dass eine Klassierung erfolgen kann.

Am ehesten scheint UML die Modellierung einer Ontologie oder einer Taxonomie zu unterstützen. Eine genaue Durchsicht der in dem Survey [MNN20] genannten Tools zeigt allerdings, dass mit UML zwar Klassenhierarchien, nicht aber die zur Klassierung speziell benötigten, hier beschriebenen Strukuren notiert werden können. Auch unser eigenes, um 2010 entwickeltes Tool *semAuth* ([Bus14]) kann nur bedingt die für Taxonomien typische Klassierung leisten. Zwar lassen sich die für Klassierungen notwendigen und hinreichenden Bedingungen durch F-Logik ([KLW95]) sehr elegant modellieren. Relationen hatten wir jedoch damals als Domain und Range modelliert, was wir heute als Fehler interpretieren.

Unsere **prototypische Implementierung** `gendifs.py` greift möglichst weitgehend auf existierende Technologie zurück. GenDifS hat kein eigenes GUI, sondern liest die von den Mindmap-Tools *Freemind* oder *Freeplane* nativ verwendete XML-Datei `*.mm` ein. Aus den Mindmap-Knoten wird zunächst der Code im Turtle-Format erzeugt, wie er auch in diesem Aufsatz abgedruckt ist. Dieser Code wird anschließend durch `rdflib` [BOP+20] wieder eingelesen. Ivan Hermans Reasoner `owlrl` [Her14] berechnet dann mittels Forward-Inferencing den deduktiven Abschluss aller über OWL-RL und RDF(S) erreichbaren Tripel. Neu entwickelt wurde also alleine die Generierung von `ttl`-Code auf Grundlage ausgewählter Mindmap-Knoten im XML-Format.

Tatsächlich glauben wir, mit der hier vorliegenden Erstpublikation von GenDifS eine Innovation anzubieten, die es so bisher noch nicht gab.

3 Methodologie

Wir verorten die Entwicklung von GenDifS als typische Design-Science-Research (DSR) ([HMPR04]), und zwar in unserem Fall spezifisch an einer Hochschule für angewandte Wissenschaften. Unsere programmatische Anwendungsorientierung schlägt sich u.A. darin nieder, dass wir zwar ein abstraktes Business-Problem lösen wollen, dieses aber vorwiegend aus der Perspektive einer Gruppe von konkreten Business-Problemen bearbeiten.

Wann und an welchen Schnittstellen in einem DSR-Prozess Evaluation stattfinden kann und soll, wird in der DSR uneinheitlich gesehen [BHM20]. Breite und Offenheit des Evaluationsbegriffs spiegeln sich in der Vielfalt der Evaluationsmethoden wider. Wir wollen Evaluation im weitesten Sinn verstehen als den systematischen Abgleich von Ist und Soll, von Vorstellung, Modell und Wirklichkeit. Hevner 2004 beschreibt drei Cycles, in denen durch Evaluation Rigor unterstützt wird. Der Relevance- (R-) Cycle stellt sicher, dass das entwickelte Artefakt tatsächlich das richtige Problem löst. Der Knowledge-Base- (KB-) Cycle stellt den Bezug zu Theorien, Methoden und dem Stand von Wissenschaft und Technik her, und der Build-and-Evaluate (BE-) Cycle *„is the heart of any design science research project. This cycle of research activities iterates more rapidly between the construction of an artifact, its evaluation, and subsequent feedback to refine the design further“* ([HMPR04], p90f).

Es stellt sich die Frage: Wie gestalten wir im Kontext von GenDifS die Evaluation im BE-Cycle und behalten gleichzeitig die Relevanz und das bestehende Wissen im Blick? In unserer Entwicklung greifen der R- und der BE-Cycle durch die Auswahl unserer Modellierungsbeispiele eng ineinander, während der KB-Cycle existierende Lösungsansätze beisteuert. Speziell verzahnen sich BE- und R-Cycle über die Auswahl der zu modellierenden informellen Wissensrepräsentationen, die für die spezifisch gewählten konkreten Business-Probleme typisch sind. Im BE-Cycle modellieren wir Beispiele in GenDifS und beurteilen das Modell subjektiv, um darauf aufbauend die verwendeten Sprachkonstrukte anzupassen und weiterzuentwickeln.

Es schließt sich der Test auf Wartbarkeit an: Eine weitere Informationsquelle wird hinzugenommen, das Modell dann bearbeitet, erweitert und vor allem korrigiert. Mit diesem Vorgehen testen wir auch die eingangs beschriebene Anforderung, der zufolge eine schrittweise Weiterentwicklung einer Ontologie möglich sein muss.

4 Ergebnisse

Die maßgebliche Motivation für GenDifS ist das Konzept der Klasse `skos:Collection`, das im SKOS-Primer [SKO] unter der Überschrift *4 Advanced SKOS: When KOSs are not Simple Anymore > 4.1 Collections of Concepts* in dem folgenden Milch-Beispiel veranschaulicht wird:

```
milk
  <milk by source animal>
    cow milk
    goat milk
    buffalo milk
```

In GenDifS wird dieses Beispiel wie in Abb. 2 formalisiert:

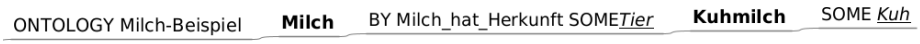

Abb. 2: Beispiel Milch

Offensichtlich sind in diesem Beispiel zwei parallele Teilmengen-Beziehungen enthalten: Die erste, in Fettdruck wiedergegebene Beziehung `Kuhmilch rdfs:subClassOf :Milch` spannt insgesamt den für Mindmaps typischen Baum auf. Mit diesem Baum eng verflochten ist aber auch eine zweite, in der Abbildung in unterstrichenem Kursivdruck wiedergegebene Beziehung `:Kuh rdfs:subClassOf :Tier` enthalten. GenDifS interpretiert obige Mindmap als zwei zusammengehörende, ineinander verflochtene Bäume. Dieses Syntax-Detail ist eine Folge von explorativem, spielerischem Modellierungshandeln und entspringt der Beobachtung, dass Unterscheidungen in einem Sekundärbaum bisweilen einen Primärbaum strukturieren.

Wir skizzieren die formale Semantik der Sprache, indem wir exemplarisch die Übersetzung ausgewählter GenDifS-Idiome in das Format Turtle (`*.ttl`) angeben.

4.1 BY … SOME

Kern der Sprache GenDifS ist das Modellierungspattern *Genus Proximum, Differentia Specifica (GPDS)*, also die Angabe eines Oberbegriffs plus Charakterisierung einer Teilmenge durch ein charakteristisches Merkmal (hier: ein charakteristisches Attribut-Wert-Paar). Dieses Modellierungspattern unterscheidet eine Taxonomie von einer reinen Klassenhierarchie und erlaubt die gesuchte Klassierung in Form eines Top-Down-Inferencings von der allgemeinen Klasse zur speziellen Klasse. Es wird in GenDifS durch `SOME ...` als Enkel-Knoten eines `BY ... SOME ...` implementiert.

Beispiel: *Wenn X eine Milch ist, und X als Herkunft eine Kuh hat, dann ist X eine Kuhmilch.* Der in obiger Abb. 2: Beispiel Milch umrandete Knoten `SOME Kuh` wird vom Compiler in folgenden Code übersetzt:

```
# namespaces to be aligned by the user
@prefix ex: <http://some.locati.on/namespace/ex#> . # OWL A-Box
@prefix : <http://some.locati.on/namespace/default#> . # OWL T-Box

# SOME.1, owl
:Kuh
   rdfs:subClassOf :Tier .

# SOME.2, owl-classification
:Milch_hat_Herkunft_SOME_Kuh a owl:Class ;
   owl:equivalentClass [ a owl:Restriction ;
      owl:onProperty :Milch_hat_Herkunft ;
      owl:someValuesFrom :Kuh ] ;
   rdfs:subClassOf :restrictions_BY .

# SOME.3, owl-classification
:Milch_AND_Milch_hat_Herkunft_SOME_Kuh a owl:Class ;
   rdfs:subClassOf :Kuhmilch ;
   owl:equivalentClass [ a owl:Class ;
      owl:intersectionOf ( :Milch :Milch_hat_Herkunft_SOME_Kuh ) ] .
```

In Prädikatenlogik entspricht dies der Formel `∀ X [ Kuhmilch(X) ← Milch(X) ∧ ∃ Y Kuh(Y) ∧ Milch_hat_Herkunft(X,Y) ]`, die zugleich auch als Definition der formalen Semantik dieses Konstruktes dient. Die GenDifS-Website erklärt an eben diesem Milch-Beispiel im Detail das zugehörige Inferencing.

Ein in GenDifS formuliertes Modell kann in verschiedene Module exportiert werden, die per Namespace unterschieden werden und beliebig miteinander kombiniert werden können. RDF-Resources mit dem Defaultnamespace `:` (Doppelpunkt) definieren wie oben gezeigt die Klassen (die T-Box) aus dem *Taxonomie-Modul*. RDF-Resources mit dem Namespace `ex:` sind automatisch angelegte Instanzen (eine A-Box) der OWL-Klassen und bilden das *Example-Modul*, das eine testgetriebene Entwicklung der Ontologie unterstützt.

```
# SOME.8, owl-test
ex:Milch_ID_571743754
   a :Milch ;
   :Milch_hat_Herkunft ex:Kuh_ID_571743754 ;
   gendifs:classifyLike ex:Kuhmilch_ID_571743754 .
ex:Kuhmilch_ID_571743754 a :Kuhmilch .
ex:Kuh_ID_571743754 a :Kuh .
```

Für die Instanz `ex:Milch_ID_571743754` ist zu erwarten, dass sie aufgrund ihrer ebenfalls exemplarisch angegebenen Attribute und Werte gleich wie die Instanz `ex:Kuhmilch_ID_571743754` klassifiziert wird. Weil diese `ex`-Beispiele aus Klassenname und einer ID zusammengesetzt sind, können als Nebeneffekt auch in der GUI von Protegé taxonomische Kategorien und exemplarische Instanzen klar unterschieden werden.

Im *SKOS-Modul* werden Konzepte eines Thesaurus als Instanzen der Klasse `skos:concept` angelegt. Diese Konzepte (die natürlich nicht selbst Elemente des SKOS-Schema sind, denn wir wollen ja nicht das SKOS-Schema selbst erweitern) halten wir im Namespace `cpt` zusammen:

```
@prefix cpt: <http://some.locati.on/namespace/cpt#> . # concepts
# SOME.9, skos
cpt:Kuh
   rdf:type skos:Concept ;
      skos:broader cpt:Tier .
```

Das durch `BY ... SOME` realisierte Modellierungspattern bildet zwar den Kern der Sprache GenDifS, ist für sich genommen aber nicht ausdrucksstark genug, um wichtige praktische Anforderungen in der Taxonomie-Entwicklung abzudecken. Welche der vielen zusätzlich möglichen Modellierungspatterns tatsächlich in GenDifS aufgenommen werden sollen sind Design-Entscheidungen der Sprache, die sich in einem verzahnten R- und BE-Cycle bewähren müssen oder verworfen werden. Einige Sprachelemente dienen primär der Anschlussfähigkeit und Akzeptanz (z.B. REL), andere ergänzen das GPDS-Paradigma in idealer Weise (z.B. SUP). Wieder andere vermisst man, wurden aber absichtlich nicht implementiert, um die Komplexität der Sprache zu reduzieren. Wir diskutieren im Folgenden exemplarische Sprachelemente.

4.2 Superclass (SUP)

Die Bedeutung von SUP lässt sich am neuen Beispiel „Rapphengst“ gut zeigen: Pferde lassen sich u.A. nach Geschlecht, Fellfarbe unterscheiden; es gibt Heiß-, Warm- oder Kaltblüter; Mutationen des sog. Grey-Gens sorgen für das Ausschimmeln des Fells, usw. Viele dieser Kombinationen haben eigene Klassenbezeichner wie z.B. Rapphengst oder Schimmelstute. Ausschließlich baumartige strukturierte Wissensrepräsentationen kommen hier an ihre Grenze, da mit zunehmender Baumtiefe die Kombinatorik zuschlägt: männliche/weibliche Pferde, schwarze/weiße Pferde, warmblütige männliche schwarze Pferde usw.

Im oberen Teilbaum der Abb. 3: Beispiel Rapphengst wird die Modellierung in einer Mindmap schwierig: Wenn eine Klasse wie in unserem Beispiel `Rapphengst` an verschiedenen Stellen redundant erstellt würde, wäre dies ein Fehler in jeder auf Verständnis und Wartbarkeit ausgerichteten Wissensrepräsentation. (Das rote X-Icon schließt in GenDifS die entsprechenden Teilbäume aus der Übersetzung aus.) Da aber ein Wechsel zu einem in der Darstellung mächtigeren Graph-Tool aus verschiedenen Gründen nicht in Frage kommt und insbesondere auch nicht zu unseren konkreten Business Problemen passen würde, wollen wir an der Mindmap-Darstellung festhalten und innerhalb dieser Wissensrepräsentation eine Problemlösung suchen.

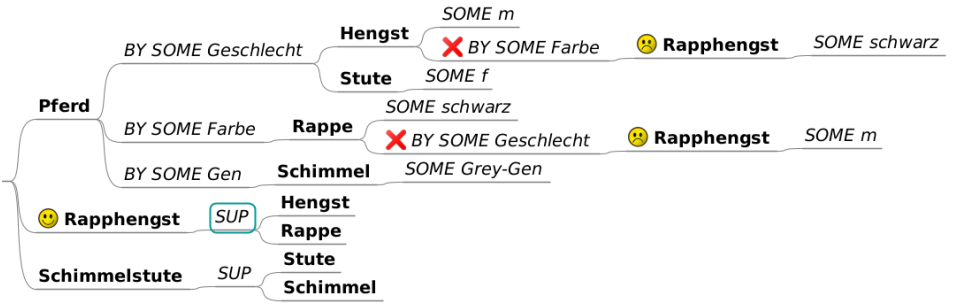

Abb. 3: Beispiel Rapphengst

Eine Lösung bietet die Idee des inversen Baums: Durch das Sprachelement SUP (Superclass) werden die *Kindknoten* eines Knotens (a) nicht wie sonst in Ontologie-Editoren (z.B. auch Protegé) als Unterbegriffe, sondern als *Oberbegriffe* einer Klasse interpretiert, wobei (b) ihre Schnittmenge insgesamt als eine *hinreichende Bedingung* für die aktuelle Klasse interpretiert wird. Für den in Abb. 3: Beispiel Rapphengst gekennzeichneten Knoten SUP erzeugt der Compiler folgenden Code:

```
# SUP.1, owl-classification
:Hengst_AND_Rappe a owl:Class ;
   rdfs:subClassOf :Rapphengst ;
   owl:equivalentClass [ a owl:Class ;
      owl:intersectionOf ( :Hengst :Rappe ) ] .

# SUP.2, owl-test
ex:Hengst_AND_Rappe_ID_693686695
   a :Hengst ;
   a :Rappe ;
   gendifs:classifyLike ex:Rapphengst_ID_693686695 .
ex:Rapphengst_ID_693686695 a :Rapphengst .
```

In Prädikatenlogik entspricht dies der Hornklausel ∀ X [Rapphengst(X) ← Rappe(X) ∧ Hengst(X)], die auch die formale Semantik dieses Sprachkonstrukts definiert.

SUP ergänzt das zentrale Modellierungspattern BY ... SOME in mehrfacher Weise. So hebt die „Richtungsumkehr" der Leserichtung von Klassendefinitionen die Einschränkung von Mindmaps, nur Bäume und keine Graphen darstellen zu können, in interessanter Weise auf. Mit SUP lässt sich ein Graph als eine Kombination aus sich verzweigenden und zusammenführenden Wegen kreuzungsfrei als Baum repräsentieren.

SUP ergänzt das GPDS-Modellierungspattern aber nicht nur in der Mindmap-Visualisierung, sondern auch theoretisch. Wenn wir mit dem GPDS-Modellierungspattern eine Kategorie durch ein Genus Proximum und eine Differentia Specifica definieren, dann liegt eine sogenannte *intensionale Definition* vor, in der die Bedeutung eines Begriffs im Wesentlichen durch ein charakteristisches Merkmal definiert wird. SUP gesellt zu dieser Definitionsart die sogenannte *Nominaldefinition* hinzu, in der neue Begriffe extensional allein durch die Schnittmengenbildung anderer, bereits definierter Begriffe konstruiert werden. Der charakteristische Unterschied liegt im „klassifikatorischen" Informationsgehalt: Bei intensionalen Definitionen wird ein neues charakteristisches Merkmal eingeführt, mit dem eine neue Unterscheidung getroffen werden kann. Dazu im Gegensatz sind Nominaldefinitionen „merkmalsfrei" in dem Sinn, dass sie keine neue Information hinzufügen, mit der man neue Unterscheidungen treffen könnte. Der alleinige, aber in der Praxis bedeutsame Wert von Nominaldefinitionen besteht darin, wichtige oder häufige Schnittmengen von bereits definierten Begriffen prägnant zu benennen.

4.3 REL und REV

Die eingangs erwähnte Forderung, insbesondere auch die Frühphasen semantischer Modellierungen zu unterstützen, macht es unerlässlich, an weit verbreitete Modellierungsgewohnheiten wie insbesondere auch Semantische Netze anschließen zu können. Dazu stellt auch GenDifS das Sprachelement `REL` (und das aus html bekannte inverse `REV`) bereit, das typisch ist für gerichtete Graphen mit gelabelten Kanten. Ein Semantisches Netz lässt sich geradlinig in einen SKOS-Thesaurus übersetzen. Die Übersetzung nach RDF(S) oder OWL bereitet hingegen Probleme, die auf der GenDifS-Website ausführlicher diskutiert werden.

Wir erwähnen die Probleme um das Konstrukt `REL` hier zumindest abstrakt, um eine wichtige Frage zu motivieren: Wie will man am Markt der Modellierungssprachen eine neu konstruierte Sprache zu anderen existierenden Sprachen positionieren? Wir verfolgen eine um Ergänzung bemühte Nischen-Strategie: GenDifS will mit anderen Wissensrepräsentationen nicht konkurrieren, sondern bestehendes Wissen, das in Form z.B. eines Semantischen Netzes, eines ER-Diagramms, eines BPMN-Modells etc. modelliert wurde, als relevant anerkennen, und bei Bedarf einzelne semantische Unbestimmtheiten aus dem Blickwinkel einer Taxonomie genauer klären.

4.4 Nicht realisierte Sprachelemente

Auch wenn es in Bezug auf Syntax, Semantik und Implementierung technisch möglich ist (und die Versuchung groß war), haben wir andere naheliegende Sprachkonstrukte nach reiflichem Überlegen *nicht* realisiert. Dazu gehören insbesondere alle Aspekte in Klassifikationen, die explizit oder implizit eine Negation enthalten (wie insbesondere ONLY oder DISJOINT), da diese ganz wesentlich die Komplexität erhöhen würden.

5 Evaluation

Eine der wichtigsten Eigenschaften von Design-Forschung ist eine systematische Evaluation als Beitrag zu *Rigor*, d.h. einer „sauber dokumentierte[n] wissenschaftliche[n] Herleitung anhand von anerkannten Kriterien“ ([OsterleWB10]). Wie im Abschnitt Methodologie dargestellt, siedelt Hevner 2004 Evaluation im oben eingeführten Relevance- (R-), Knowledge-Base- (KB-) und Build-and-Evaluate- (BE-) Cycle an. Wir stimmen vom Brocke ([SB12], [BHM20]) zu, für den eine Vielzahl unterschiedlicher Evaluationen den Mörtel zwischen allen Stufen im Design-Science-Research-Prozess darstellt.

Unsere eigene Evaluation siedelt sich bisher vorwiegend im hochfrequenten BE-Cycle an. Anhand vieler minimalistischer Beispiele motivierten und bewerteten wir einzelne Design-Entscheidungen insbesondere zum Teilartefakt Syntax und Semantik. Im BE-Cycle entdeckten wir immer wieder typische Problemlagen, interessante Modellierungspatterns und auch eigene blinde Flecken. (So kam etwa das Sprachkonstrukt `SUP` erst relativ spät zu GenDifS hinzu.) Faktisch sind wir in diesem Cycle so vorgegangen: (1) Identifiziere hoch vorstrukturierte Sachtexte oder existierende Modelle; (2) stelle diese als GenDifS-Mindmap dar; (3) beurteile die Mindmap und das erzeugte Modell. Das Evaluationsinteresse war hier: Wie geradlinig lässt sich der dargestellte Text in voll formalisierte Statements übersetzen? Wie wird die GenDifS-Methode unterstützt, d.h. lassen sich unklare Beziehungen zunächst vorläufig darstellen, jedoch so, dass der Übergang zur vollen Formalisierung leicht möglich ist? Wie könnten die existierenden Modellierungspatterns und Sprachkonstrukte von GenDifS verbessert werden?

Im Kern stellt sich die Frage, was man mit einem bestimmten Evaluations-Arrangement tatsächlich evaluiert. Denn wie in der Einleitung dargestellt, haben wir ja in unserem Fall vier Teil-Artefakte – Modellierungsmethode, Modellierungspatterns, konkrete Syntax und Semantik, Tool und Implementierung – vor uns, die eng zusammenhängen und ein komplexes System bilden. Wie evaluieren wir die Sprache nicht indirekt über die damit erzeugte Ontologie und auch nicht das Tool, und auch nicht die Kompetenz einer konkreten Nutzerin, sondern tatsächlich die Sprache? (Und wir wiederholen uns: Was immer man sich unter „Sprache“ genau vorstellen will?)

Verständlichkeit und Verwendbarkeit einer Sprache wie GenDifS in konkreten Business-Cases haben wir in Selbstanwendung ja auch schon bewertet und können uns auch ein Evaluationsdesign unter Rigor-Anforderungen vorstellen. Eine direkte Evaluation der Sprache für den abstrakten Usecase scheint uns evidenzbasiert dagegen nur schwer vorstellbar. Statt auf empirische Untersuchungen würden wir hier eher auf einen argumentativen Expertendiskurs setzen, im Sinne der Design-Science-Research also Bezüge zur Knowledge-Base herstellen. Ein wichtiges Kriterium für die *direkte Evaluation* einer Ontologiesprache haben wir mit Tom Grubers *minimal encoding bias* schon kennengelernt. Durch die leitende Idee von GenDifS, eine Taxonomie in einer abstrakten Baum-Darstellung zu modellieren, die keine frühe Entscheidung für eine Encodierung in OWL, RDF(S) oder SKOS mit der damit jeweils automatisch verbundenen Semantik erfordert, sowie die Selbstbeschränkung auf wenige Sprachkonstrukte, die die Auswahl von Rea-

sonern und OWL2-Profilen einfacher macht, kommen wir diesem Kriterium näher als andere Ontologie-Notationen.

Als anderes wichtiges Kriterium für die direkte Evaluation von GenDifS erachten wir die Kompatibilität mit anderen Modellierungssprachen. GenDifS konzentriert sich auf das Kerngeschäft Taxonomie und spielt hier seine Stärken aus. Bezüglich dem praxisrelevanten Evaluationskriterium der Anschlussfähigkeit war uns wichtig, keine Konkurrenz, sondern eine friedliche Koexistenz und Komplementarität mit Darstellungen wie ER-Schemata oder UML-Klassendiagrammen herzustellen. Mit der Mindmap-Notation schließt GenDifS nahtlos an andere etablierte Ökosystemen des nicht voll formalisierten Wissensmanagements an und erlaubt den Brückenschlag in die Welt der vollständigen logisch-semantischen Formalisierung.

6 Methodenkritik

Bei der Suche nach geeigneten Evaluationsmethoden direkt zur Sprache sind wir allerdings auf ein größeres Problem gestoßen: Je länger wir uns mit dem Thema „Evaluation einer neuen Sprache“ befasst haben, desto weniger glauben wir, dass ein Artefakt vom Typ Sprache in ähnlicher Weise wie andere Artefakt-Typen gemäß dem Design-Science-Paradigma nach Hevner 2004 mit Rigor entwickelt und evaluiert werden kann. Was ist in Bezug auf unser aus Methode, Modellierungspatterns, Syntax und Semantik, Tool und Implementierung bestehendes System gemeint mit dem Anspruch „Wir evaluieren eine Sprache“? In der DSR besteht Konsens, dass eine Artefaktentwicklung natürlich ein zunehmend gutes Verständnis der Problemstellung erzeugt:

In the early stages of a discipline or with significant changes in the environment, each new artifact created for that discipline or environment is “an experiment” that “poses a question to nature” (Newell and Simon 1976, p 114). [HMPR04]

Im besonderen Fall, in dem das Artefakt eine neue Sprache ist und eine solche Sprache genau dann gelungen ist, wenn man mit ihr bestimmte Dinge schöner, konziser, eleganter, differenzierter etc. ausdrücken kann: Dann verändert solch eine Sprache unsere Sicht auf die Welt. Konsequenterweise müsste man Artefakte vom Typ Sprache danach evaluieren, in welcher Hinsicht sie eine neue Weltsicht erzeugen. Inwiefern diese Evaluation aber mit Rigor erfolgen kann: Dazu gibt Design-Science-Research, solange sie im Sinne von Thomas Kuhn als Normalwissenschaft betrieben wird, keine Anleitung.

Ob im speziellen Fall einer neuen Ontologie-Sprache auch eine neue philosophische Ontologie folgt, können wir nicht beurteilen. Aber wenn es einen Unterschied macht, ob man ein Begriffssystem in RDF(S), in SKOS, in OWL oder eben auch in GenDifS formalisiert: Dann hat auch GenDifS das Potential, einen Unterschied zu machen – nicht die schlechteste Eigenschaft für eine Sprache, die die Unterscheidung von Unterscheidungen zu ihrer Sache macht.

Literatur

[BOP+20] Boettiger,C; Ooms, J; Leinweber, K; Hester, J: rdflib v0.2.3. Zenodo, January 2020. doi:10.5281/ZENODO.1098478. Stand 2022 aktuell ist RDFlib 6.1.1, https://rdflib.dev/ (2022-06-25)

[BSWZ07] Braun, S; Schmidt, A.P.; Walter, A.; Zacharias, V.: The Ontology Maturing Approach for Collaborative and Work Integrated Ontology Development: In Chen, L. et al (ed.): Proceedings of the First International Workshop on Emergent Semantics and Ontology Evolution, CEUR-WS.org, 2007.

[Bus14] Busse, J.: Semantische Modelle Mit Mindmaps. In Keller, S.A.; Schneider, R.; Volk, B. (Hrsg.): Wissensorganisation und -Repräsentation mit digitalen Technologien, S. 115–127. DE GRUYTER, December 2014. doi:10.1515/9783110312812.115.

[FVO+20] Franco W.; Viktor, C.; Oliveira, A. et al: Ontology-based Question Answering Systems over Knowledge Bases: A Survey:. In Proceedings of the 22nd International Conference on Enterprise Information Systems, 532–539. Prague, Czech Republic, 2020. doi:10.5220/0009392205320539.

[Gro05] W3C Working Group. Representing Specified Values in OWL: „Value Partitions“ and „Value Sets“. https://www.w3.org/TR/swbp-specified-values/, May 2005. (2022-06-25)

[Gru95] Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? International Journal of Human-Computer Studies, 43(5-6):907–928, November 1995. doi:10.1006/ijhc.1995.1081.

[Her14] Herman, I.: Owl-RL, A Simple Owl2 RL Reasoner On Top Of Rdflib. Zenodo, October 2014. doi:10.5281/ZENODO.14543.

[HMPR04] Hevner, A.R.; March, S.T.; Park, J.; Ram, S.: Design Science in Information Systems Research. MIS Quarterly, 28(1):75–105, March 2004.

[IMMS13] Iqbal, R. et al: An Analysis of Ontology Engineering Methodologies: A Literature Review. Research Journal of Applied Sciences, Engineering and Technology, 6(16):2993–3000, September 2013. doi:10.19026/rjaset.6.3684.

[KLW95] Kifer, M.; Lausen, G.; Wu, J.: Logical foundations of object-oriented and frame-based languages. Journal of the ACM, 42(4):741-843, July 1995. doi:10.1145/210332.210335.

[Kra02] Krathwohl, D.R.: A Revision of Bloom‘s Taxonomy: An Overview. Theory into Practice, 2002. doi:10.1207/s15430421tip4104_2.

[MNN20] Mkhinini, M.M. et al.: Combining UML and ontology: An exploratory survey. Computer Science Review, 35:100223, February 2020. doi:10.1016/j.cosrev.2019.100223.

[Mus15] Musen, M.A.: The protégé project: a look back and a look forward. AI Matters, 1(4): 4–12, June 2015. doi:10.1145/2757001.2757003.

[SB12] Sonnenberg, C.; vom Brocke, J.: Evaluations in the Science of the Artificial – Reconsidering the Build-Evaluate Pattern in Design Science Research. In Hutchison, D. et al: Design Science Research in Information Systems. Advances in Theory and Practice, volume 7286, pages 381–397. Springer 2012. doi:10.1007/978-3-642-29863-9_28.

[SGL12] Suárez-Figueroa, M.G.; Gómez-Pérez, A.; Fernández-López, M.: The NeOn Methodology for Ontology Engineering. In Suárez-Figueroa, M.G. et al. (ed): *Ontology Engineering in a Networked World*, pages 9–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-24794-1_2.

[SKOS] SKOS Simple Knowledge Organization System Primer. W3C Working Group Note 18 August 2009. https://www.w3.org/TR/skos-primer/. (2022-06-25)

[SP21] Spoladore, D.; Pessot, E.: Collaborative Ontology Engineering Methodologies for the Development of Decision Support Systems. Case Studies in the Healthcare Domain. Electronics, 10(9):1060, April 2021. doi:10.3390/electronics10091060.

[SSS04] Sure, Y.; Staab, S.; Studer, R.: On-To-Knowledge Methodology (OTKM). In Staab, S. and Studer, R. (ed): Handbook on Ontologies, pages 117–132. Springer, 2004.

Die Online-Dokumentation zu GenDifS unter http://jbusse.de/gendifs/ beschreibt die hier verwendete Version 0.5 vom Sommer 2022. Das Handout zum GenDifS-Talk auf der Tagung AKWI 2022 findet sich unter http://jbusse.de/gendifs/akwi2022.html.